Journal of Membrane Computing (2020) 2:121–136
https://doi.org/10.1007/s41965-020-00041-3

REGULAR PAPER

Membrane computing with water

Thomas Hinze[1] · Hendrik Happe[1] · Alec Henderson[2] · Radu Nicolescu[2]

Received: 1 October 2019 / Accepted: 21 April 2020 / Published online: 25 May 2020

Abstract
We introduce water tank systems as a new class of membrane systems inspired by a decentrally controlled circulation of water or other liquids throughout cells called tanks and capillaries called pipes. To our best knowledge, this is the first proposal addressing the behavioural principle of floating and stored water for modelling of information processing in terms of membrane computing. The volume of water within a tank stands for a non-negative rational value when acting in an analogue computation or it can be interpreted in a binary manner by distinction of "(nearly) full" or "(nearly) empty". Water tanks might be interconnected by pipes for directed transport of water. Each pipe can be equipped with valves which in turn either fully open or fully close the hosting pipe according to permanent measurements whether the filling level in a dedicated water tank exceeds a certain threshold or not. We demonstrate dedicated water tank systems together with simulation case studies: a ring oscillator for generation of clock signals and for iteratively making available amounts of water in a cyclic scheme, analogue arithmetics by implementation of addition, non-negative subtraction, division, and multiplication complemented by systems in binary mode for implementation of selected logic gates.

Keywords Water-based computing · Autonomous operation · Arithmetics · Membrane system

1 Introduction

Visualisation of computing processes in detail is an important and demanding task for teaching to students of engineering. Having an experimental setup at hand, the students can observe and trace the course of signals throughout a computation. They understand the occurrence of delays and latencies in signal processing along with the necessity of synchronisation in order to keep signals in a valid state. Moreover, the students get sensibilised for imprecision and slight deviation of signal levels to cope with for successful hardware implementations. They learn about advantages and disadvantages of analogue and digital modes of computation and resulting consequences in algorithmic design and hardware development [15]. State-of-the-art computers utilise electrical signals. In spite of their preference in practice, electrical signals turn out to be invisible and more or less cumbersome in teaching at the beginners level.

In this context, it is an obvious idea to apply an alternative floating medium. Pure water in its liquid form seems to be an ideal candidate. On the one hand, underlying natural principles of floating water in comparison to the flow of electricity show a close relationship when replacing the current by volumetric flow rate and after substitution of voltage by pressure. On the other hand, floating and stored water enriched with a dye and passing through a macroscopic environment is easily visible.

Beyond its usefulness for teaching, such a system can be implemented in a decentralised manner inspired by a tissue of biological cells. Indeed, higher organisms incorporate mechanisms of computing with water, for instance the control loop for adjustment of the human intraocular pressure [10] or for setting a local specific pH environment by purposive dilution in which enzymes exhibit their maximum activity [21]. Regulated transportation of water enriched with

✉ Thomas Hinze
thomas.hinze@uni-jena.de

Hendrik Happe
hendrik.happe@uni-jena.de

Alec Henderson
ahen386@aucklanduni.ac.nz

Radu Nicolescu
r.nicolescu@auckland.ac.nz

[1] Department of Bioinformatics, Friedrich Schiller University Jena, Ernst-Abbe-Platz 2, 07743 Jena, Germany

[2] The University of Auckland, Private Bag 92019, Auckland 1142, New Zealand

reactive substances throughout and across tissues together with a blood stream complemented by temporal storage and processing within cells is a crucial part in numerous biological control loops [2, 12]. A first step towards corresponding models and simulation environments consists in consideration of systems for computing with water which promises a fascinating potential worth to be tackled.

The idea to employ water for information processing is not new. First attempts date back more than 2000 years. The ancient water cascade [14] for soil irrigation gives an illustrative example in which an initial pool of water becomes separated into nearly equal portions by passing a sequence of water tanks whose overflow forwards the water to the next tank after the portion has been collected; see Fig. 1. In the 1930s, the first real water-based analogue computer called water integrator had been invented in the Soviet Union and applied for numerical solution of ordinary differential equation systems [1, 13, 24]. This computer comes with an external manual control and utilises mechanical elements for individual regulation of the volumetric flow rate in each of the pipes. Fine-grained variation of flow rates can be seen as a characteristic attribute of this computer whose constituents comprise many expensive movable mechanical elements, but nearly no sensors for measurement of filling levels or other parameters. Later, hydraulic systems came into the focus to mimic a computer [3, 4, 20]. Their principle of operation is mainly based on adjustment of pressure throughout the system in order to gain the intended functionality [23, 25].

Tom Head and his co-workers successfully initiated and developed microflow systems [6, 7]. They have in common that they employ the flow of water in order to bring together substances for chemical reactions carried out in so-called micro-chambers. To do so, they are optimised to manage on a minimum of water. In addition to their microscaled size, they utilise single droplets of water able to hit each other causing an interaction [17, 22]. Microflow systems complement the water droplets by reactants and ingredients to achieve the capacity of information processing [5, 11]. Indeed, modelling of spatially distributed reaction systems

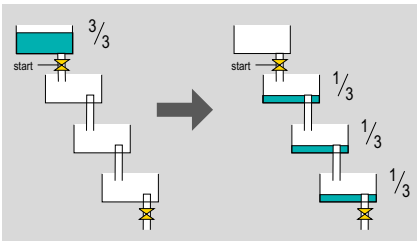


Fig. 1 Schematic representation of an ancient water cascade for separation of a water pool into three nearly equal portions by opening the start valve. Overflow pipes manage transportation of water to the next tank after a portion of $\frac{1}{3}$ has been completed

along with the exchange of molecules and their application for information processing is a keystone of membrane computing [16, 19]. More general, the field of membrane computing includes identification and formalisation of principles for information processing found in nature and in biological systems [18]. Although chemical reactions represent many aspects of biological information processing, decentrally controlled storage and circulation of water considered as a physical phenomenon without any chemistry embodies also an issue worth attracting attention within membrane computing.

Our approach to construct a water-based computer is inspired by the objective to have no central control, and instead, we envisage a distributed system in which the flow of water is exclusively managed by local measurements of filling levels. To this end, we introduce the framework of water tank systems as a new class of membrane systems. Its main ingredient is a finite number of water tanks. A water tank can contain an initial volume of water and it is able to store and to collect an amount of water up to a maximum volume. The volume of water in a tank serves as data carrier and as a medium of data processing by variation of this volume over time. To do so, water tanks might be interconnected via pipes. A pipe enables the directed transport of water from a tank to another one if opened. A pipe is allowed to be equipped with one or more freely configurable valves. Each valve either fully opens or fully closes its hosting pipe depending on the permanent measurement of the filling level in a dedicated water tank whether or not it exceeds a given threshold or indicates an empty or nearly empty tank. Please note that a valve is restricted to distinguish merely two states “fully open” and “fully closed”, there are no states in between. This setting allows for equal volumetric flow rates in all pipes within a water tank system opened for water transport and supplied by non-empty tanks. A pipe transports water if and only if all of its valves are fully opened and the tank acting as water supply is not empty. Moreover, the entrance of a pipe might be placed at an arbitrary filling level of its supply tank in a fixed position. So, this tank needs to contain a minimum amount of water before filling the pipe. The mentioned assumptions and prerequisites for our water tank systems have been selected in a way to keep the construction simple and free from any central control.

The resulting water tank systems autonomously operate in a decentralised manner based on local measurements of filling levels without the need of an external control. To our best knowledge, this is the first contribution addressing this type of behaviour for computing purpose. Since a water tank can be seen as a membrane permeable to inflow and/or outflow of water molecules whose presence is dynamically regulated by local measurements (interaction rules), our approach fits into the scope of membrane systems. For technical reasons, each water tank system comprises a special

water tank called reservoir which stands for a huge amount of water responsible for supply if needed and subsuming excessive water collected from overflow pipes of the tanks. For better visibility, we allow a water tank system to start its activities at a configurable point in time. So, the initial water volumes of the tanks can be noticed prior to successive modification. To do so, all outward flow pipes of water tanks acting as inputs or those initially non-empty and employed for auxiliary usage, consistently get a uniform start valve that opens as soon as the entire water tank system is set into operation.

A water tank system might operate either in analogue or in binary mode. In analogue mode, the volume of water within a tank represents a non-negative rational value. For this purpose, we introduce water tank systems for arithmetic operations addition, non-negative subtraction, division, and multiplication. They can be assembled for performing sequenced or nested computations. In addition, a ring oscillator consisting of a cyclic structure with at least three water tanks emulates a clock signal. In binary mode, an empty or nearly empty water tank refers to the logical value "0" and a full or nearly full one to "1" with latencies when the tank gets filled or emptied. We define logic gates OR, AND, and a bit duplicator. Water-based logic gates can be connected towards Boolean circuits equipped with the capability of inherent self-synchronisation without external control striving for construction of any register machine known to be a Turing-complete model of computation, shown for example (with chemical logic gates and an oscillating species concentration interpreted as clock) in [8]. All water tank systems presented throughout this paper come with simulation case studies revealing system's temporal behaviour in detail. The underlying simulation source code complements our ongoing project on a Java Environment for Nature-inspired Approaches (JENA) [9].

The paper is structured as follows: in Sect. 2, we familiarise the reader with the description and formal definition of water tank systems and their behaviour. Hereafter, three case studies selected from different application scenarios demonstrate its practicability. First, as an introductory example, we shed light on a ring oscillator in Sect. 3 followed by analogue arithmetics in Sect. 4. The third study is dedicated to emulation of combinable logic gates together with a bit duplicator by water tank systems in binary mode in Sect. 5. A discussion concludes the benefits and challenges raising open questions for future work.

2 Water tank systems

Let A and B be arbitrary sets, $\emptyset$ the empty set, $\mathbb{N}$ the set of natural numbers including zero, $\mathbb{Q}$ the set of rational numbers, and $\mathbb{Q}_+$ the set of non-negative rational numbers. The Cartesian product $A \times B = \{(a,b) \mid a \in A \wedge b \in B\}$ collects all tuples from A and B. $\wp(A)$ symbolises the power set of A. A *fractional multiset* over A is a mapping $\mathcal{F} : A \longrightarrow \mathbb{Q}_+ \cup \{+\infty\}$. Multisets in general can be written as an elementwise enumeration of the form $\{(a_1, \mathcal{F}(a_1)), (a_2, \mathcal{F}(a_2)), \ldots\}$ since $\forall (a,b_1), (a,b_2) \in \mathcal{F} : b_1 = b_2$.

A *water tank system* $\Pi_\sim$ is a construct

$$\Pi_\sim = (\mathcal{W}, A, P, t_{\text{start}}, t_{\text{end}})$$

with its components

$\mathcal{W} : \Sigma \longrightarrow \mathbb{Q}_+ \cup \{+\infty\}$ . . . . . finite fractional multiset of water tanks

Let Σ be an alphabet of water tank identifier symbols. $w \in \Sigma$ specifies a water tank whose initial volume of water is given by $\mathcal{W}(w)$. A water tank is allowed to be initially empty. A water tank $r \in \Sigma$ acting as reservoir is characterised by $\mathcal{W}(r) = +\infty$. The volume of water contained in a water tank w at time $t \in \mathbb{N}$ is captured by $V_w(t) \in \mathbb{Q}_+ \cup \{+\infty\}$. It holds $V_w(0) = \mathcal{W}(w)$.

$A \subset \wp(\mathbb{N} \times \{0,1\})$ . . . . . finite set of valves

Let $w \in \Sigma$ be a water tank identifier, $\Theta \in \mathbb{Q}_+$ a threshold value, and $t \in \mathbb{N}$ a point in time. A valve (also called actor) is a function $\mathsf{a} : \mathbb{N} \longrightarrow \{0,1\}$ having one of the forms

$$\mathsf{a}(t) = \begin{cases} 1 \text{ iff } V_w(t) > \Theta \\ 0 \text{ otherwise} \end{cases}$$

$$\mathsf{a}(t) = \begin{cases} 1 \text{ iff } V_w(t) \geq \Theta \\ 0 \text{ otherwise} \end{cases}$$

$$\mathsf{a}(t) = \begin{cases} 1 \text{ iff } V_w(t) < \Theta \\ 0 \text{ otherwise} \end{cases}$$

$$\mathsf{a}(t) = \begin{cases} 1 \text{ iff } V_w(t) \leq \Theta \\ 0 \text{ otherwise} \end{cases}$$

$$\mathsf{a}_{\text{start}}(t) = \begin{cases} 1 \text{ iff } t \geq t_{\text{start}} \\ 0 \text{ otherwise,} \end{cases}$$

meaning that 1 marks the valve to be fully open and 0 fully closed, respectively.

$P \subset \Sigma \times \Sigma \times A \times \mathbb{Q}_+ \times \mathbb{Q}_+$ . . . . . finite set of pipes

Let $s \in \Sigma$ and $w \in \Sigma$ with $s \neq w$ be water tanks, $B \subseteq A$ a set of valves, $\mathsf{b}_{\text{start}} \in B$ the start valve, $\dot{v} \in \mathbb{Q}_+$ with $\dot{v} > 0$ a volumetric flow rate, and $\Theta_s \in \mathbb{Q}_+$ the minimum volume of water required in s to supply. A pipe $p = (s, w, B, \dot{v}, \Theta_s) \in P$ enables the directed transport of the volume portion $\dot{v}$ of water from s to w within one time step if and only if $(\forall \mathsf{b} \in B : \mathsf{b}(t) = 1) \wedge (\mathsf{b}_{\text{start}}(t) = 1) \wedge (V_s(t) \geq \dot{v}) \wedge (V_s(t) \geq \Theta_s)$. The formal

conditions mean that all valves placed at the pipe have to be open, the supply tank must contain at least the required portion $\dot{v}$ of water, and—in case the entry of the pipe is placed at a higher filling level than 0—the supply tank must even contain at least Θ_s volume units of water.

$t_{\text{start}} \in \mathbb{N}$ . . . . . point in time to start

$t_{\text{end}} \in \mathbb{N}$ with $t_{\text{end}} > t_{\text{start}}$ . . . . . point in time to terminate

2.1 Systems behaviour

The spatiotemporal behaviour of a water tank system aims at *tracing* of the water volumes in each water tank successively over time. To this end, we assume a global clock $t \in \mathbb{N}$ starting with $t = 0$. Each discrete processing step increases the clock by 1 until the point in time to terminate t_{end} is reached. A configuration of a water tank system is defined by the water volume (non-negative rational value) in each water tank, formally expressed by

$$V_w(t) \;\forall\; w \in \Sigma.$$

Initially, we set $V_w(0) = \mathcal{W}(w)$ for all $w \in \Sigma$. A processing step results from the following update scheme in which the water tank volumes $V_w(t+1)$ will be obtained from $V_w(t)$ taking into consideration the measurements of filling levels and corresponding states of the valves available within the system. The update scheme reads as follows:

t:= 0
while $t < t_{\text{end}}$
  for all $(s, w, B, \dot{v}, \Theta_s) \in P$
    if $\left(\prod_{\mathsf{b} \in B} \mathsf{b}(t) = 1\right) \wedge (\mathsf{b}_{\mathsf{start}}(t) = 1) \;\wedge\; (V_s(t) \geq \dot{v}) \wedge (V_s(t) \geq \Theta_s)$
      $V_s(t+1)$:= $V_s(t) - \dot{v}$
      $V_w(t+1)$:= $V_w(t) + \dot{v}$
  t:= $t+1$

The water volume in the reservoir persists at $+\infty$ for the entire time span in which the system is in operation. For passage of water throughout a pipe, we assume no additional consumption of time beyond one processing step having in mind that pipes typically bridge short distances and an extra delay is seen to be neglectable in comparison to the time needed to fill or to empty a water tank.

3 Ring oscillator

For generation of self-sustained clock signals able to synchronise computing processes and for iteratively making available an amount of water in order to control a primitive recursion, an oscillatory course of the water volume within a tank is required. Hence, we enrich the pool of water tank systems by a module acting as ring oscillator. To do so, a cyclic scheme consisting of at least three water tanks suffices; see the left part of Fig. 2. An amount of water initially placed in one of these tanks rotates from one tank to the next one over time. The inflow of a tank inside the cycle might get opened as soon as the tank ahead from the tank before has been emptied. For instance, in the cycle $w_1 \to w_2 \to w_3$, water tank w_2 can be filled after w_3 has been emptied since w_1 is completely filled at this point in time. By placing the corresponding valves, we achieve an autonomous principle of operation. Complementing the cyclic scheme, we add an output water tank y whose temporal course of the water volume becomes managed by w_1. In the same way, further output tanks beyond y can be integrated if necessary.

Our water tank system that mimics a ring oscillator can be formalised by Π_{RO}. We assign a water volume of 10 units to indicate a full tank. All filled pipes share a common volumetric flow rate $\dot{v} = 0.1$. The system is started at $t = 0$:

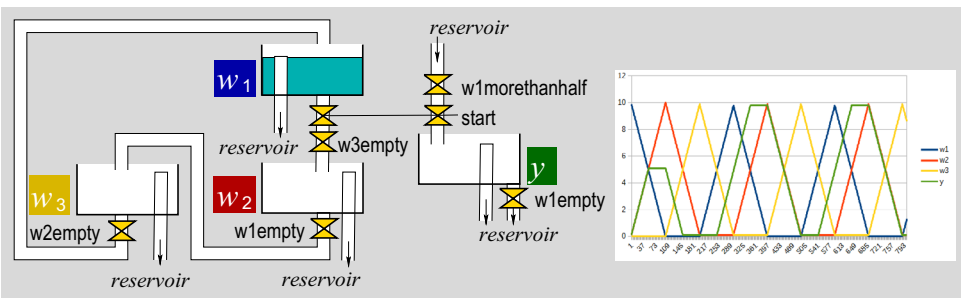


Fig. 2 Water-based ring oscillator composed of a cyclic scheme of three tanks w_1, w_2, and w_3. The water in tank w_1 manages the output course provided in tank y in terms of a clock signal

$$
\begin{aligned}
\Pi_{\mathrm{RO}} =& (\mathcal{W}, A, P, 0, 800) \text{ with} \\
\mathcal{W} =& \{(w_1, 10), (w_2, 0), (w_3, 0), (y, 0), (\mathit{reservoir}, +\infty)\} \\
A =& \left\{ \left(\mathsf{start}(t) = \begin{cases} 1 \text{ iff } t \geq 0 \\ 0 \text{ otherwise} \end{cases} \right), \left(\mathsf{w1empty}(t) = \begin{cases} 1 \text{ iff } V_{w_1}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases} \right), \right. \\
& \left(\mathsf{w2empty}(t) = \begin{cases} 1 \text{ iff } V_{w_2}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases} \right), \\
& \left(\mathsf{w3empty}(t) = \begin{cases} 1 \text{ iff } V_{w_3}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases} \right), \\
& \left. \left(\mathsf{w1morethanhalf}(t) = \begin{cases} 1 \text{ iff } V_{w_1}(t) > 5 \\ 0 \text{ otherwise} \end{cases} \right) \right\} \\
P =& \{(w_1, w_2, \{\mathsf{start}, \mathsf{w3empty}\}, 0.1, 0), (w_1, \mathit{reservoir}, \emptyset, 0.1, 10), \\
& (w_2, w_3, \{\mathsf{w1empty}\}, 0.1, 0), (w_2, \mathit{reservoir}, \emptyset, 0.1, 10), \\
& (w_3, w_1, \{\mathsf{w2empty}\}, 0.1, 0), (w_3, \mathit{reservoir}, \emptyset, 0.1, 10), \\
& (\mathit{reservoir}, y, \{\mathsf{start}, \mathsf{w1morethanhalf}\}, 0.1, 0), \\
& (y, \mathit{reservoir}, \{\mathsf{w1empty}\}, 0.1, 0), (y, \mathit{reservoir}, \emptyset, 0.1, 10)\}
\end{aligned}
$$

The simulation results considering 800 time steps are depicted in the right part of Fig. 2. The temporal course of the water volume in tank y forms the output. After a short transient phase, a stable and self-sustained oscillation becomes evident. Its high and low signal levels are balanced to each other supporting its function of a clock.

4 Emulation of analogue arithmetics

Studying concepts of analogue computing embodies an inspirational field in computer science, and it promises to gain enduring insights for education. Commonly, analogue computers feature by a simple setup and by an illustrative principle of operation close to the adopted mechanisms. In comparison to digital information processing, arithmetics can be directly implemented in an easy way without taking care of distinction between single digits. Analogue computing based on water utilises the volume of water within a tank directly to represent a non-negative rational value able to be measured and modified. In consequence, analogue computers have to cope with slight imprecisions in processing and they tend to be prone to perturbations. Water tank systems operating in analogue mode make use of a uniform volumetric flow rate in all pipes flooded by water. This assumption turns out to be helpful in order to keep the systems small and effective. We focus on analogue water tank systems for addition, non-negative subtraction, integer division, and integer multiplication in a way that the output of an operation can act as an input for a downstream operation which enables evaluation of composed terms.

4.1 Addition

Addition of non-negative rational numbers belongs to the simplest tasks when carried out by a water-based analogue machine. We expect both of the input summands to be available as volumes of water in tanks x and y. For summation, a union of these water volumes into a common output tank $z = x + y$ suffices. Figure 3 illustrates in its left part the underlying scheme.

A formal specification of the corresponding water tank system is given below. We exemplify an instance in which

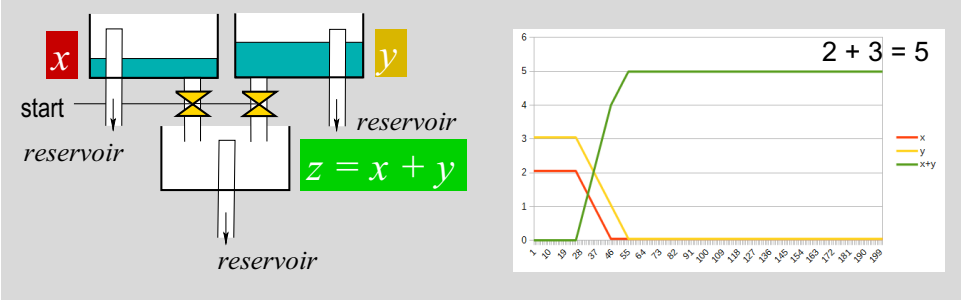


Fig. 3 Analogue implementation of water-based addition with input tanks x, y, and output tank $z = x + y$ whose resulting water volume is available for downstream usage as soon as both input tanks are emptied

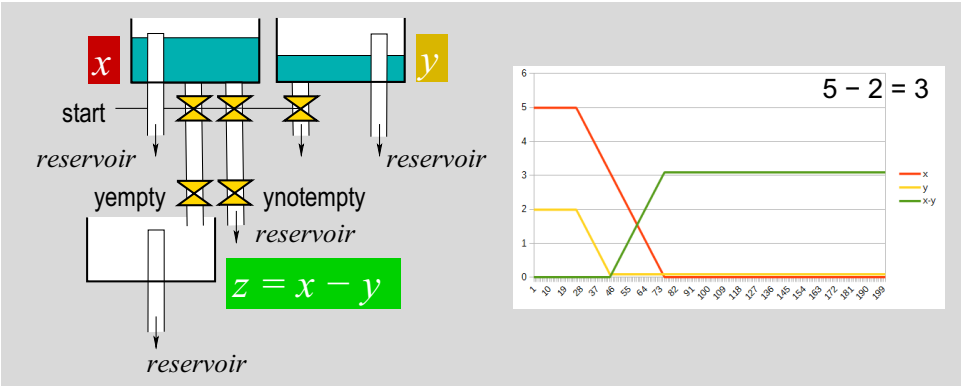


Fig. 4 Water tank scheme for non-negative subtraction with input tanks x, y and output tank $z = x \dot{-} y$. After both input tanks have been completely emptied, the output water volume $V_z(t)$ is ready for further processing

the sum $2 + 3 = 5$ is calculated. This implies initial water volumes $V_x(0) = 2$ and $V_y(0) = 3$. The system is set into operation at $t = 25$ by opening the start valves and it is configured to terminate at $t = 200$. Furthermore, we uniformly assign a volumetric flow rate $\dot{v} = 0.1$ to all filled pipes. The overflow pipes in all tanks have been placed in a way to enable the handling of numbers within a range between 0 and 90.

$$
\begin{aligned}
\Pi_{\text{Add}} &= (\mathcal{W}, A, P, 25, 200) \text{ with} \\
\mathcal{W} &= \{(x, 2), (y, 3), (z, 0), (\mathit{reservoir}, +\infty)\} \\
A &= \left\{ \left(\mathsf{start}(t) = \begin{cases} 1 \text{ iff } t \geq 25 \\ 0 \text{ otherwise} \end{cases} \right) \right\} \\
P &= \{(x, z, \{\mathsf{start}\}, 0.1, 0), (x, \mathit{reservoir}, \emptyset, 0.1, 90), (y, z, \{\mathsf{start}\}, 0.1, 0), \\
&\quad (y, \mathit{reservoir}, \emptyset, 0.1, 90), (z, \mathit{reservoir}, \emptyset, 0.1, 90)\}
\end{aligned}
$$

The simulation results shown in Fig. 3 (right part) disclose a duration of 30 time steps to conduct the calculation. After both of the input water tanks x and y have been completely emptied, the output water volume might be transferred for downstream usage.

4.2 Non-negative subtraction

For emulation of a non-negative subtraction $z = x \dot{-} y = \begin{cases} x - y & \text{iff } x > y \\ 0 & \text{otherwise} \end{cases}$, we make use of a strategy based on the assurance that all pipes flooded by water exhibit the same volumetric flow rate $\dot{v}$. Having this feature at hand, we construct the system beginning with two separate input tanks x and y whose start valves can be opened at the same time. Along with this, the water simultaneously runs from x and from y to the reservoir until y is (nearly) empty. As soon as all water has vanished from y, the outflow of x immediately switches over to fill z from now on. In this way, the desired difference expressed by the portion of the initial water volume in x exceeding those in y has accumulated in z; see left part of Fig. 4.

The visualisation is done by a system instance to compute $5 \dot{-} 2 = 3$ in which the start takes place at $t = 25$. All filled pipes possess a uniform volumetric flow rate $\dot{v} = 0.1$. Each overflow pipe gets supplied in case that a tank contains at least 90 volume units of water. In addition to the start valves, two auxiliary valves yempty and ynotempty are required indicating the empty and non-empty state of tank y.

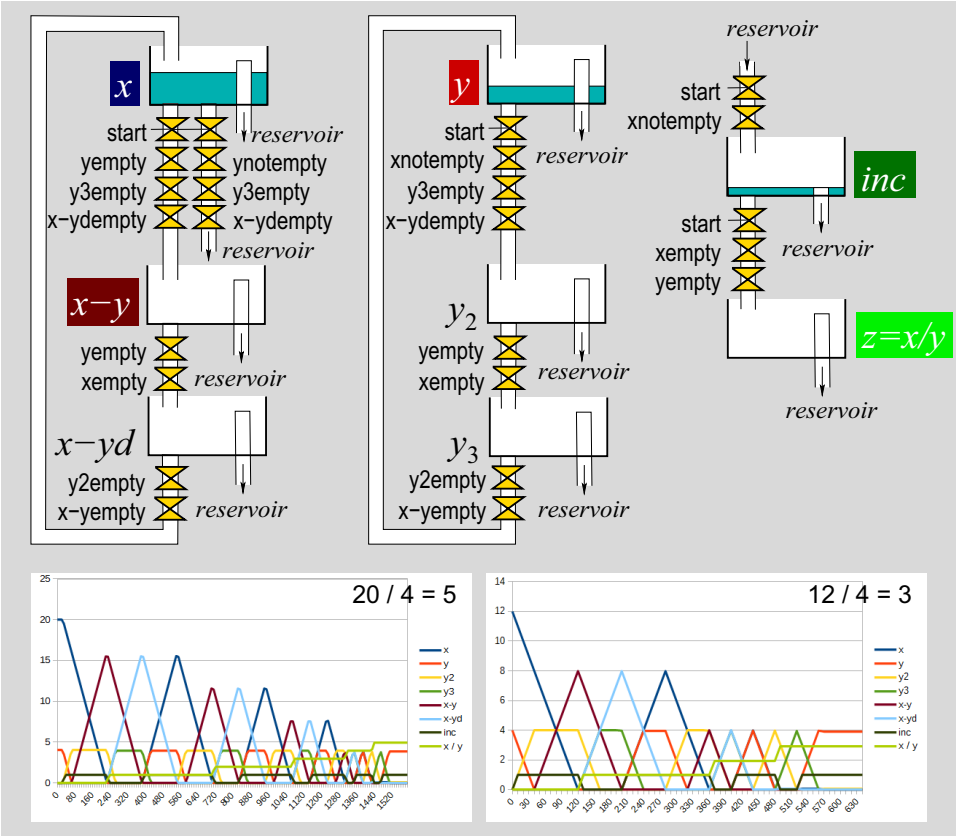


Fig. 5 Water tank scheme for integer division with input tanks x, y and output tank $z = x/y$. After input tank x and auxiliary tanks $x - y$, $x - yd$, and *inc* have been completely emptied, the output water volume $V_{x/y}(t)$ is ready for further processing

$$\Pi_{\text{Sub}} = (\mathcal{W}, A, P, 25, 200) \text{ with}$$

$$\mathcal{W} = \{(x, 5), (y, 2), (z, 0), (reservoir, +\infty)\}$$

$$A = \left\{ \left(\mathsf{start}(t) = \begin{cases} 1 \text{ iff } t \geq 25 \\ 0 \text{ otherwise} \end{cases} \right), \left(\mathsf{yempty}(t) = \begin{cases} 1 \text{ iff } V_y(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases} \right), \right.$$
$$\left. \left(\mathsf{ynotempty}(t) = \begin{cases} 1 \text{ iff } V_y(t) > 0 \\ 0 \text{ otherwise} \end{cases} \right) \right\}$$

$$P = \{(x, z, \{\mathsf{start}, \mathsf{yempty}\}, 0.1, 0), (x, \text{reservoir}, \{\mathsf{start}, \mathsf{ynotempty}\}, 0.1, 0),$$
$$(x, \text{reservoir}, \emptyset, 0.1, 90), (y, \text{reservoir}, \{\mathsf{start}\}, 0.1, 0),$$
$$(y, \text{reservoir}, \emptyset, 0.1, 90), (z, \text{reservoir}, \emptyset, 0.1, 90)\}$$

We let the simulation run for 200 time steps. The right part of Fig. 4 captures the temporal courses of water volumes in the tanks x, y, and z. 50 time steps after starting the system, the calculation result is available in the output tank z and can be used for further processing if needed.

4.3 Integer division

Implementation of integer division by means of a water tank system requires a well-elaborated algorithmic strategy. We decided to employ the recursive definition

$$\left\lceil \frac{x}{y} \right\rceil = \begin{cases} 0 & \text{iff } x < y \ \wedge \ x = 0 \\ 1 & \text{iff } x < y \ \wedge \ x > 0 \\ \left\lceil \frac{x \dot{-} y}{y} \right\rceil + 1 & \text{otherwise,} \end{cases}$$

in which $x, y \in \mathbb{N}$ and $y \geq 1$ holds. For emulation of the recursive scheme, we adapt the ring oscillator and combine it with a subtractor. To this end, we place two separate input tanks x and y complemented by a tank $x - y$ whose contents corresponds to the non-negative subtraction $x \dot{-} y$ as described before. Via the intermediate tank $x - yd$, we transport the water volume representing $x - y$ back to tank x completing a cycle. Each iteration of water throughout this cycle diminishes its rotating volume by y until no water remains and all three tanks x, $x - y$, and $x - yd$ forming the cycle have been emptied. In each iteration, the portion y of water is needed again in order to execute the subtraction. So, we have to take care that y gets restored along with the iteration. For this purpose, we implement a second cycle consisting of three tanks y, y_2, and y_3. The iterations in both cycles run simultaneously in a self-synchronised manner controlled by cross-placed valves; see upper part of Fig. 5. The underlying division is done by a separate auxiliary tank called *inc* with a storing capacity of exactly one volume unit of water. Each iteration opens the supply of tank *inc* for a short moment to refill. Finally, this amount of one unit of water moves to the output water tank $z = x/y$ which in turn accumulates these portions over the time spent for iterations. After that, the output tank contains the desired division result.

The formal specification of a water tank system for integer division comprises eight tanks in total. Moreover, nine types of valves are required and 17 pipes manage the transport of water. The system instance given below is dedicated to the division example $20/4 = 5$. We assign a common volumetric flow rate $\dot{v} = 0.1$ to all filled pipes. All water tanks except *inc* have a capacity of 90 volume units. We start the system at $t = 25$ and observe it for 1600 time steps:

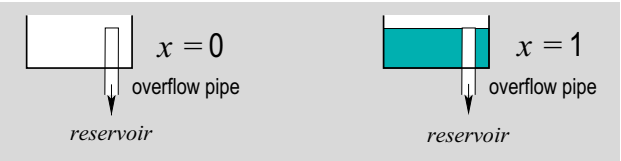


Fig. 7 Representation of logical values by a water tank

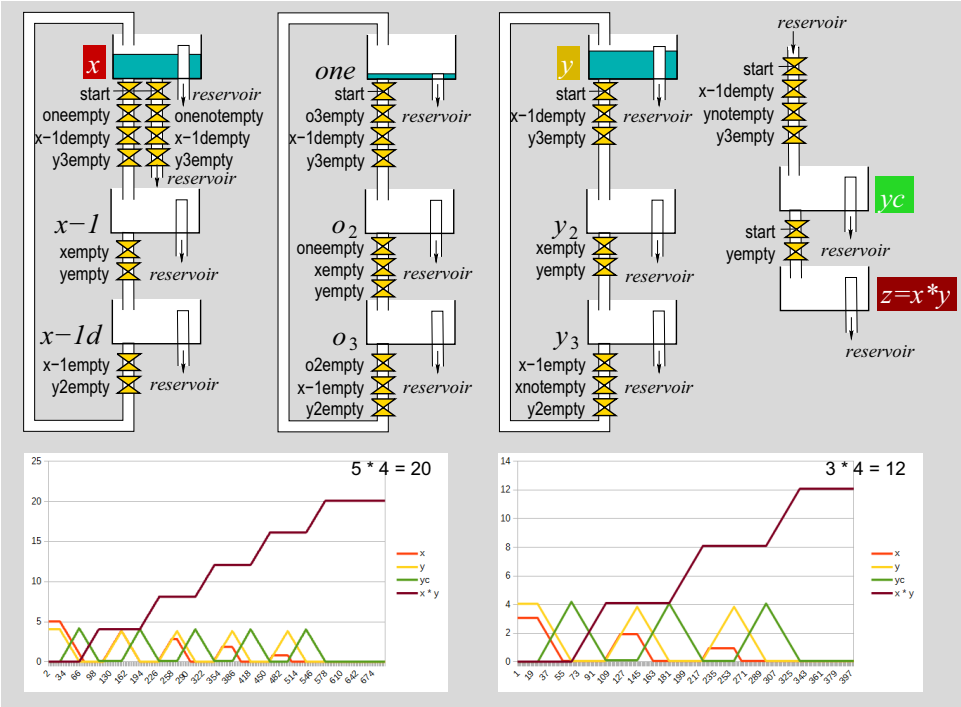


Fig. 6 Water tank scheme for integer multiplication with input tanks x, y and output tank $z = x \times y$. After input tank x and auxiliary tanks $x - 1$, $x - 1d$, and yc have been completely emptied, the output water volume $V_{x \times y}(t)$ is ready for further processing

$$
\begin{aligned}
\Pi_{\mathrm{Div}} &= (\mathcal{W}, A, P, 25, 1600) \text{ with}\\
\mathcal{W} &= \{(x,20),(y,4),(x-y,0),(x-yd,0),(y_2,0),(y_3,0),(inc,0),(x/y,0),\\
&\quad (\text{reservoir},+\infty)\}\\
A &= \left\{ \left(\mathsf{start}(t) = \begin{cases} 1 \text{ iff } t \geq 25 \\ 0 \text{ otherwise} \end{cases}\right), \left(\mathsf{xempty}(t) = \begin{cases} 1 \text{ iff } V_x(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),\right.\\
&\quad \left(\mathsf{xnotempty}(t) = \begin{cases} 1 \text{ iff } V_x(t) > 0 \\ 0 \text{ otherwise} \end{cases}\right),\\
&\quad \left(\mathsf{yempty}(t) = \begin{cases} 1 \text{ iff } V_y(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),\\
&\quad \left(\mathsf{ynotempty}(t) = \begin{cases} 1 \text{ iff } V_y(t) > 0 \\ 0 \text{ otherwise} \end{cases}\right),\\
&\quad \left(\mathsf{x-yempty}(t) = \begin{cases} 1 \text{ iff } V_{x-y}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),\\
&\quad \left(\mathsf{x-ydempty}(t) = \begin{cases} 1 \text{ iff } V_{x-yd}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),\\
&\quad \left(\mathsf{y2empty}(t) = \begin{cases} 1 \text{ iff } V_{y_2}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),\\
&\quad \left.\left(\mathsf{y3empty}(t) = \begin{cases} 1 \text{ iff } V_{y_3}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right)\right\}\\
P &= \{(x, x-y, \{\mathsf{start}, \mathsf{yempty}, \mathsf{y3empty}, \mathsf{x-yempty}\}, 0.1, 0),\\
&\quad (x, \text{reservoir}, \{\mathsf{start}, \mathsf{ynotempty}, \mathsf{y3empty}, \mathsf{x-yempty}\}, 0.1, 0),\\
&\quad (x, \textit{reservoir}, \emptyset, 0.1, 90), (x-y, \text{reservoir}, \emptyset, 0.1, 90),\\
&\quad (x-y, x-yd, \{\mathsf{yempty}, \mathsf{xempty}\}, 0.1, 0), (x-yd, \text{reservoir}, \emptyset, 0.1, 90),\\
&\quad (x-yd, x, \{\mathsf{y2empty}, \mathsf{x-yempty}\}, 0.1, 0), (y, \text{reservoir}, \emptyset, 0.1, 90),\\
&\quad (y, y_2, \{\mathsf{start}, \mathsf{y3empty}, \mathsf{x-ydempty}, \mathsf{xnotempty}\}, 0.1, 0),\\
&\quad (y_2, y_3, \{\mathsf{yempty}, \mathsf{xempty}\}, 0.1, 0), (y_3, \text{reservoir}, \emptyset, 0.1, 90),\\
&\quad (y_3, y, \{\mathsf{y2empty}, \mathsf{x-yempty}\}, 0.1, 0), (inc, \text{reservoir}, \emptyset, 0.1, 1),\\
&\quad (\text{reservoir}, inc, \{\mathsf{start}, \mathsf{xnotempty}\}, 0.1, 0), (x/y, \text{reservoir}, \emptyset, 0.1, 90),\\
&\quad (y_2, \text{reservoir}, \emptyset, 0.1, 90), (inc, x/y, \{\mathsf{start}, \mathsf{xempty}, \mathsf{yempty}\}, 0.1, 90)\}
\end{aligned}
$$

Figure 5 shows in its lower part two simulation runs reflecting the divisions $20/4 = 5$ and $12/4 = 3$. The successive reduction of the water volume available in tank x along with each iteration and the accumulation of the water portions of 1 volume unit becomes apparent from the diagrams. After the tanks inc and x together with its successors $x - y$ and $x - yd$ are empty at the same time, the output has been finalised and can be utilised for further processing.

4.4 Integer multiplication

Within water-based analogue arithmetics, integer multiplication necessitates most effort for construction of the underlying system. Our algorithmic strategy focuses on a recursive definition

$$
x \cdot y = \begin{cases} y & \text{iff } x = 1 \\ (x-1) \cdot y + y & \text{otherwise,} \end{cases}
$$

whereas $x, y \in \mathbb{N}$ and $x \geq 1$. This approach needs to combine a ring oscillator with a subtractor responsible for the decrement to control the descent of x into $x - 1$ when successively tracing the recursion. In consequence, we construct a cycle consisting of three water tanks called x, $x - 1$, and $x - 1d$ for this purpose; see upper left part of Fig. 6. A second cycle composed of three tanks called *one*, o_2, and o_3 manages the iterated presence of the subtrahend 1 to turn x into $x - 1$ by subtraction. The iterations in both cycles need to be synchronised to each other. This is done by cross-placement of valves. Along with each iteration, a water volume of y units has to be made available and successively accumulated to obtain the final multiplication result. This requirement implies a third cycle formed by three tanks called y, y_2, and y_3. An initial water volume of y units rotates along these tanks simultaneously to the iterations taking place in either aforementioned cycles. We need to place additional valves to synchronise this cycle as well. The water with a volume of y units periodically present in tank y controls the supply of a separate tank named *yc* to provide a copy of the water volume y. As soon as an iteration has finished, this volume gets added to the water volume in the output tank $x \times y$. Along with each iteration, a portion of y units is summed up. After all water vanished from x by a number of cycles, the desired calculation result $x \times y$ persists in the correspondent output tank.

We exemplify the calculation $5 \times 4 = 20$ for formulation of a system instance. It is composed of 11 water tanks, 13 types of valves, and 23 pipes in total. A volumetric flow rate $\dot{v} = 0.1$ has been assigned to all filled pipes. A uniform rate is mandatory to ensure the function of the entire system. Moreover, we place the overflow pipes in a way to handle water volumes up to 90 units except tank *one* whose capacity is restricted to 1 unit. The system starts at $t = 25$ and its temporal behaviour has been observed for 700 time steps. Accordingly, the formal specification reads:

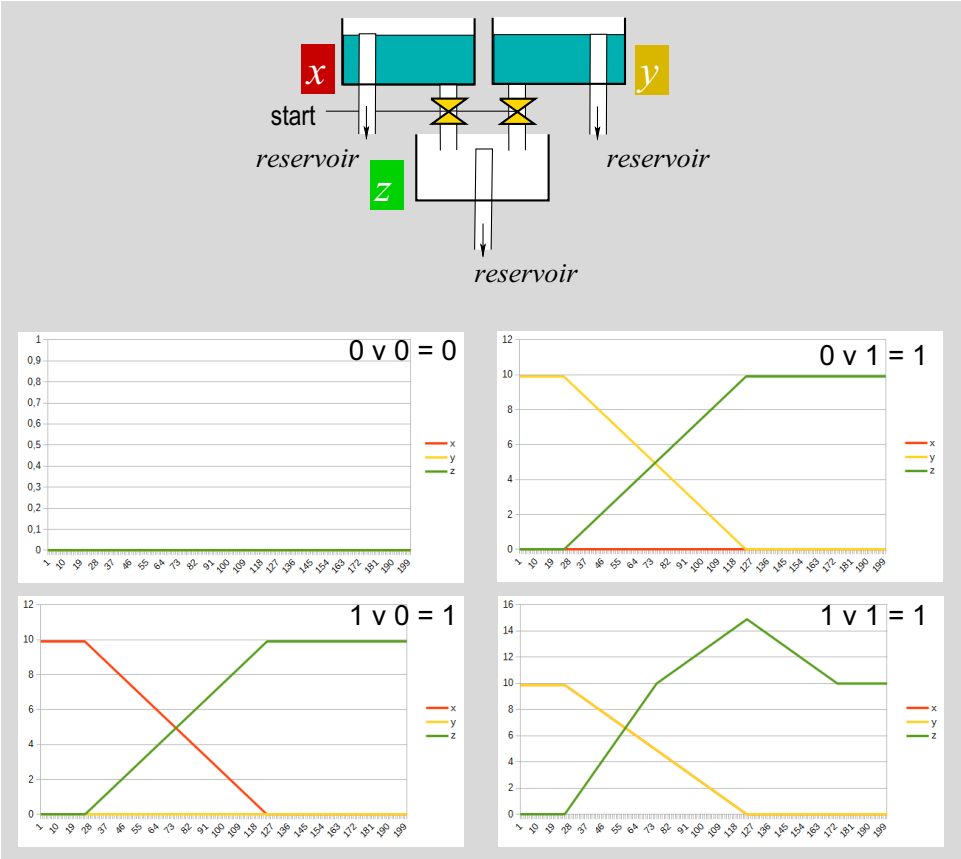


Fig. 8 Schematic representation of a water tank system which mimics an OR gate by $z = x \vee y$. The contents of the output tank z is valid and can be employed for downstream logic gates as soon as both input tanks x, y are empty and tank z is not overfilled any more

$\Pi_{\text{Mul}} = (\mathcal{W}, A, P, 25, 700)$ with

$\mathcal{W} = \{(x,5),(y,4),(x-1,0),(x-1d,0),(one,1),(o_2,0),(o_3,0),(y_2,0),$
$(y_3,0),(yc,0),(x\cdot y,0),(\text{reservoir},+\infty)\}$

$A = \left\{ \left(\mathsf{start}(t) = \begin{cases} 1 \text{ iff } t \geq 25 \\ 0 \text{ otherwise} \end{cases}\right), \left(\mathsf{xempty}(t) = \begin{cases} 1 \text{ iff } V_x(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),\right.$
$\left(\mathsf{xnotempty}(t) = \begin{cases} 1 \text{ iff } V_x(t) > 0 \\ 0 \text{ otherwise} \end{cases}\right),$
$\left(\mathsf{x-1empty}(t) = \begin{cases} 1 \text{ iff } V_{x-1}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),$
$\left(\mathsf{x-1dempty}(t) = \begin{cases} 1 \text{ iff } V_{x-1d}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),$
$\left(\mathsf{oneempty}(t) = \begin{cases} 1 \text{ iff } V_{one}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),$
$\left(\mathsf{onenotempty}(t) = \begin{cases} 1 \text{ iff } V_{one}(t) > 0 \\ 0 \text{ otherwise} \end{cases}\right),$
$\left(\mathsf{o2empty}(t) = \begin{cases} 1 \text{ iff } V_{o_2}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),$
$\left(\mathsf{o3empty}(t) = \begin{cases} 1 \text{ iff } V_{o_3}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),$
$\left(\mathsf{yempty}(t) = \begin{cases} 1 \text{ iff } V_y(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),$
$\left(\mathsf{ynotempty}(t) = \begin{cases} 1 \text{ iff } V_y(t) > 0 \\ 0 \text{ otherwise} \end{cases}\right),$
$\left(\mathsf{y2empty}(t) = \begin{cases} 1 \text{ iff } V_{y_2}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right),$
$\left.\left(\mathsf{y3empty}(t) = \begin{cases} 1 \text{ iff } V_{y_3}(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases}\right)\right\}$

$P = \{(x, x-1, \{\mathsf{start}, \mathsf{oneempty}, \mathsf{x-1empty}, \mathsf{y3empty}\}, 0.1, 0),$
$(x, \text{reservoir}, \{\mathsf{start}, \mathsf{onenotempty}, \mathsf{x-1empty}, \mathsf{y3empty}\}, 0.1, 0),$
$(x, \text{reservoir}, \emptyset, 0.1, 90), (x-1, x-1d, \{\mathsf{xempty}, \mathsf{yempty}\}, 0.1, 0),$
$(x-1, \text{reservoir}, \emptyset, 0.1, 90), (x-1d, x, \{\mathsf{x-1empty}, \mathsf{y2empty}\}, 0.1, 0),$
$(x-1d, \text{reservoir}, \emptyset, 0.1, 90), (one, \text{reservoir}, \emptyset, 0.1, 1),$
$(one, o_2, \{\mathsf{start}, \mathsf{o3empty}, \mathsf{x-1dempty}, \mathsf{y3empty}\}, 0.1, 0),$
$(o_2, o_3, \{\mathsf{oneempty}, \mathsf{xempty}, \mathsf{yempty}\}, 0.1, 0), (o_2, \text{reservoir}, \emptyset, 0.1, 90),$
$(o_3, \text{one}, \{\mathsf{o2empty}, \mathsf{x-1empty}, \mathsf{y2empty}\}, 0.1, 0), (o_3, \text{reservoir}, \emptyset, 0.1, 90),$
$(y, y_2, \{\mathsf{start}, \mathsf{x-1dempty}, \mathsf{y3empty}\}, 0.1, 0), (y, \text{reservoir}, \emptyset, 0.1, 90),$
$(y_2, y_3, \{\mathsf{xempty}, \mathsf{yempty}\}, 0.1, 0), (y_2, \text{reservoir}, \emptyset, 0.1, 90),$
$(y_3, y, \{\mathsf{x-1empty}, \mathsf{xnotempty}, \mathsf{y2empty}\}, 0.1, 0), (y_3, \text{reservoir}, \emptyset, 0.1, 90),$
$(\text{reservoir}, yc, \{\mathsf{start}, \mathsf{x-1dempty}, \mathsf{ynotempty}, \mathsf{y3empty}\}, 0.1, 0),$
$(yc, x\cdot y, \{\mathsf{start}, \mathsf{yempty}\}, 0.1, 0), (yc, \text{reservoir}, \emptyset, 0.1, 90),$
$(x\cdot y, \text{reservoir}, \emptyset, 0.1, 90)\}$

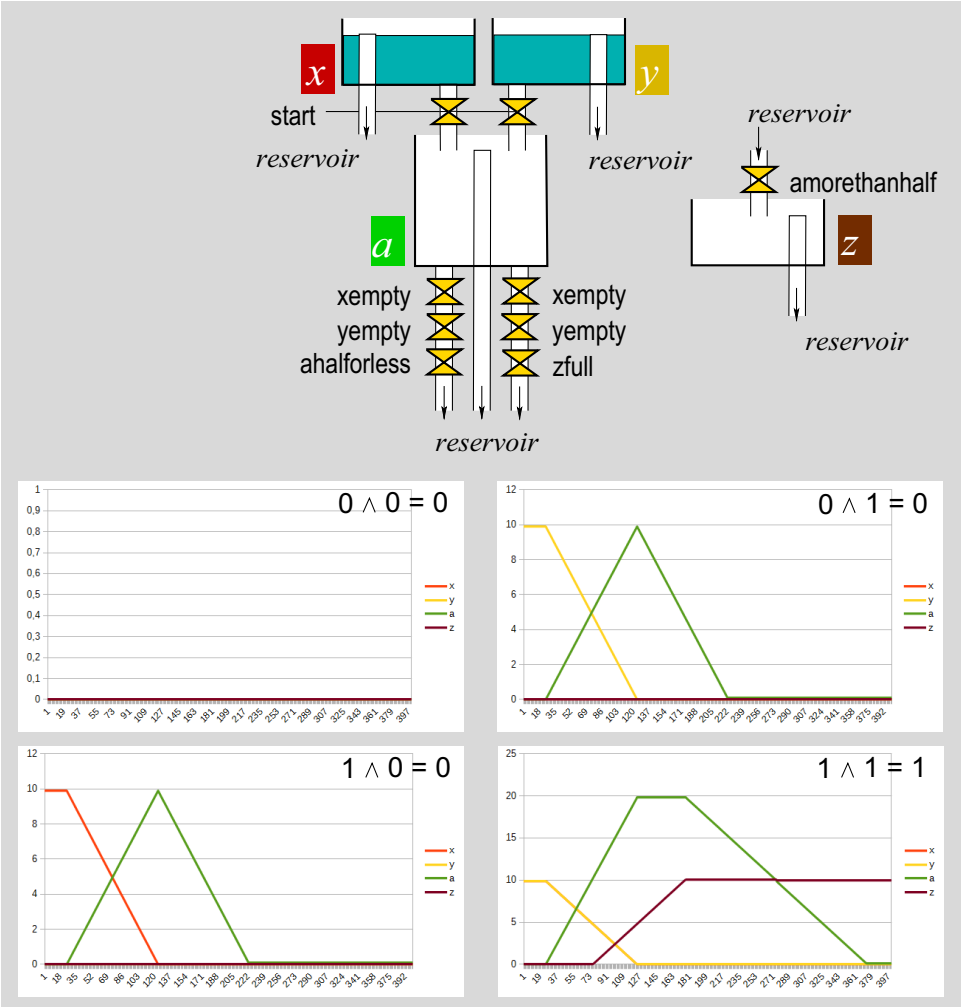


Fig. 9 Schematic representation of a water tank system which mimics an AND gate by $z = x \wedge y$. The contents of the output tank z is valid as soon as all other tanks have been completely emptied

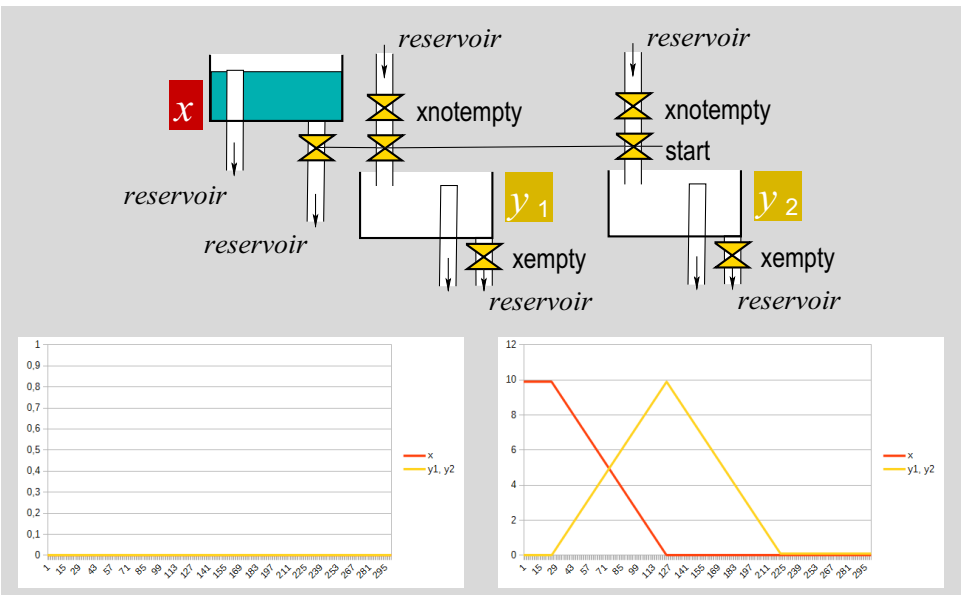


Fig. 10 Scheme of a water tank system acting as bit duplicator $y_i = x$ with $i = 1, 2$. The contents of each output tank y_i can be accessed as soon it is full or empty

The lower part of Fig. 6 shows simulation results for the calculations $5 \times 4 = 20$ and $3 \times 4 = 12$. The courses of the water volumes in the tanks x, y, yc, and $x \times y$ have been depicted over time. The successive increase of the water volume in $x \times y$ by portions of y becomes evident while the water volume in x gets diminished by 1 within each iteration. The final result is available for downstream usage after the tanks x, $x-1$, $x-1d$, and yc have been emptied at the same time.

$$
\begin{aligned}
\Pi_{\mathrm{OR}} &= (\mathcal{W}, A, P, 25, 200) \text{ with} \\
\mathcal{W} &= \{(x, 10), (y, 10), (z, 0), (\text{reservoir}, +\infty)\} \\
A &= \left\{ \left(\mathsf{start}(t) = \begin{cases} 1 \text{ iff } t \geq 25 \\ 0 \text{ otherwise} \end{cases} \right) \right\} \\
P &= \{(x, z, \{\mathsf{start}\}, 0.1, 0), (x, \text{reservoir}, \emptyset, 0.1, 10), (y, z, \{\mathsf{start}\}, 0.1, 0), \\
&\quad (y, \text{reservoir}, \emptyset, 0.1, 10), (z, \text{reservoir}, \emptyset, 0.1, 10)\}
\end{aligned}
$$

5 Emulation of logic gates

For acting in binary mode, a water tank $w \in \Sigma$ constitutes a binary variable whereas an empty or nearly empty tank with $V_w(t) < \dot{v}$ refers to the logical value "0" and a high filling level $V_w(t) \geq 100 \times \dot{v}$ encodes "1", respectively. The time span to fill or to empty a water tank implies a latency in which its logical value is invalid.

In addition to possible outflow pipe(s) applied for information processing, a water tank incorporates an overflow pipe whose entrance position corresponds to the desired high filling level. The overflow pipe leads excessive water from the tank into the reservoir; see Fig. 7.

5.1 OR gate

The disjunction OR is the simplest logic gate to be implemented by a water tank system since a merge of both input water volumes providing their union as output suffices in principle. To do so, we place two input water tanks called x and y whose outflows feed the output water tank $z = x \vee y$, depicted in Fig. 8 (upper part). By opening the start valves, the system is set into operation.

In order to keep the water volume representing the logical value "1" globally uniform within the entire system, we need to take care that the output water volume gets accordingly reduced in case of supplied by either input tanks to calculate $1 \vee 1 = 1$. For this purpose, we place an overflow pipe for transportation of all excessive water to the reservoir. Figure 8 (lower part) shows simulation results for a system instance in which a water volume of 10 units represents "1" and the start is done at $t = 25$. We assign a volumetric flow rate $\dot{v} = 0.1$ to all filled pipes. The formal system's description (initialisation for $1 \vee 1 = 1$) is:

The simulation studies exhibit the most effort for performing the calculation $1 \vee 1 = 1$ since it consumes much time and the result is obtained after $t = 174$. When using the output z as an input for downstream logic gates, it is valid as soon as both input tanks are empty or nearly empty and z is not overfilled. A pipe processing the output needs to be controlled by corresponding valves xempty (open if and only if $V_x(t) < \dot{v}$), yempty ($V_y(t) < \dot{v}$), and znotoverfilled ($V_z(t) \leq 10$), respectively.

5.2 AND gate

The topology of our water-based AND gate (conjunction) resembles its counterpart from disjunction but it incorporates a crucial difference. Since both of the input water tanks x and y must have a high filling level at the logical value "1" to generate the output result "1", their contents is merged and collected in an oversized auxiliary water tank a able to store the double amount of water in comparison to x and y. After a has been filled more than half, the inflow of the output tank z gets opened, supplied by the reservoir. As soon as z has reached its high filling level to encode "1", tank a is emptied again. In case that merely one or none of the input tanks x, y is set to "1", a cannot collect enough water to flood z which implies the desired output of "0" leaving z in empty state. In order to prepare the system for subsequent operation, all water is finally removed from a. Figure 9 (upper part) illustrates the water tank system.

For simulation, we assign 10 volume units of water to represent "1". Furthermore, all filled pipes constitute $\dot{v} = 0.1$. We expect the system to start at $t = 25$. The corresponding formal description initialised to calculate $1 \wedge 1 = 1$ reads:

$\Pi_{\text{AND}} = (\mathcal{W}, A, P, 25, 400)$ with

$\mathcal{W} = \{(x, 10), (y, 10), (a, 0), (z, 0), (\text{reservoir}, +\infty)\}$

$$A = \left\{ \left(\mathsf{start}(t) = \begin{cases} 1 \text{ iff } t \geq 25 \\ 0 \text{ otherwise} \end{cases} \right), \left(\mathsf{xempty}(t) = \begin{cases} 1 \text{ iff } V_x(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases} \right), \right.$$
$$\left(\mathsf{yempty}(t) = \begin{cases} 1 \text{ iff } V_y(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases} \right), \left(\mathsf{zfull}(t) = \begin{cases} 1 \text{ iff } V_z(t) \geq 10 \\ 0 \text{ otherwise} \end{cases} \right),$$
$$\left(\mathsf{ahalforless}(t) = \begin{cases} 1 \text{ iff } V_a(t) \leq 10 \\ 0 \text{ otherwise} \end{cases} \right),$$
$$\left. \left(\mathsf{amorethanhalf}(t) = \begin{cases} 1 \text{ iff } V_a(t) > 10 \\ 0 \text{ otherwise} \end{cases} \right) \right\}$$

$P = \{(x, a, \{\mathsf{start}\}, 0.1, 0), (x, \text{reservoir}, \emptyset, 0.1, 10), (y, a, \{\mathsf{start}\}, 0.1, 0),$
$(y, \text{reservoir}, \emptyset, 0.1, 10), (z, \text{reservoir}, \emptyset, 0.1, 10),$
$(a, \text{reservoir}, \{\mathsf{xempty}, \mathsf{yempty}, \mathsf{ahalforless}\}, 0.1, 0),$
$(a, \text{reservoir}, \{\mathsf{xempty}, \mathsf{yempty}, \mathsf{zfull}\}, 0.1, 0),$
$(a, \text{reservoir}, \emptyset, 0.1, 20), (\text{reservoir}, z, \{\mathsf{amorethanhalf}\}, 0.1, 0)\}$

The lower part of Fig. 9 depicts the temporal courses of the water volumes within tanks x, y, a, and z over time from $t = 0$ to 400 for all logical assignments. It becomes apparent that the calculation of $1 \wedge 1 = 1$ consumes most time revealing the result after $t = 380$. The output is available for downstream usage as soon as the water tanks x, y, and a have been completely emptied.

5.3 Bit duplicator

When water tank systems are applied for calculation of Boolean expressions, it might happen that a Boolean variable occurs several times within the term. Since water is treated as a fugitive medium, a number of copies from a bit-encoding water tank according to the multiplicity of its Boolean variable is needed in order to satisfy all occurrences. To this end, we provide a bit duplicator responsible for making copies from the water volume present in a tank. Figure 10 shows in its upper part a schematic representation of the corresponding water tank system with the input tank x and two output tanks y_1, y_2 tracing the input with a certain delay. The output tanks get supplied by the reservoir whenever x is not empty and the start valve has been opened. In case of an empty water tank x, the output tanks will be emptied as well.

As an example for simulation, let us assume a bit duplicator with input x and two outputs y_1, y_2. A filling level of 10 volume units corresponds to logical "1". All filled pipes possess a uniform volumetric flow rate $\dot{v} = 0.1$. We start the system at $t = 25$ logically initialised with $x = 1$.

$\Pi_{\text{BD}} = (\mathcal{W}, A, P, 25, 300)$ with

$\mathcal{W} = \{(x, 10), (y_1, 0), (y_2, 0), (\text{reservoir}, +\infty)\}$

$$A = \left\{ \left(\mathsf{start}(t) = \begin{cases} 1 \text{ iff } t \geq 25 \\ 0 \text{ otherwise} \end{cases} \right), \left(\mathsf{xempty}(t) = \begin{cases} 1 \text{ iff } V_x(t) < \dot{v} \\ 0 \text{ otherwise} \end{cases} \right), \right.$$
$$\left. \left(\mathsf{xnotempty}(t) = \begin{cases} 1 \text{ iff } V_x(t) > 0 \\ 0 \text{ otherwise} \end{cases} \right) \right\}$$

$P = \{(x, \text{reservoir}, \{\mathsf{start}\}, 0.1, 0), (x, \text{reservoir}, \emptyset, 0.1, 10),$
$(\textit{reservoir}, y_1, \{\mathsf{start}, \mathsf{xnotempty}\}, 0.1, 0), (y_1, \text{reservoir}, \emptyset, 0.1, 10),$
$(y_1, \text{reservoir}, \{\mathsf{xempty}\}, 0.1, 0), (y_2, \text{reservoir}, \{\mathsf{xempty}\}, 0.1, 0),$
$(\text{reservoir}, y_2, \{\mathsf{start}, \mathsf{xnotempty}\}, 0.1, 0), (y_2, \text{reservoir}, \emptyset, 0.1, 10)\}$

We observe the system's behaviour from $t = 0$ to 300 for both cases $x = 0$ and $x = 1$; see lower part of Fig. 10. Either courses of water volumes in y_1 and y_2 follow the input course

with a delay of 100 time steps when deviating from being empty. As soon as each of the output tanks y_1, y_2 is completely empty or completely filled, its contents is available for usage in downstream logic gates.

6 Conclusions

Water tank systems open an inspirational and interesting concept for modelling and visualisation of computational processes. A crucial feature of a water tank system lies in its ability to function without any central or external control. By means of valves that fully open or fully close the hosting pipes according to the filling levels in dedicated water tanks, a self-synchronisation can be obtained supported by water-based modules like a ring oscillator. Cycles composed of at least three tanks in which a volume of water can successively rotate provide a powerful instrument to describe iterations also sufficient for trace of primitive recursion schemes. In our opinion, water tank systems convince due to their simplicity and by their clear operation making all details visible. All Java source codes for the simulation studies presented throughout this paper are available from the first author upon request.

Future work is directed to obtain a direct implementation of a NOT gate and evaluation of composed logic expressions which includes three tasks: (1) termination detection—a control tank that will be filled exactly when the whole evaluation is completed (regardless if the result is empty or not). (2) System reset—to prepare the next evaluation. (3) Simplifying the control valves on output pipes on multistage expression directed acyclic graphs—at any level, only one or two control valves, thus avoiding a combinatorial explosion on the number of valves.

We are aware of the fact that water tank systems in the present form are away from any deployment outside academics, but they could be a prototype in artificial life when equipped with biochemical reactions and substrates residing in the tanks which in turn forms a blood circulation affected by organs. Particularly, description, modelling, and exploration of biological control loops are seen as a promising field of research to be tackled by water tank systems with appropriate extensions.

Acknowledgements Open Access funding provided by Projekt DEAL.

References

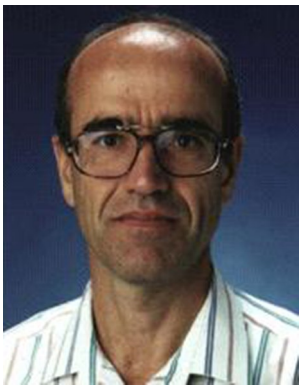
1. Angrist, S. (1964). Fluid control devices. *Scientific American*, *211*(6), 80–88.
2. Belsterling, C. A. (1971). *Fluidic systems design*. New York: Wiley Interscience.
3. Berque, D., Serlin, I., & Vlahov, A. (2004). A brief water excursion: Introducing computer organization students to a water driven 1-bit half-adder. *ACM SIGCSE Bulletin*, *36*(2), 52–56.
4. Bissell, C. (2007). Historical perspectives—the Moniac. A Hydromechanical Analog Computer of the 1950s. *IEEE Control Systems Magazine*, *27*(1), 69–74.
5. Cheow, L. F., Yobas, L., & Kwong, D. L. (2007). Digital microfluidics: Droplet based logic gates. *Applied Physics Letters*, *90*, 054107.
6. Head, T., & Gal, S. (2004). Aqueous computing: Writing into fluid memory. *Current Trends in Theoretical Computer Science*, *1*(1), 493–503.
7. Head, T., & Gal, S. (2006). Aqueous computing: Writing on molecules dissolved in water. In J. Chen, N. Jonoska, & G. Rozenberg (Eds.), *Nanotechnology: Science and computation* (pp. 321–331). Berlin: Springer.
8. Hinze, T., Fassler, R., Lenser, T., & Dittrich, P. (2009). Register machine computations on binary numbers by oscillating and catalytic chemical reactions modelled using mass-action kinetics. *International Journal of Foundations of Computer Science*, *20*(3), 411–426.
9. Hinze, T. (2018). The Java Environment for Nature-inspired Approaches (JENA): A workbench for biocomputing and biomodelling enthusiasts. In C. Graciani, A. Riscos-Nunez, G. Păun, G. Rozenberg, A. Salomaa (Eds.) *Enjoying natural computing, lecture notes in computer science*, vol. 11270, pp. 155-169.
10. Kass, M. A., & Sears, M. L. (1977). Hormonal regulation of intraocular pressure. *Survey of Ophthalmology*, *22*(3), 153–176.
11. Katsikis, G., Cybulski, J. S., & Prakash, M. (2015). Synchronous universal droplet logic and control. *Nature Physics*, *11*, 588–596.
12. Kirshner, J. (1975). *Design theory of fluidic components*. New York: Academic Press.
13. Lukyanov, V. S. (1939). Hydraulic apparatus for engineering computations. *Bulletin of the Russian Academy of Sciences: Physics URSS, Otdeleniye Tekhnicheskikh Nauk*, *2*, 53–67.
14. Mahatantila, K., et al. (2008). Spatial and temporal changes of hydrogeochemistry in ancient tank cascade systems in Sri Lanka: Evidence for a constructed wetland. *Water and Environment Journal*, *22*, 17–24.
15. Mano, M. M., & Kime, C. R. (2004). *Logic and computer design fundamentals*. New Jersey: Pearson Education International.
16. Manca, V. (2019). Metabolic computing. *Journal of Membrane Computing*, *1*(3), 223–232.
17. Mertaniemi, H., Forchheimer, R., Ikkala, O., & Ras, R. H. A. (2012). Rebounding droplet-droplet collisions on superhydrophobic surfaces: From the phenomenon to droplet logic. *Advanced Materials*, *24*(42), 5738–5743.
18. Păun, G. (2003). Membrane Computing. In A. Lingas, B.J. Nilsson (Eds). Fundamentals of Computation Theory. FCT 2003. Lecture Notes in Computer Science, vol. 2751, pp. 284-295
19. Păun, G., Rozenberg, G., & Salomaa, A. (2010). *The oxford handbook of membrane computing*. Oxford: Oxford University Press.

20. Petrovic, A. (2004). Development of the first hydraulic analog computer. *Archives Internationales d'Histoire des Sciences*, *54*(153), 97–110.
21. Ramos, A., et al. (1995). Enzyme basis for pH regulation of citrate and pyruvate metabolism by *Leuconostoc oenos*. *Applied and Environmental Microbiology*, *61*(4), 1303–1310.
22. Rhee, M., & Burns, M. A. (2009). Microfluidic pneumatic logic circuits and digital pneumatic microprocessors for integrated microfluidic systems. *Lab on a Chip*, *9*(21), 3131–3143.
23. Taberlet, N., Marsal, Q., Ferrand, J., & Plihon, N. (2018). Hydraulic logic gates: building a digital water computer. *European Journal of Physics, European Physical Society*, *39*(2), 025801.
24. Trogemann, G., Nitussov, A. Y., & Ernst, W. (2001). *Computing in Russia: The history of computer devices and information technology revealed*. Köln: Vieweg.
25. Wang, Y., & Huang, J. (2014). A water-based molecular flip-flop. *The European Physical Journal Applied Physics*, *68*(3), 30403.

Thomas Hinze studied computer science and received the PhD at Dresden University of Technology (Germany) for his thesis entitled "Universal Models and Selected Algorithms of DNA Computing" in 2002. In 2012, he became a senior university lecturer at the Friedrich Schiller University Jena (Germany) along with his professorial dissertation entitled "Molecular Computing". Currently, he is the head of a research group focussing on modelling in bioinformatics, systems biology, and artificial life which includes applications of membrane computing.

Hendrik Happe is an advanced master student of computer science at Friedrich Schiller University Jena (Germany). His main interest covers aspects of software development and algorithmic design. He is familiar with many programming languages and succeded in a number of projects in which implementations of heuristics form the crucial part.

Alec Henderson is a PhD student at the University of Auckland (New Zealand). His main interests are in distributed computing, artificial intelligence, astrophysics and alternative models of computation.



Radu Nicolescu is a senior lecturer in the School of Computer Science at the University of Auckland (New Zealand). He holds a PhD in mathematics from the University of Bucharest (Romania). Before joining the University of Auckland, he first worked at the University of Bucharest, and then 5 years on industrial optimization and robotics projects (Austria and Germany). His research interests include discrete mathematical models, information coding and complexity, compiler construction, logical and functional programming, models for parallel and distributed computing.