Memcomputing NP-complete problems in polynomial time using polynomial resources and collective states

Fabio L. Traversa,1, 2 Chiara Ramella,2 Fabrizio Bonani,2† and Massimiliano Di Ventra1,‡

1Department of Physics, University of California, San Diego, La Jolla, California 92093, USA
2Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy

(Dated: December 4, 2014)

Memcomputing is a novel non-Turing paradigm of computation that uses interacting memory cells (memprocessors for short) to store and process information on the same physical platform.1 It was recently proved mathematically that memcomputing machines have the same computational power of non-deterministic Turing machines.2 Therefore they can solve NP-complete problems in polynomial time and, using the appropriate architecture, with resources that only grow polynomially with the input size. The reason for this computational power stems from three main properties inspired by the brain and shared by any universal memcomputing machine: intrinsic parallelism, functional polymorphism and information overhead,3 namely the capability of storing more information than the number of memory elements by using the collective state of the memprocessor network. Here, we show an experimental demonstration of an actual memcomputing architecture that solves the NP-complete version of the subset-sum problem in only one step and is composed of a number of memprocessors that scales linearly with the size of the problem. We have fabricated this architecture using standard microelectronic technology so that it can be easily realized in any laboratory setting, whether academic or industrial. Even though the particular machine presented here is eventually limited by noise, it represents the first proof-of-concept of a machine capable of working with the collective state of interacting memory cells, unlike the present-day single-state machines built using the von Neumann architecture.

There are several classes of computational problems that require time and resources that grow exponentially with the input size when solved. This is true when these problems are solved with deterministic Turing machines, namely machines based on the well-known Turing paradigm of computation which is at the heart of any computer we use nowadays.3, 4 Prototypical examples of these difficult problems are those belonging to the class that can be solved in polynomial (P) time if a hypothetical Turing machine—named non-deterministic Turing machine—could be built. They are classified as non-deterministic polynomial (NP) problems, and the machine is hypothetical because, unlike a deterministic Turing machine, it requires a fictitious “oracle” that chooses which path the machine needs to follow to get to an appropriate state.3, 5, 6 As of today, no one knows whether NP problems can be solved in polynomial time by a deterministic Turing machine.7, 8 If that were the case we could finally provide an answer to the most outstanding question in computer science, namely whether NP=P or not.3

Very recently a new paradigm, named memcomputing has been advanced. It is based on the brain-like notion that one can process and store information within the same units (memprocessors) by means of their mutual interactions. This paradigm has its mathematical foundations on an ideal machine, alternative to the Turing one, that was formally introduced by two of us (FT and MD) and dubbed universal memcomputing machine (UMM) 2. Most importantly, it has been proved mathematically that UMMs have the same computational power of a non-deterministic Turing machine,2 but unlike the latter, UMMs are fully deterministic machines and, as such, they can actually be fabricated. A UMM owes its computational power to three main properties: intrinsic parallelism—interacting memory cells simultaneously and collectively change their states when performing computation; functional polymorphism—depending on the applied signals, the same interacting memory cells can calculate different functions; and finally information overhead—a group of interacting memory cells can store a quantity of information which is not simply proportional to the number of memory cells itself.

These properties ultimately derive from a different type of architecture: the topology of memcomputing machines is defined by a network of interacting memory cells (memprocessors), and the dynamics of this network are described by a collective state that can be used to store and process information simultaneously. This collective state is reminiscent of the collective (entangled) state of many qubits in quantum computation, where the entangled state is used to solve efficiently certain types of problems such as factorization.9 Here, we prove experimentally that such collective states can also be implemented in classical systems by fabricating appropriate networks of memprocessors, thus creating either linear or non-linear combinations out of the states of each memprocessor. The result is the first proof-of-concept machine able to solve an NP-complete problem in polynomial time.

The experimental realization of the memcomputing machine presented here, and theoretically proposed in Ref. 2, can solve the NP-complete10 version of the subset-sum problem (SSP) in polynomial time with poly-
nominal resources. This problem is as follows: if we consider a finite set $G \subset \mathbb{Z}$ of cardinality $n$, is there a non-empty subset $K \subseteq G$ whose sum is a given integer number $s$? As we discuss in the following paragraphs, the machine would be scalable to very large numbers of memprocessors only in absence of noise. This problem derives from the fact that in the present realization we use the frequencies of the collective state to encode information and, to maintain the energy of the system bounded, the amplitudes of the frequencies are damped exponentially with the number of memprocessors involved. However, this latter limitation is due to the particular choice of encoding the information in the collective state, and could be overcome by employing other realizations of memcomputing machines. For example in Ref. [2] two of us (FT and MD) proposed a different way to encode a quadratic information overhead in a network of memristors that is not subject to this energy bound.

Another example in which information overhead does not need exponential growth of energy is again quantum computing. For instance, a close analysis of the Shor’s algorithm [11] shows that the collective state of the machine implements all at once (through the superposition of quantum states) an exponential number of states, each one with the same probability that decreases exponentially with the number of qubits involved. Successively, the quantum Fourier transform reorganizes the probabilities encoded in the collective state and “selects” those that actually solve the implemented problem (the prime factorization in the case of the Shor’s algorithm).

Here, it is also worth stressing that our results do not answer the $NP=\overline{P}$ question, since the latter has its solution only within the Turing-machine paradigm: although a UMM is Turing-complete [2], it is not a Turing machine. In fact, (classical) Turing machines employ states of single memory cells and do not use collective states. Other unconventional approaches to the solution of NP-complete problems have been proposed [8, 12, 16], however none of them reduces the computational complexity or requires physical resources not exponentially growing with the size of the problem. On the contrary, our machine can solve an $NP$-complete problem with only polynomial resources. As anticipated, this last claim is valid for an arbitrary large input size only in the absence of noise.

**IMPLEMENTING THE SSP**

The machine we built to solve the SSP is a particular realization of a UMM based on the memcomputing architecture described in Ref. [2], namely it is composed of a control unit, a network of memprocessors (computational memory) and a read-out unit as schematically depicted in Figure 1. The control unit is composed of generators applied to each memprocessor. The memprocessor itself is an electronic module fabricated from standard electronic devices, as depicted in Figure 2 and detailed in the Supplementary Information material. Finally, the read-out unit is composed of a frequency shift module and two multimeters. All the components we have used employ commercial electronic devices.

The control unit feeds the memprocessor network with sinusoidal signals (that represent the input signal of the network) as in Figure 1. It is simple to show that the collective state of the memprocessor network of this machine (that can be read at the final terminals of the network) is given by the real (up terminal) and imaginary (down terminal) part of the function

$$ g(t) = 2^{-n} \prod_{j=1}^{n} (1 + \exp[i\omega_j t]) $$

where $n$ is the number of memprocessors in the network and $i$ the imaginary unit (see Supplementary Information or Ref. [2]). If we indicate with $a_j \in G$ the $j$-th element (integer with sign) of $G$, and we set the frequencies as $\omega_j = 2\pi a_j f_0$ with $f_0$ the fundamental frequency equal for any memprocessor, we are actually encoding the elements of $G$ into the memprocessors through the control-unit feeding frequencies. Therefore, the frequency spectrum of the collective state (1) (or more precisely the spectrum of $g(t) - 2^{-n}$) will have the harmonic amplitude, associated with the normalized frequency $f = \omega / (2\pi f_0)$, proportional to the number of subsets $K \subseteq G$ whose sum $s$ is equal to $f$. In other words, if we read the spectrum of the collective state (1), the harmonic amplitudes are the solution of the subset sum problem for any $s$. From this first analysis we can make the following considerations.

**Information overhead:** the memprocessor network is fed by $n$ frequencies encoding the $n$ elements of $G$, but the collective state (1) encodes all possible sums of subsets of $G$ into its spectrum. It is well known [7] that the number of possible sums $s$ (or equivalently the scaled frequencies $f$ of the spectrum) can be estimated in the worst case as $O(A)$ where $A = \max\{\sum_{a_j > 0} a_j, -\sum_{a_j < 0} a_j\}$. Obviously $A$ (sometimes called the capacity of the problem) has exponential growth [17] on the minimum number $p$ of bits used to represent the elements of $G$ ($p$ is called precision of the problem and we have $A = O(2^p)$, if we take the precision in bits). Thus the spectrum of the collective state (1) encodes an information overhead that grows exponentially with the precision of the problem.

**Computation time:** the collective state (1) is a periodic function of $t$ with minimum period $T = 1/f_0$ because all frequencies involved in (1) are multiples of the fundamental frequency $f_0$. Therefore, $T$ is the minimum time required for computing the solution of the SSP within the memprocessor network and so it can be interpreted as the computation time of the machine. However, this computation time is *independent* of both $n$ and $p$.

**Energy expenditure:** the energy required to compute
the SSP can be estimated as that quantity proportional to the energy of the collective state in one period $E = \int_0^T |g(t)|^2 dt$. By using (1) we have $E \leq \int_0^T dt \leq 1/f_0$, so also the energy needed for the computation is independent of both $n$ and $p$. It is worth remarking here that, in order to keep the energy bounded, all generators have the coefficient 0.5 (see Figure 1) then introducing (see Supplementary Information) the factor $2^{-n}$ in Eq. (1). This means that all frequencies involved in the collective state (1) are dampened by the factor $2^{-n}$. In the case of the ideal machine, i.e., a noiseless machine, this would not represent an issue because no information is lost. On the contrary, when noise is accounted for, the exponential factor represents the hardest limitation of the experimentally fabricated machine, which we reiterate is a technological limit for this particular realization of a memcomputing machine but not for all of them.

**READING THE SSP SOLUTION**

With this analysis we have proven that the UMM represented in Figure 1 can solve the SSP with $n$ memprocessors, a control unit formed by $n+1$ generators and taking a time $T$ and an energy $E$ independent of both $n$ and $p$. Therefore, at first glance, it seems that this machine (without the read-out unit) can solve the SSP using only resources polynomial (specifically, linear) in $n$. However, we need one more step: we have to read the result of the computation. Unfortunately, we cannot simply read the collective state (1) using, e.g., an oscilloscope and performing the Fourier conversion. This is because the most optimized algorithm to do this (see Supplementary Information and Ref. [2]) is exponential in $p$, i.e., it has the same complexity of standard dynamic programming [17].

However, a solution to this problem can be found by just using standard electronics to implement a read-out unit capable of extracting the desired frequency amplitude without adding any computational burden. In Figure 1 we sketch the read-out unit we have used. It is composed of a frequency-shift module and two multimeters. The frequency shift module is in turn composed of two voltage multipliers and two sinusoidal generators as depicted in Figure 2 and it works as follows. If we connect to one multiplier the real part of a complex signal $s$ and to the other the imaginary part, then we get the desired frequency component of $s$.
TABLE I. Measurements from the read-out unit of Figure 1 for a 6-memprocessor network with fundamental frequency \( f_0 = 100 \text{ Hz} \) encoding the set \( G = \{ 130, -130, -146, -166, -44, 118 \} \). In the fifth and sixth columns the voltages are respectively given by \( V_s = 2^p(V_{DC_{up}} + V_{DC_{down}}) \) and \( V_{-s} = 2^p(V_{DC_{up}} - V_{DC_{down}}) \). The last two columns are the analytical results.

| s | \( \omega_s/(2\pi) \) [KHz] | \( V_{DC_{up}} \) [mV] | \( V_{DC_{down}} \) [mV] | \( V_s \) [V] | \( V_{-s} \) [V] | #subs. sum s | #subs. sum \(-s\) |
|---|------------------|------------------|------------------|-------|-------|--------|--------|
| 0 | 0 | 31.7 | 0 | 1.02 | 1.02 | 1 | 1 |
| 74 | 7.4 | 15.3 | 15.0 | 1.94 | 0.02 | 2 | 0 |
| 130 | 13.0 | -0.2 | 14.9 | 0.94 | -0.97 | 1 | 1 |
| 146 | 146.0 | 14.8 | 15.8 | -0.06 | 1.96 | 0 | 2 |
| 248 | 24.8 | 7.6 | 7.2 | 0.95 | 0.02 | 1 | 0 |
| 485 | 48.5 | -0.4 | -0.7 | -0.07 | 0.02 | 0 | 0 |
| 486 | 48.6 | -8.9 | 6.6 | -0.14 | 0.99 | 0 | 1 |

It is worth noticing that, if we wanted to simulate our machine by including the read-out unit the computational complexity would be \( O(2^p) \) (close to the standard dynamic programming that is \( O(n2^p) \) [17]). In fact, from the Nyquist-Shannon sampling theorem [18], the minimum number of samples of the shifted collective state (i.e., the outputs of the frequency-shift module) must be equal to the number of frequencies of the signal (in our case of \( O(2^p) \)) in order to accurately evaluate even one of the harmonic amplitudes [19]. The last claim can be intuitively seen from this consideration: the DC voltage \( V_{DC_{up}} \) must be calculated in the simulation by evaluating the integral \( V_{DC_{up}} = T^{-1} \int_0^T v_{up}(t)\,dt \) and this requires at least \( O(2^p) \) samples for an accurate evaluation [18]. On the other hand, the multimeter of the hardware implementation, being essentially a narrow low-pass filter, performs an analog implementation of the integral over a continuous time interval \( T \) (independent of \( n \) and \( p \)), directly providing the result, thus avoiding the need of sampling the waveform and computing the integral.

In Figure 3 the absolute value of the spectrum of the collective state for networks of 4, 5 and 6 memprocessors is compared with the theoretical results given by the spectrum of \( g(t) \) (see Supplementary Information for more details on the hardware and measurement process). Non-idealities of the circuit and electronic noise in general are the sources of the small discrepancies with respect to the theoretical results. Nevertheless, the machine we fabricated demonstrates that using the collective state of all memprocessors, instead of the uncoupled states of the individual memory units, we can carry out difficult computing tasks (NP-complete problems) with polynomial resources. Finally, in Table I the measurements at the read-out circuit are listed for different harmonics for a 6-memprocessor network. The precision is up the third digit.
CONCLUSIONS

In conclusion we have demonstrated experimentally a deterministic memcomputing machine that is able to solve an NP-complete problem in polynomial time (actually in one step) using only polynomial resources. The actual machine we built clearly suffers from technological limitations due to unavoidable noise that impair the scalability. These limitations derive from the fact that we encode the information directly into frequencies, and so ultimately into energy. This issue can, however, be overcome in other UMMs using other ways to encode such information. Irrespective, this machine represents the first experimental realization of a UMM that uses the collective state of the whole memprocessor network to exploit the information overhead theoretically introduced in Ref. [2]. Finally, it is worth mentioning that the machine we have fabricated is not a general purpose one. However, other realizations of UMMs are general purpose and can be easily built with available technology [20–24]. Their practical realization would thus be a powerful alternative to current Turing-like machines.

ACKNOWLEDGEMENTS

The hardware realization of the machine presented here was supported by Politecnico di Torino through the Neural Engineering and Computation Lab initiative. MD acknowledges partial support from CMRR.

[1] Massimiliano Di Ventra and Yuriy V. Pershin. The parallel approach. *Nature Physics*, 9:200–202, 2013.
[2] F. L. Traversa and M. Di Ventra. Universal Memcomputing Machines. preprint on arXiv:1405.0931, 2014.
[3] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.
[4] John L. Hennessy and David A. Patterson. *Computer Architecture, Fourth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.
[5] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. of the London Math. Soc.*, 42:230–265, 1936.
[6] Alan M. Turing. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life, Plus The Secrets of Enigma*. Oxford University Press, 2004.
[7] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
[8] Kan Wu, Javier García de Abajo, Cesare Soci, Perry Ping Shum, and Nikolay I Zheludev. An optical fiber network oracle for np-complete problems. *Light: Science & Applications*, 3(2):e147, 2014.
[9] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge Series on Information and the Natural Sciences. Cambridge University Press, 10th Anniversary edition, 2010.
SUPPLEMENTARY INFORMATION

CONSIDERATIONS ON THE OPERATING FREQUENCY RANGE

The operating frequency range of the experimental setup needs to be discussed with the measurement target in mind. For the measurement of the entire collective state, the limiting frequency is due to the oscilloscope we use to sample the full signal at the output of the memprocessor network. On the other hand, the measurement of some isolated harmonic amplitudes using the readout unit, transfers this bottleneck to the voltage generators and internal memprocessor components. It is worth stressing that measuring the collective state is not the actual target of our work because it has the same complexity of the standard algorithms for the SSP as discussed above and in Ref. [2]. Here, for completeness, we provide measurements of the collective state only to prove that the setup works properly. The actual frequency range of the setup is discussed in the next section.

SETUP FREQUENCY RANGE

Let us consider \( a_j \in G \), and the integer
\[
A = \max \left\{ -\sum_{a_j < 0} a_j, \sum_{a_j > 0} a_j \right\}.
\]

We also consider \( f_0 \in \mathbb{R} \) and we encode the \( a_j \) in the frequencies by setting the generators at frequencies \( f_j = |a_j| f_0 \), so the maximum frequency of the collective state will be \( f_{\text{max}} = A f_0 \). From these considerations, we can first determine the range of the voltage generators: it must allow for minimum frequency (resolution)
\[
f_{g_{\text{min}}} \leq f_0
\]
and maximum frequency (bandwidth)
\[
f_{g_{\text{max}}} \geq f_0 \max\{|a_j|\}.
\]

We employed the Agilent 33220A Waveform Generator, which has 1 \( \mu \text{Hz} \) resolution and 20 MHz bandwidth. This means that, in principle, we can accurately encode \( G \) when composed of integers with a precision up to 13 digits (which is quite the same precision of the standard double-precision integers) provided that a stable and accurate external clock references, such as a rubidium frequency standard is used. This because the internal reference of such generators, introduces a relative uncertainty on the synthesized frequency in the range of some parts-per-billion (10^{-9}) thus limiting the resolution at high frequency, down to few mHz at the maximum frequency. However, as anticipated, this issue can be solved by employing an external reference providing higher accuracies.
Inverting Differentiator

The quadrature signal with respect to the input sum and one difference. Since we can find a reasonable $f_0$ satisfying the frequency constraints.

MEMPROCESSOR

The memprocessor, synthetically discussed previously and sketched in Figure 2, is shown in the pictures of Figure 4(a) and a more detailed circuit schematics is given in Figure 4(b). Each module has been realized as a single Printed Circuit Board (PCB) and connections are performed through coaxial cables with BNC terminations. According to Figure 2, each memprocessor must perform one derivative $-\omega_j \frac{dv}{dt}$, four multiplications, one sum and one difference. Since $v(t) = 0.5[1+\cos(2\pi f_0 a_j t)]$ and $\omega_j = 2\pi f_0 a_j$, then $-\omega_j^{-1} \dot{v}(t) = 0.5 \sin(2\pi f_0 a_j t)$, i.e. the quadrature signal with respect to the input $v(t)$. This can be easily obtained with the simple OP-AMP-based inverting differentiator depicted in Figure 4(b), designed to have unitary gain at frequency $\omega_j$. Similarly, an inverting summing amplifier and a difference amplifier can be realized as sketched in Figure 4(b) to perform sum and difference of voltage signals, respectively. The OP-AMP selected is the Texas Instruments LM7171 Voltage Feedback Amplifier.

Implementing multiplication is slightly more challenging. OP-AMP based analog multipliers are very sensitive circuits. Therefore, they need to be carefully calibrated. This makes a discrete OP-AMP based realization quite challenging and the integration expensive. We thus adopted a pre-assembled analog multiplier: the Texas Instrument (Burr-Brown) MPY634 Analog Multiplier, which ensures four-quadrant operation, good accuracy and wide enough bandwidth (10MHz). The only drawback of this multiplier is that the maximum precision is achieved with a gain of 0.1. Therefore, being the input signals in general quite small, this further lowering of the precision can make the output signal comparable to the offset voltages of the subsequent OP-AMP stages. For this reason, we have included in the PCB a gain stage (inverting amplifier) before each output to compensate for the previous signal inversion and lowering. These stages permit also manual offset adjustment by means of a tunable network added to the non-inverting input, as shown in the schematic. Finally a low-pass filter with corner frequency $f_e \gg A f_0$ has been added to the outputs to limit noise. Figure 4(a) shows a picture of one of the modules, which have been realized on a 100 mm x 80 mm PCB. The power consumption of each module
is quite high since all of the 10 active components work with ±15 V supply. OP-AMPs have a quiescent current of 6.5 mA, while the multipliers a quiescent current of 4 mA, yielding a total DC current of around 50 mA per module.

Finally, we briefly discuss how connected memprocessors work. From Figure 4-(c) if we have \( v(t) = 0.5[1 + \cos(\omega t)] \), \( v_1 = \text{Re}[f(t)] \) and \( v_2 = \text{Im}[f(t)] \) for an arbitrary complex function \( f(t) \), at the output of the memprocessor we will find

\[
v_{1\text{out}} = 0.5[1 + \cos(\omega t)]\text{Re}[f(t)] - 0.5\sin(\omega t)\text{Im}[f(t)] = \text{Re}[0.5(1 + e^{i\omega t}) f(t)]
\]

\[
v_{2\text{out}} = 0.5[1 + \cos(\omega t)]\text{Im}[f(t)] + 0.5\sin(\omega t)\text{Re}[f(t)] = \text{Im}[0.5(1 + e^{i\omega t}) f(t)]
\]

Since we can only set positive frequencies for the generators, in order to encode negative frequencies (i.e., a negative \( a_j \)) we can simply invert the input and output terminals as depicted in Figure 4-(c). Therefore, if we set \( f(t) = 1 \) for the first memprocessor, i.e., \( v_1 = 1 \) and \( v_2 = 0 \), at the output of the first memprocessor we will find the real and imaginary parts of \( 0.5(1 + \exp[i\omega t]) \) that will be the new \( f(t) \) for the second memprocessor. Proceeding in this way we find the collective state \( |1\rangle \) at the end of the last memprocessor.

COLLECTIVE STATE MEASUREMENT

In order to test if the memprocessor network correctly works, we have carried out the measurement of the full collective state \( g(t) \) at the end of the last memprocessor. This task requires an extra discussion on the operating frequency range. Indeed, the instrument we used to acquire the output waveform is the LeCroy WaveRunner 6030 Oscilloscope. The measurement process consists in acquiring the output waveforms and apply the FFT in software. Being the collective state a purely periodic signal, i.e., a signal containing only frequencies multiples of \( f_0 \) and with known maximum frequency \( f_{\text{max}} = A f_0 \), from the discrete Fourier transform theory and Nyquist-Shannon sampling theorem, we need to sample the interval \([0, 1/f_0]\) into \( N = 2 f_{\text{max}}/f_0 + 1 \) subintervals of width \( \Delta t = (N f_0)^{-1} \) to compute the exact spectrum of \( g(t) \). In other words we need samples \( g(t_j) \) with \( t_j = k \Delta t \) and \( k = 0, ..., N - 1 \) to compute the exact spectrum of \( g(t) \). Therefore, with the oscilloscope we must be able to acquire at least \( N + 1 \) samples of \( g(t) \) into the time interval \([0, 1/f_0]\).

This relationship turns out to be a constraint on the usable frequency range since we must perform the measurement in a reasonable time, and we cannot exceed the maximum sampling frequency of the instrument that we use for acquisition, nor its maximum memory capability. In our experimental proof, the LeCroy WaveRunner 6030 Oscilloscope is characterized by 350 MHz bandwidth, 2.5 GSa/s sampling rate and \( 10^5 \) discretization points when saving waveforms in ascii format (i.e., \( N_{\text{Omax}} = 10^5 \)). The bandwidth of the oscilloscope is very large thus it is not really a constraint, while the limit \( N_{\text{Omax}} \) is, in fact we have the constraint

\[
f_{\text{max}} \leq \frac{f_0}{2}(N_{\text{Omax}} - 1).
\]

With this value, choosing \( f_0 = 1 \) Hz allows to have \( A \leq (10^5 - 1)/2; f_{\text{max}} \lesssim 50 \) KHz and requires 1 s measurement time which is a quite long time in electronics. Therefore, without varying \( A \), we can choose larger \( f_0 \) that allows for a smaller measurement time. We choose \( f_0 = 100 \) Hz which means a measurement time of only few tens of ms, and \( f_{\text{max}} \lesssim 5 \) MHz.

EXPERIMENTAL SET-UP

The laboratory set-up we have employed is sketched in Fig. 6-(a), while Fig. 5 reports a picture of the same. The order of cascade connection of the memprocessors is arbitrary. For this test, we ordered the module such that \( G = \{130, -130, -146, -166, -44, 118\} \) (see Figure 6-(a)) in order to have the two memprocessors related to the two positive numbers \((130\) and \(118\)) at the beginning and at the end of the chain, respectively, thus minimizing the number of the “swapped” connections (see Figure 6-(c)). A two-output power supply (model Agilent E3631A) is used to generate both the 0 and 1 V at the inputs of the first memprocessor and the ±15 V supply for all the modules (parallel connection). The input \( v(t) \) of each module is generated by a Agilent 33220A waveform generator, while the output is observed through both an oscilloscope (model LeCroy Waverunner 6030, see Figure 6-(c)) and a multimeter (model Agilent 34401A). In particular, the oscilloscope is used for the AC waveform, while the multimeter measured the DC component in order to avoid errors due to the oscilloscope probes which are very inaccurate at DC and may show DC offsets up to tens of mV.

Another issue concerns the synchronization of the generators. The six generators we use must share the same time base and they must have the same starting instant (the \( t = 0 \) instant) when all the cosine waveforms must have amplitude 0.5 V. To have a common time base we used the 10 MHz time base signal of one of the generators (master) and we connected the master output signal to all the other (slave) generators at the 10 MHz input. In this way they ignore their own internal time base and lock to the external one. In order to have a common \( t = 0 \) instant we must run the generators in the infinite burst mode. In this mode the generators produce no output signal until a trigger input is given, and then they run indefinitely until they are manually stopped. The
trigger input can be external or manual (soft key): the master device is set up to expect a manual input while the slave devices are controlled by an external trigger coming from the master. Finally, in order to correctly visualize the output waveforms we must “synchronize” also the oscilloscope. The trigger signal of the oscilloscope must have the same frequency of the signal to be plotted, or at least one of its subharmonics. Otherwise, we see the waveform moving on the display and, in case two or more signals are acquired, we lose the information concerning their phase relation. To solve this problem we connect the external trigger input of the oscilloscope to a dedicated signal generator, producing a square waveform at frequency $f_0$, which is the greatest common divisor of all the possible frequency components of the output signals. This dedicated generator is also used as the master for synchronization. Fig. 6(b) shows how the generators must be connected in order to obtain the required synchronization.

FIG. 5. Pictures of the laboratory test bench.

FIG. 6. (a) Sketch of the laboratory test bench. (b) Synchronization of the waveform generators. (c) Collective state of 6 memprocessors network read by the oscilloscope.