# Exploiting Chaos to Design Flexible Hardware

## A new direction in harnessing chaos: Chaos for Computation

Sudeshna Sinha

The Institute of Mathematical Sciences, Chennai

homepage: http://www.imsc.res.in/∼sudeshna

# Chaos as a Computing Medium

- Chaos provides a rich variety of behaviors :

  Can serve as a versatile pattern generator

- Exploit this flexibility for implementing computational tasks

Sinha & Ditto, Physical Review Letters, September 1998;
Physcial Review E, 1999

## Thresholding as a strategy for extracting a wide range of spatiotemporal patterns from a chaotic system in a controlled manner

Enables us to exploit the richness of chaos in a direct and efficient manner

Consider a general dynamical system, and choose a state variable to be monitored

Threshold Mechanism is triggered whenever the value of the variable exceeds a critical threshold $x^\star$

The variable is then re-set to $x^\star$

The dynamics continues till the next occurence of the variable exceeding the threshold

If $x > x^*$ then $x = x^*$

## Different regular dynamical patterns obtained for different thresholds

- Example: chaotic logistic map $f(x) = 4x(1-x)$ under thresholding

## Different regular dynamical patterns obtained for different thresholds

- Example: chaotic logistic map $f(x) = 4x(1-x)$ under thresholding
- $x^* < 0.5$ : Fixed point

## Different regular dynamical patterns obtained for different thresholds

- Example: chaotic logistic map $f(x) = 4x(1-x)$ under thresholding

- $x^* < 0.5$ : Fixed point
- $0.5 < x^* < 0.809$ : Period 2

## Different regular dynamical patterns obtained for different thresholds

- Example: chaotic logistic map $f(x) = 4x(1-x)$ under thresholding

- $x^* < 0.5$ : Fixed point
- $0.5 < x^* < 0.809$ : Period 2
- $0.809 < x^* < 0.85$ : Period 4

## Different regular dynamical patterns obtained for different thresholds

- Example: chaotic logistic map $f(x) = 4x(1-x)$ under thresholding

- $x^* < 0.5$ : Fixed point
- $0.5 < x^* < 0.809$ : Period 2
- $0.809 < x^* < 0.85$ : Period 4
- $x^* = 0.86$ : Period 6

## Different regular dynamical patterns obtained for different thresholds

- Example: chaotic logistic map $f(x) = 4x(1-x)$ under thresholding

- $x^* < 0.5$ : Fixed point
- $0.5 < x^* < 0.809$ : Period 2
- $0.809 < x^* < 0.85$ : Period 4
- $x^* = 0.86$ : Period 6
- $x^* = 0.88$ : Period 7

## Different regular dynamical patterns obtained for different thresholds

- Example: chaotic logistic map $f(x) = 4x(1-x)$ under thresholding

- $x^* < 0.5$ : Fixed point
- $0.5 < x^* < 0.809$ : Period 2
- $0.809 < x^* < 0.85$ : Period 4
- $x^* = 0.86$ : Period 6
- $x^* = 0.88$ : Period 7
- $x^* = 0.9$ : Period 9

## Principle : Restricts available phase space

## Dynamic Range Limiter

- Clips (prunes) the temporal sequences to stable desired patterns

- Enforces a periodicity on the sequences through the thresholding action which acts as a re-setting of initial conditions

- Chaos advantageous as it possesses a rich range of temporal patterns which can be clipped to different behavious

# Be-heading the Chaotic Map

- Study the forward iterates of the map with initial value at threshold: $f(x^*), f^2(x^*), \ldots$
- Ascertain which iterate exceeds the threshold
- If the $k^{\text{th}}$ iterate exceeds the threshold then we obtain period $k$
- Formulate the different solutions using the inverse map: L and R

- Exact correspondence between threshold and controlled period can be obtained analytically through symbolic dynamics

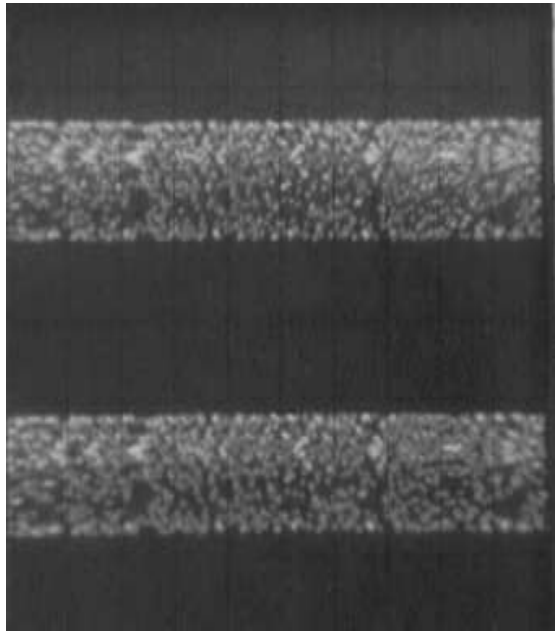
- The system is trapped in a super-stable cycle the instant it exceeds threshold

Sinha, Physical Review E, 1993;

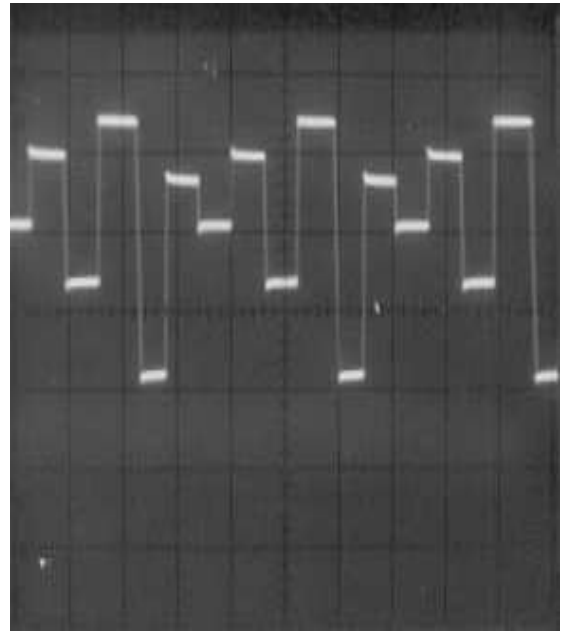
Sinha, Physics Letters A, 1994;

Reviewed in Int. J. of Modern Physics, 1995

## Experimental verification of clipping chaos to periods of wide ranging orders



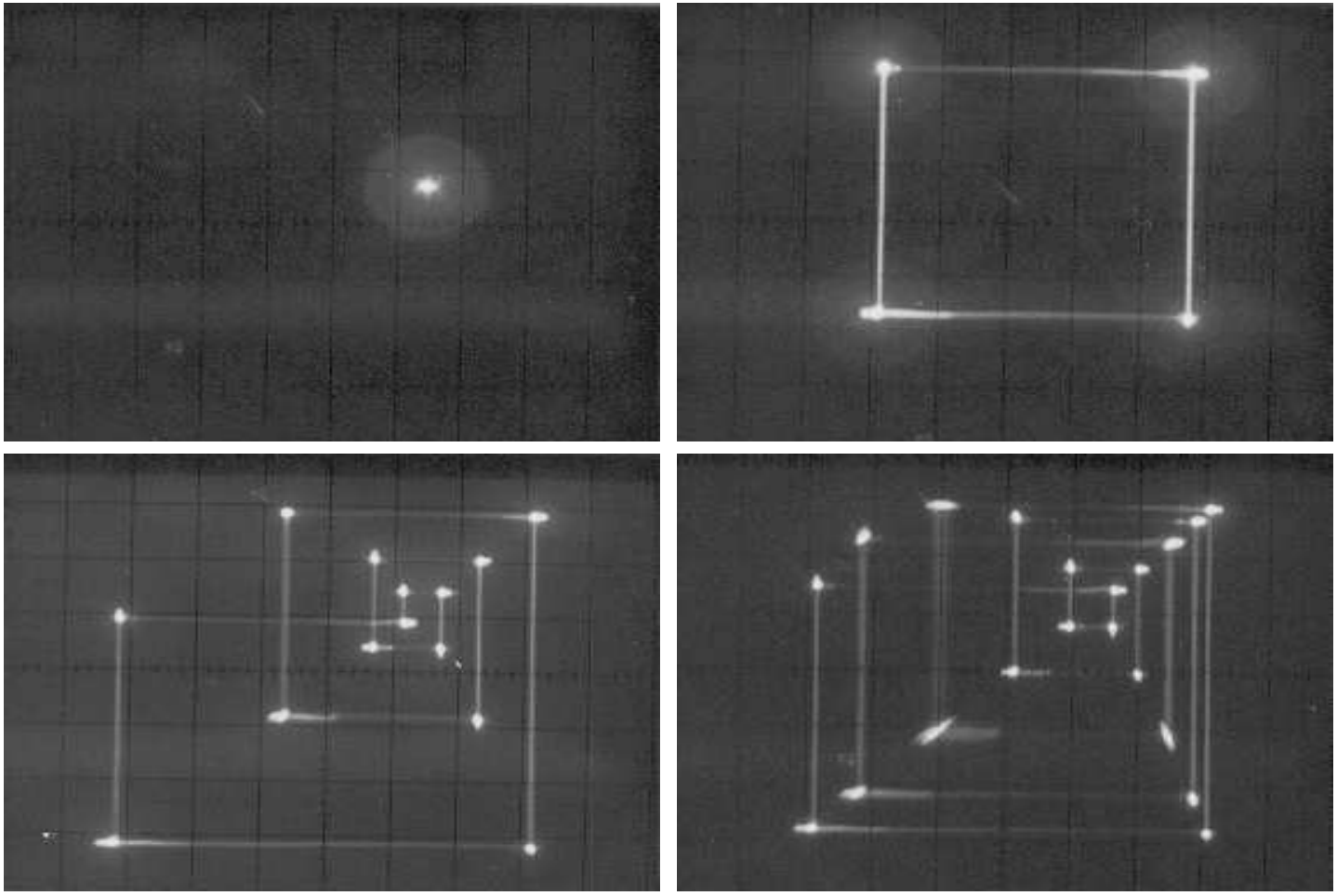
Chaotic Trace



6 - Cycle

## Circuit Realization of the Logistic Map

Murali, Sinha & Ditto, Physical Review E, 2003



Complete agreement with theoretical analysis

- Look-up table to directly extract widely varying temporal patterns
  Requires no run-time computations

- Look-up table to directly extract widely varying temporal patterns
  Requires no run-time computations

- Yields a wide range of response patterns from the same module
  Thus useful for designing components that can switch flexibly between different behaviours

- Look-up table to directly extract widely varying temporal patterns
  Requires no run-time computations

- Yields a wide range of response patterns from the same module
  Thus useful for designing components that can switch flexibly between different behaviours
- Transience is extremely short; Very robust

- Look-up table to directly extract widely varying temporal patterns
  Requires no run-time computations

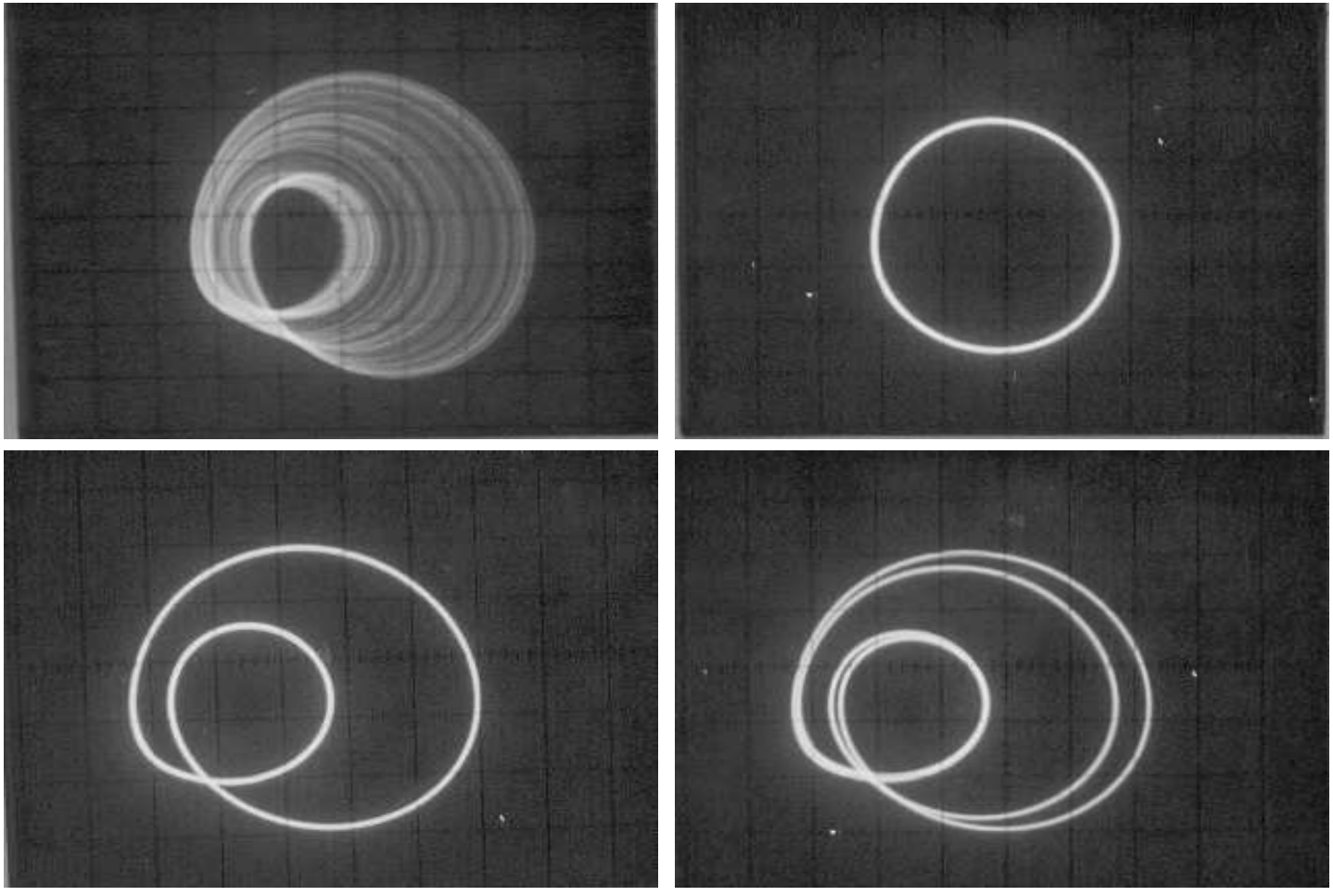
- Yields a wide range of response patterns from the same module
  Thus useful for designing components that can switch flexibly between different behaviours
- Transience is extremely short; Very robust
- Controller simple

## Does thresholding work beyond iterative 1d maps?

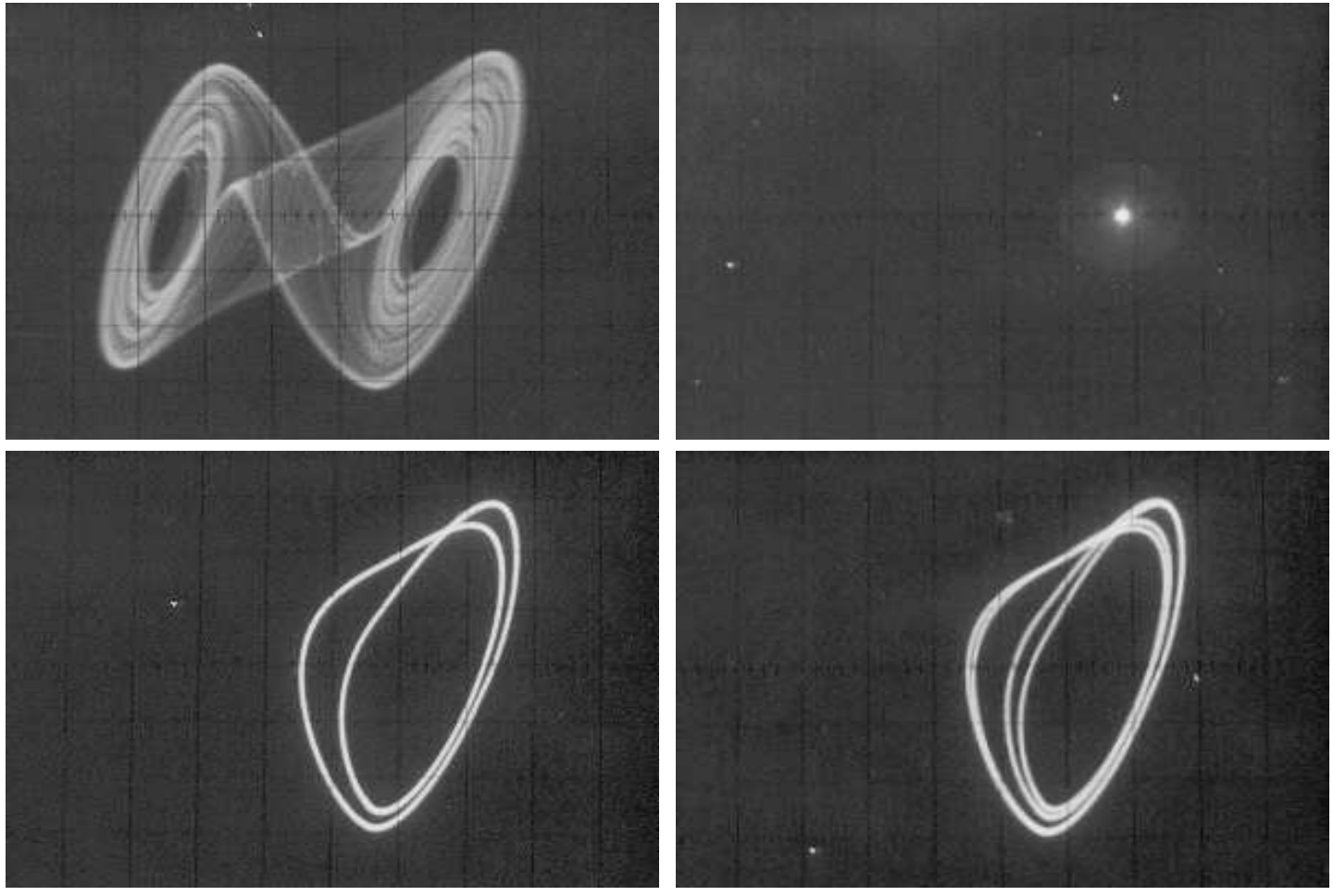
Can continuous time higher dimensional (possibly hyper-chaotic) systems be clipped?

No exact results : must rely on numerics and experimentation

# Circuit realization of coupled third order nonlinear differential equations



# Thresholding Chua's Circuit



Murali & Sinha, Physical Review E, 2003

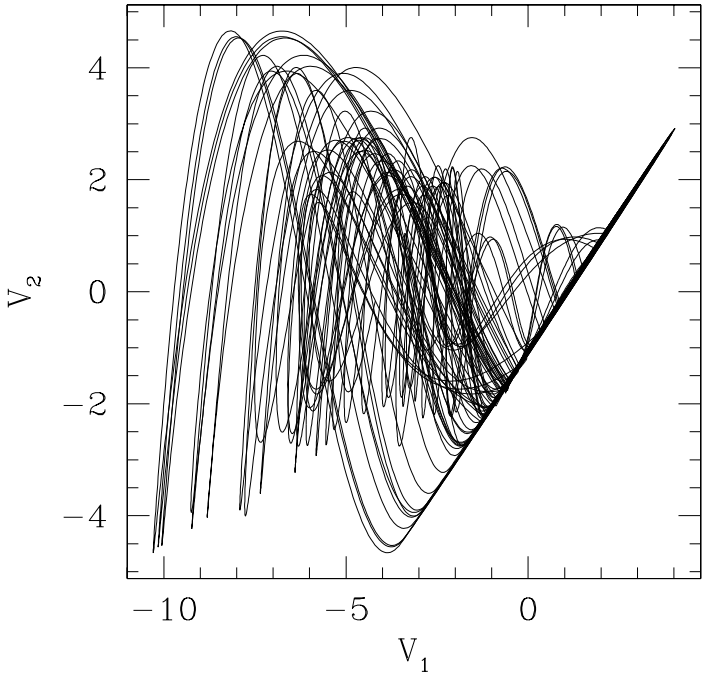
## Hyperchaotic Systems

Constitutes a stringent test of the control method

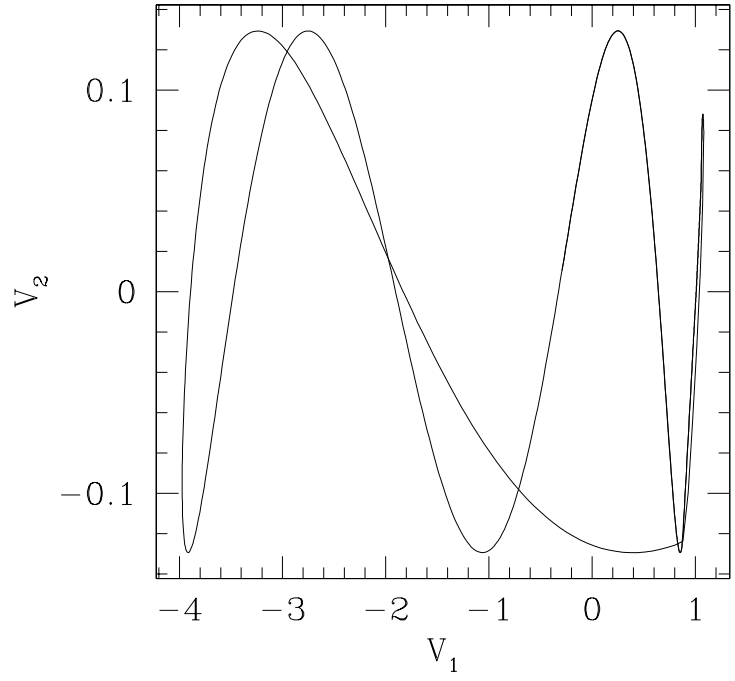
The system posseses more than one positive lyapunov exponent and thus has more than one unstable eigendirection to be reigned in

Thresholding : uses a single variable

## Hyper Chaotic Attractor



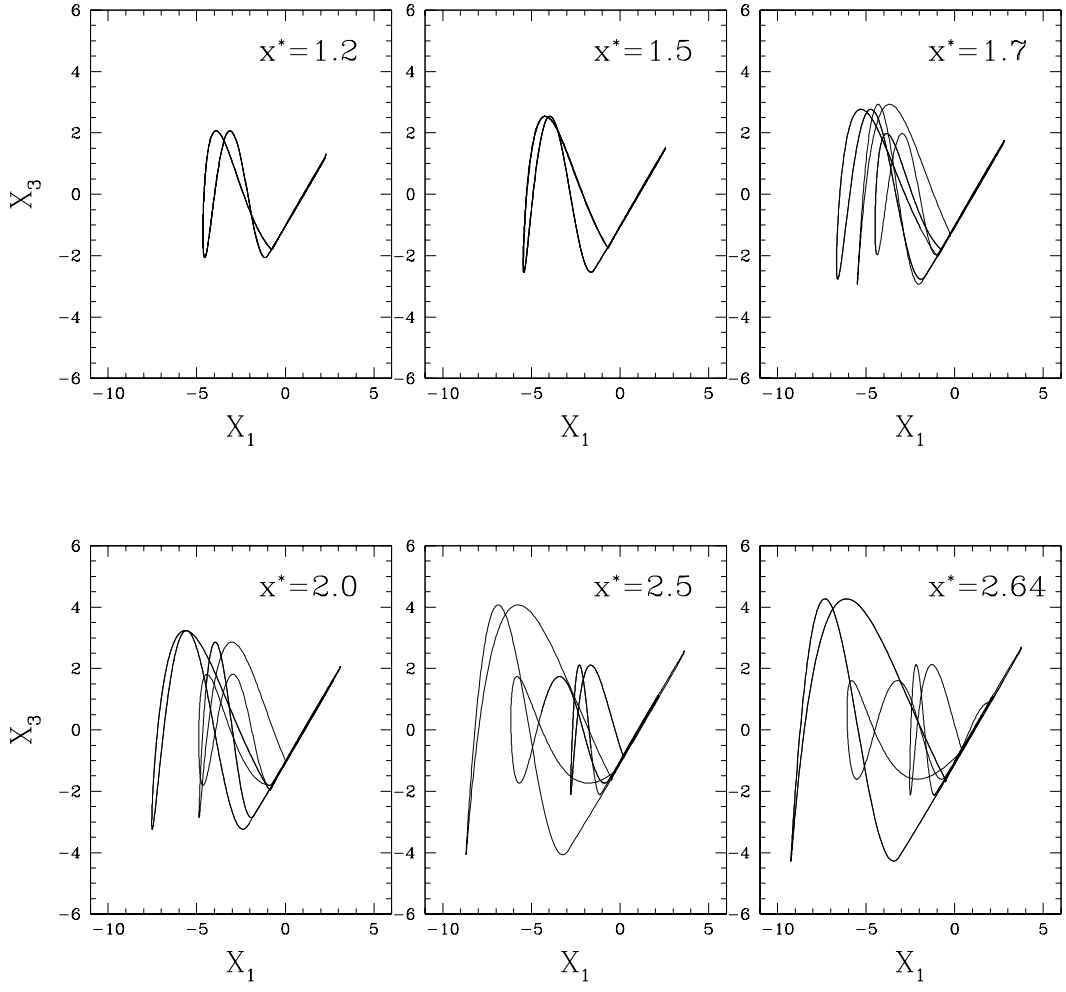

## Controlled Orbit




Murali & Sinha, Physical Review E, 2003

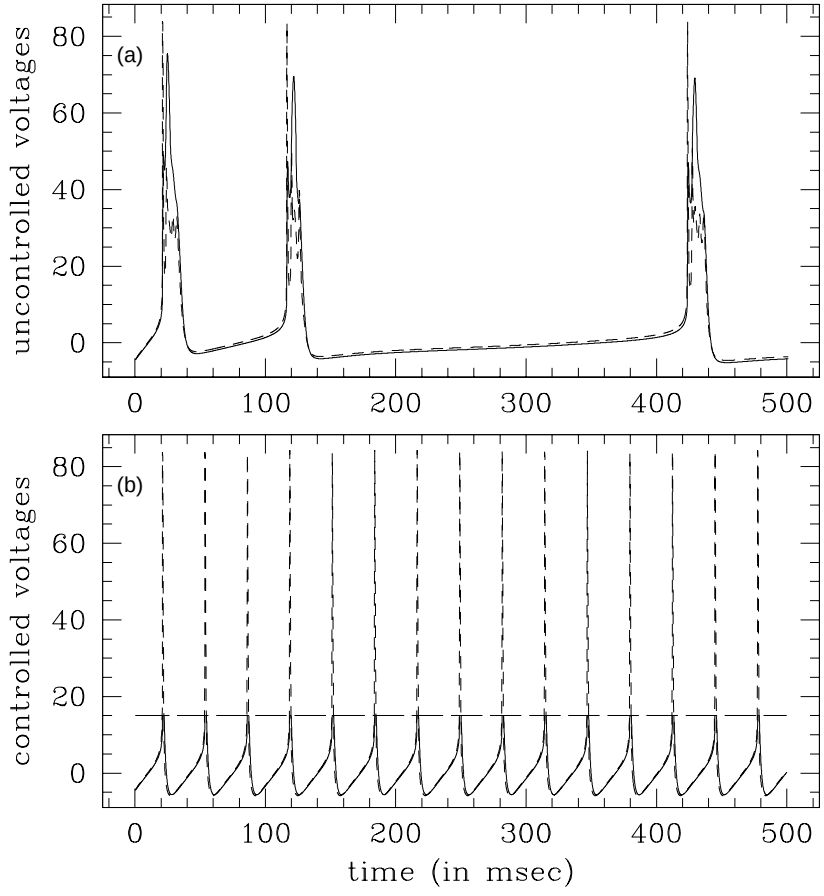
## Hyperchaotic electrical circuit

# Simple Thresholding selects out a very wide variety of patterns even in hyperchaotic systems



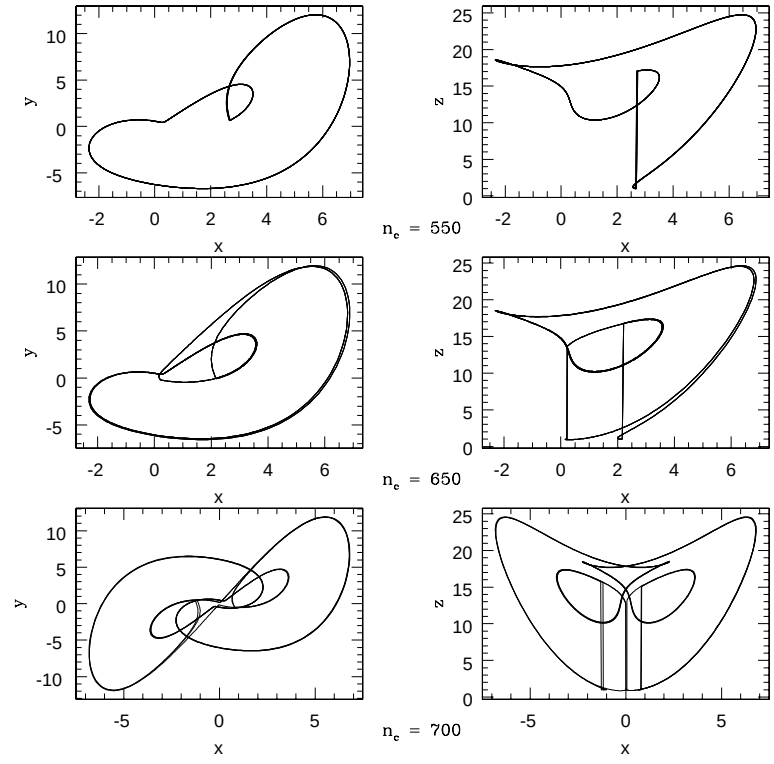
# Pinsky-Rinzel Neuron : Controlling Spiking

8 coupled ODEs : thresholding one variable




Sinha & Ditto, Physical Review E, 2001

# Varying control intervals offers flexibility in selecting different patterns




Sinha, Physical Review E, 2001

# Opportunities offered by Chaos

- Large range of controlled responses : Obtained from very simple mechanisms

# Opportunities offered by Chaos

- Large range of controlled responses : Obtained from very simple mechanisms
- Richness of temporal behaviour : can be used to obtain a wide range of temporal patterns

# Opportunities offered by Chaos

- Large range of controlled responses : Obtained from very simple mechanisms
- Richness of temporal behaviour : can be used to obtain a wide range of temporal patterns
- Determinism : allows reverse engineering

Hardware : Threshold activated chaotic elements
Chaotic Chip, Chaotic Processor

Programming these elements consists of fixing thresholds
such that some desired operation is performed
i.e. certain I/O relations are satisfied

Aim :

Implement all the basic logic gates flexibly using a chaotic element

With the ability to switch between different operational roles

This will allow a more dynamic architecture

Serve as ingredients of a general purpose device more flexible than statically wired hardware

## Implementing the fundamental NOR gate

A system is capable of universal general purpose computing if it can emulate a NOR gate

All basic logic operations can be constructed by combining the NOR operation

Provides a proof-of-principle demonstration of universal computing ability

Inputs : State of the chaotic element $x \equiv x^\star + I_1 + I_2$

Output : Excess emitted by threshold mechanism after chaotic update

$$O = f(x) - x^\star \quad \text{if} \quad f(x) > x^\star$$
$$O = 0 \quad \text{if} \quad f(x) < x^\star$$

## Necessary and sufficient conditions for NOR gate response

| $I_1$ | $I_2$ | Output | Condition to be satisfied simultaneously |
|---|---|---|---|
| $0$ | $0$ | $1$ | $f(x^\star + 0 + 0) - x^\star = I$ |
| $0/1$ | $1/0$ | $0$ | $f(x^\star + 0 + I) < x^\star$ |
| $1$ | $1$ | $0$ | $f(x^\star + I + I) < x^\star$ |

$I$ is a common positive constant for the operations

Input-output : equivalent

Enables gate elements to be wired easily into arrays

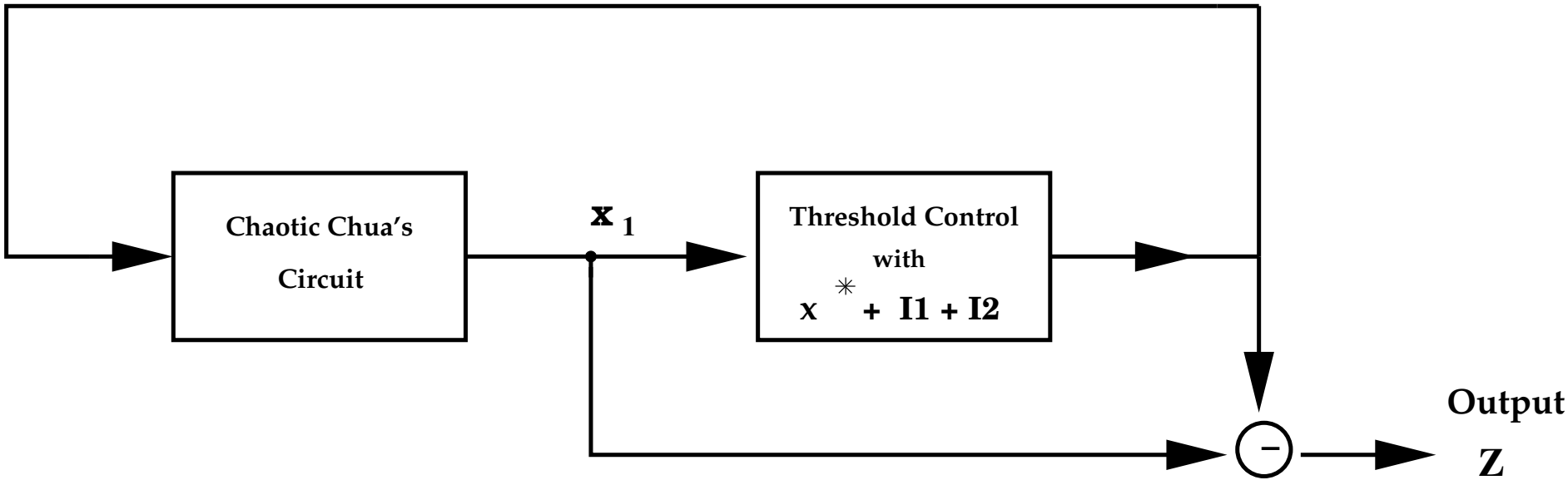
Logistic Map $f(x) = 4x(1-x)$

The NOR input-to-output mapping is realized in the threshold range 0.696 – 0.75

Wide and Robust operational range

Verified experimentally

## Inputs : Set threshold to $x^\star + I_1 + I_2$




Murali, Sinha & Ditto, Int. J. of Bif. & Chaos (Letts), Sept 2003

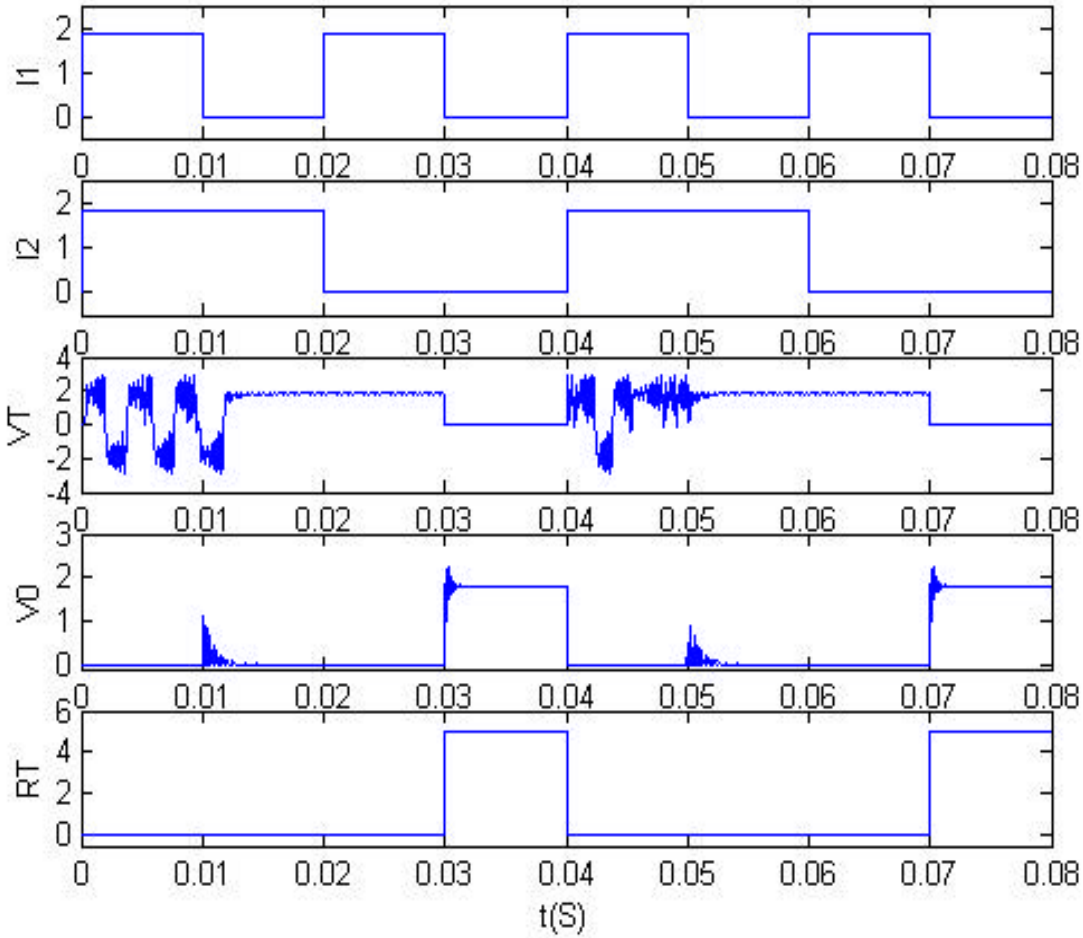



Fig.3. Timing sequences from top to bottom: (a) First input $I_1$, (b) Second input $I_2$, (c) Output VT, (d) Output $V_0$ and (e) Output NOR ($I_1$, $I_2$) obtained by thresholding.

All basic logic operations can be obtained by combination of the fundamental NOR gate

But such conversion processes are inefficient and has space costs
especially considering that such operations are performed billions of times

Aim : Direct implementation of all basic gates
Such as AND, OR, XOR, NOT and NAND

Proposed the direct implementation of all the logic gates which are basic and sufficient components of computer architecture today

The implementation is flexible : the same processor can serve as any of the gates by simple change of threshold setting

Sinha, Munakata & Ditto, Phys. Rev. E, 2002

Munakata, Sinha & Ditto, IEEE Trans. on Circuits and Systems, 2002

# Necessary and Sufficient conditions to be satisfied simultaneously

| AND | OR | XOR |
|---|---|---|
| $f(x_0) \leq x^*$ | $f(x_0) \leq x^*$ | $f(x_0) \leq x^*$ |
| $f(x_0 + I) \leq x^*$ | $f(x_0 + I) - x^* = I$ | $f(x_0 + I) - x^* = I$ |
| $f(x_0 + 2I) - x^* = I$ | $f(x_0 + 2I) - x^* = I$ | $f(x_0 + 2I) \leq x^*$ |

| NAND | NOT |
|---|---|
| $f(x_0) - x^* = I$ | $f(x_0) - x^* = I$ |
| $f(x_0 + I) - x^* = I$ | $f(x_0 + I) \leq x^*$ |
| $f(x_0 + 2I) \leq x^*$ | |

| Operation | AND | OR | XOR | NAND | NOT |
|---|---|---|---|---|---|
| $x_0$ | $0$ | $\frac{1}{8}$ | $\frac{1}{4}$ | $\frac{3}{8}$ | $\frac{1}{2}$ |
| $x^*$ | $\frac{3}{4}$ | $\frac{11}{16}$ | $\frac{3}{4}$ | $\frac{11}{16}$ | $\frac{3}{4}$ |

## Scheme has been experimentally verified

Simple mechanism allows one to switch with ease between behaviours emulating different logic gates

## This provides sufficient ingredients for directly implementing all operations

Including bit-by-bit arithmetic and memory

Contrast with periodic elements:

It is not possible to extract all the different logic responses from the same element in case of periodic components, as the temporal patterns are inherently limited.

Contrast with random elements:

One cannot design components : need determinism for reverse engineering

Only Chaotic dynamics enjoys both

richness
and
determinism

So one can select out all the different temporal responses necessary to obtain all the different logic patters with a single evolution function

This ability allows us to construct flexible hardware

## Implementation of Parallel Logic Operations

Objective : Obtain $N$ clearly defined logic gate response patterns from the $N$ components characterizing the state of a $N$-dimensional system

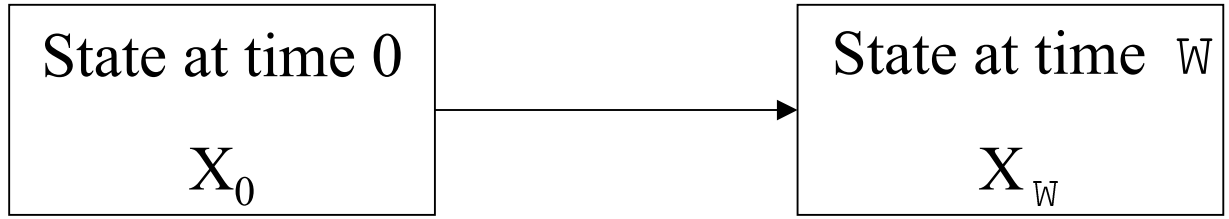
This will enable us to implement $N$ operations in parallel with a single $N$-dimensional element

Thus one can gain processing power without enhancing space costs

***Can execute several operations concurrently***

Dynamical System

of dimension N: $\{X\} = (x_1, x_2, \ldots, x_{N)}$




Encodes N inputs

$(I_1, I_2, \ldots, I_N)$

Encodes N outputs

$(O_1, O_2, \ldots, O_N)$

## Implementation of parallel logic by a 2-dimensional map

Consider a 2-d model for neurons (Aihara, Chialvo):

$$x_{n+1} = x_n^2 \ \exp(y_n - x_n) + k$$

$$y_{n+1} = a y_n - b x_n + c$$

Implement bit-by-bit addition in parallel

| $I_1$ | $I_2$ | $O_1$ | $O_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0/1 | 1/0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Bit by bit arithmetic addition

*Involves 2 logic operations:*

$O_1$ is the first digit of the sum (rightmost) : determined by XOR

$O_2$ is the carry of the answer :determined by AND

x variable implements XOR

y variable implements AND

## Necessary and sufficient conditions for parallelized bit-by-bit arithmetic addition

| Initial State | XOR | AND |
|---|---|---|
| $x^\star,\ \ y^\star$ | $x_n < x^\star$ | $y_n < y^\star$ |
| $x^\star + I,\ \ y^\star + I$ | $x_n = x^\star + I$ | $y_n < y^\star$ |
| $x^\star + 2I,\ \ y^\star + 2I$ | $x_n < x^\star$ | $y_n = y^\star + I$ |

$I$ is a common positive constant for the operations

Large bands of solutions exist, satisfying the table of simultaneous conditions

Similar considerations for other parallel logic operations can be straight-forwardly formulated

Proposed a flexible chaos based computing paradigm

Richness of the dynamics allows one to select out all the different requisite responses from the same processor

Universal General Purpose computing device

More versatile than static hardware

Arrays of such flexible units can conceivably be programmed on the run to give the optimal hardware for the task at hand

For instance, may serve flexibly as an arithmetic processing unit or a component of memory, as the need demands, and can be swapped to be one the other

Programmable hardware ; Re-configurable hardware

# Building blocks of a Dynamical Logic Architecture

- Pre-determined dynamic logic configuration

# Building blocks of a Dynamical Logic Architecture

- Pre-determined dynamic logic configuration
- Out-come dependent dynamic logic configuration

Possibility of the hardware design evolving during the computation

Attempted to harness the abundantly available chaotic phenomena in engineered and natural systems for the development of a novel computing device