# Analog- and Hybridcomputing
# Past, Present, Future

Prof. Dr. Bernd Ulmann

14 February 2020

## Why analog/hybrid computing?

Why is it worthwhile to look into a different computing paradigm like analog computing in the 21st century?

- Traditional stored program digital computers reach fundamental boundaries (energy consumption grows quadratically with clock frequency, on-chip-structures cannot shrink much further etc.), which may be overcome by specialized co-processors.
- Analog computers are not true general purpose computers but excel at problems that can be described by (systems of) differential equations.
- Analog computers exhibit an extremely high energy-efficiency.
- Analog computers are inherently parallel and do not suffer from AMDAHL's law etc.

# What is an analog computer?

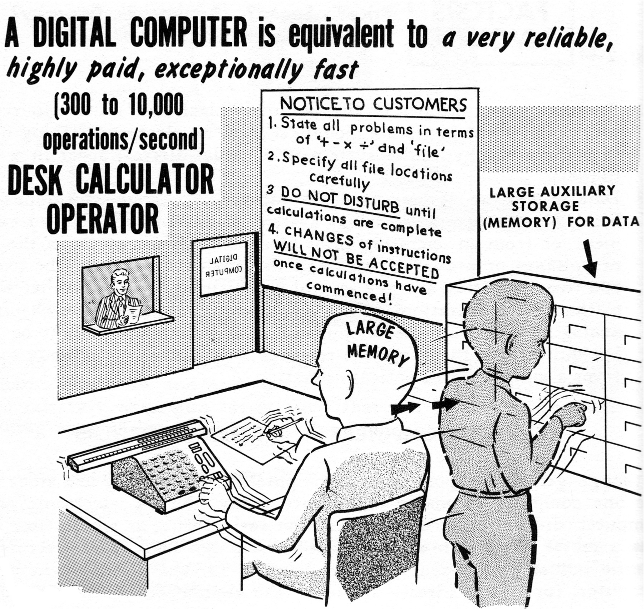
## What is an analog computer?

An *analog computer* is a machine with the following characteristics:

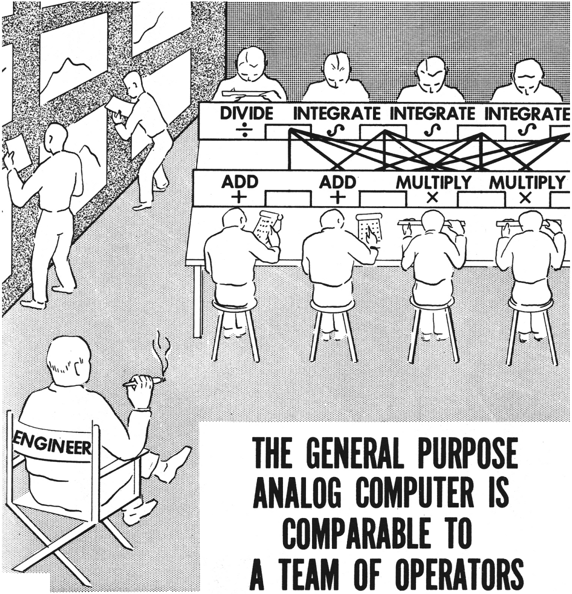
- It contains a plethora of computing elements such as integrators (!), summers, multipliers etc.
- These computing elements are connected to each other in order to form an *analogue* for the problem to be solved (that's where the name stems from).
- It has no memory in the traditional sense as there is no algorithm controlling the machine's operation.
- All computing elements work in full parallelism exchanging time varying values by means of their connections.
- Values are typically (but not necessarily so) represented as continuous voltages or currents.

The following (historic) pictures show the main differences between a digital computer and an analog computer:[1]

[1] Cf. [Truitt et al.(1960), pp. 1-40/41].


A DIGITAL COMPUTER is equivalent to a very reliable, highly paid, exceptionally fast
(300 to 10,000 operations/second)
DESK CALCULATOR OPERATOR
NOTICE TO CUSTOMERS
1. State all problems in terms of '+ - × ÷' and 'file'
2. Specify all file locations carefully
3 DO NOT DISTURB until calculations are complete
4. CHANGES of instructions WILL NOT BE ACCEPTED once calculations have commenced!
DIGITAL COMPUTER
LARGE AUXILIARY STORAGE (MEMORY) FOR DATA
LARGE MEMORY

# Analog computer



Basically there are two types of analogies on which an analog computer can be based:

Direct analogies: Here, a physical process similar to the problem being investigated is used. Classic examples are models using Nylon stockings to model plate structures in mechanics (like the roof of the Munich olympic stadium) etc.

Indirect analogies: An indirect analog computer uses a different underlying principle than the problem to be solved. These are more general then direct analog computers and are described here. These machines typically consist of a multitude of individual computing elements which can be implemented in various ways. Analog electronic computing elements are most common.

What are the advantages of an analog electronic analog computer?

- Continuous value representation – a value is transferred through a single line and not in a discrete bit-serial or -parallel way which saves a lot of power since there is no need to dis-/charge millions or billions of parasitic capacitors billions of times per second.
- The computing elements work in full parallelism, no memory accesses, no need for explicit synchronisation etc.
- Numerical stability is no problem at all.
- Easily adapted to sensors/actors in an analog world. :-)

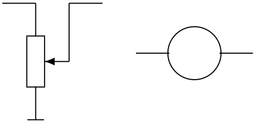
# Disadvantages

Since there is nothing like a free lunch, analog computers have their own share of disadvantages:

- Limited precision – typically about 3 to 4 decimal places (it turns out that this is not a limiting factor for most practical applications).
- Only time is directly available as free variable of integration, so solving partial differential equations requires some transformations/tricks.
- Generating arbitrary functions, especially those having more than one argument, is difficult.
- Values are limited to the interval $[-1, 1]$, so problems must typically be *scaled* accordingly.

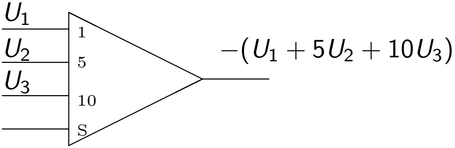
What computing elements comprise a typical analog computer?

- Coefficient units (e. g. a voltage divider/multiplying DAC etc.)



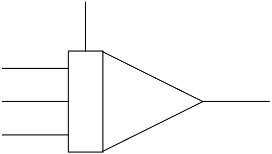
- Summers, computing

$$-U_{\text{out}} = \sum_{i=1}^{n} a_i U_i$$



- Integrators, computing[2]

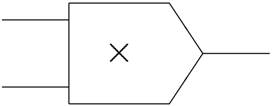
$$-U_{\text{out}} = \int\limits_0^t \sum_{i=1}^n a_i U_i(t)\,\mathrm{d}t + U(0)$$



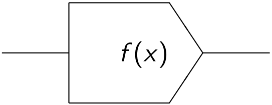
An integrator has three modes of operation: *initial condition*, *operate*, *halt*.

[2] The "magic component" – (near) perfect integration with a resistor, a capacitor and an operational amplifier.

- Multipliers




- *Open amplifiers* which can be used to implement inverse functions such as square root, division as well as special functions like limiters etc.
- Comparators with electronic switches
- Function generators (traditionally using polygonal interpolation, nowadays table-lookups using ADCs and DACs)



# Basic programming – ODEs

## Programming

Programming an analog computer is much simpler than programming a stored program digital computer as there is nothing like an intricate algorithm controlling the system.

An analog computer programm, i. e. the interconnection scheme for its computing elements, is derived from the problem equations.

A classic way to develop an analog computer setup is due to Lord KELVIN:

1. Solve the equation for the highest derivative.
2. Generate all lower derivatives by repeated integrations.
3. Derive all terms on the right hand side of the equation based on these lower derivatives.
4. Tie all these terms together. According to the equation this process started with, this must be equal to the highest derivative, so this value can be fed back into the circuit as this highest derivative.

Often, a sine/cosine signal pair is required in a simulation. A typical way to generate this on an analog computer is by solving the DEQ

$$\ddot{y} + \omega^2 y = 0 \tag{1}$$

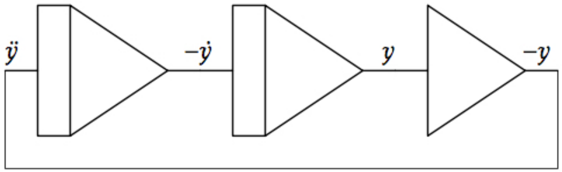
with the initial conditions

$$y_0 = a\sin(\varphi) \text{ und}$$
$$\dot{y}_0 = a\omega\cos(\varphi)$$

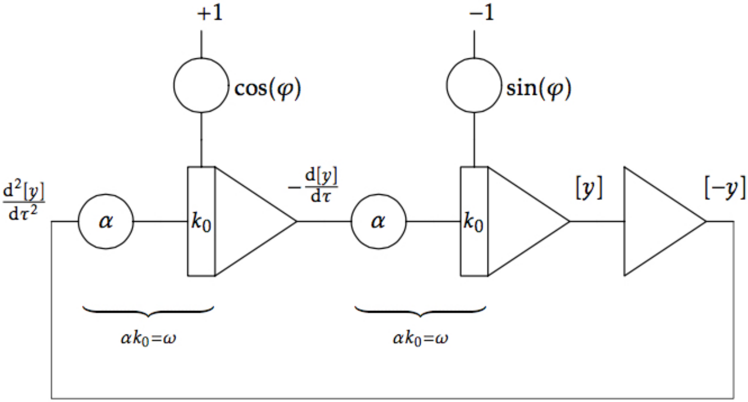
Solving (1) for $\ddot{y}$ yields

$$\ddot{y} = -\omega^2 y.$$

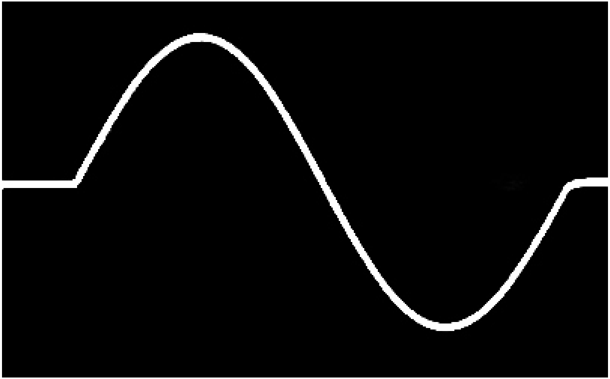
Ignoring $\omega^2$, this can be transformed into this program sketch:



Taking $\omega$ into account as well as the initial conditions yields the following program:



This yields the following output at the second integrator (actual screen shot – the analog computer was running for the time of one period as determined by $\omega$):



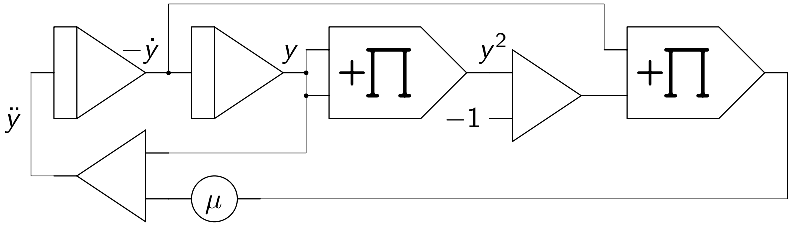
A more complex example is the van der Pol equation

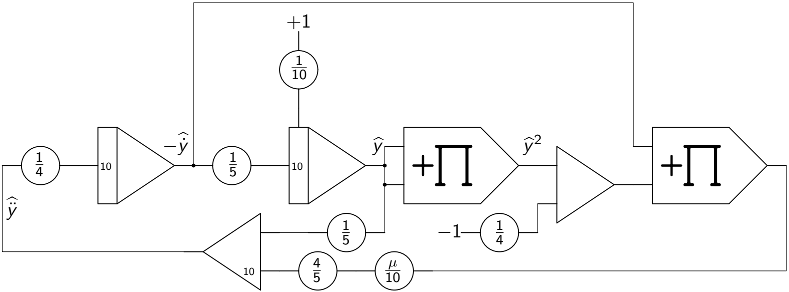
$$\ddot{y} + \mu\left(y^2 - 1\right)\dot{y} + y = 0,$$

which can be rearranged, yielding

$$\ddot{y} = -y - \mu\left(y^2 - 1\right)\dot{y}.$$

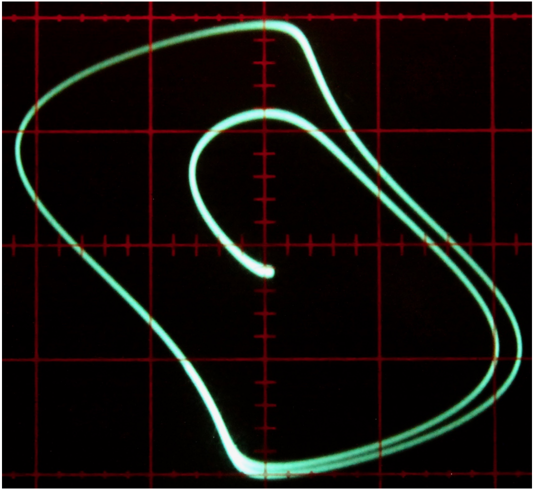
The unscaled analog computer program looks like this:






A typical phase space plot generated with this setup looks like this:

# van der Pol equation



# What about PDEs?

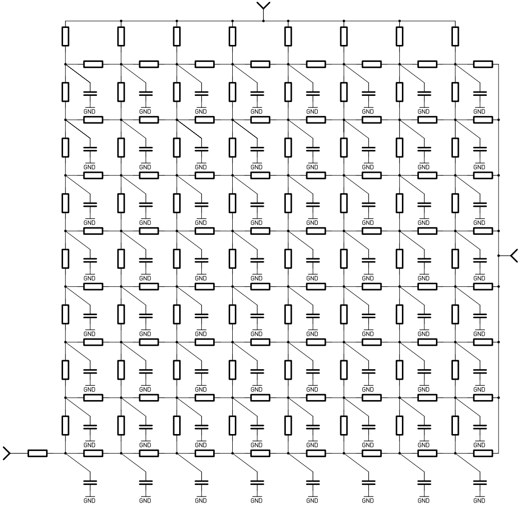
# What about PDEs?

There are several ways to treat PDEs on an analog computer:

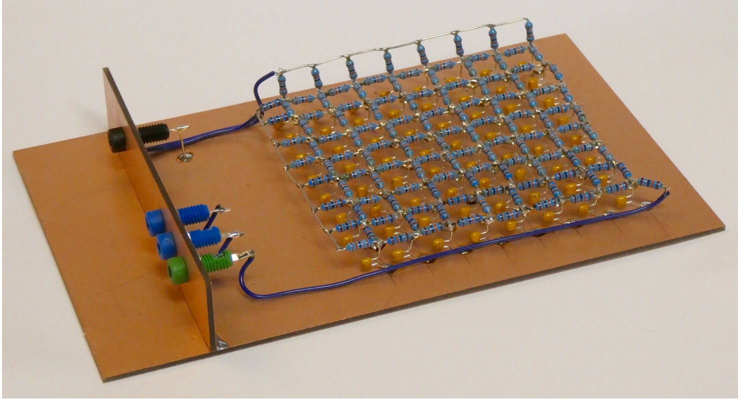
- Seperation of variables
- Transform the PDE into ODEs by means of the LAPLACE-transform
- Replace derivatives by finite differences
  - Typically time remains continuous while $x$, $y$, $z$ are discretized
  - It is also possible to discretize with respect to time and have one continuous coordinate in space
- Use an analog computer together with a digital computer, essentially forming a *hybrid computer*

## A passive analogue

The following example shows a simple passive model for the two-dimensional heat-equation:

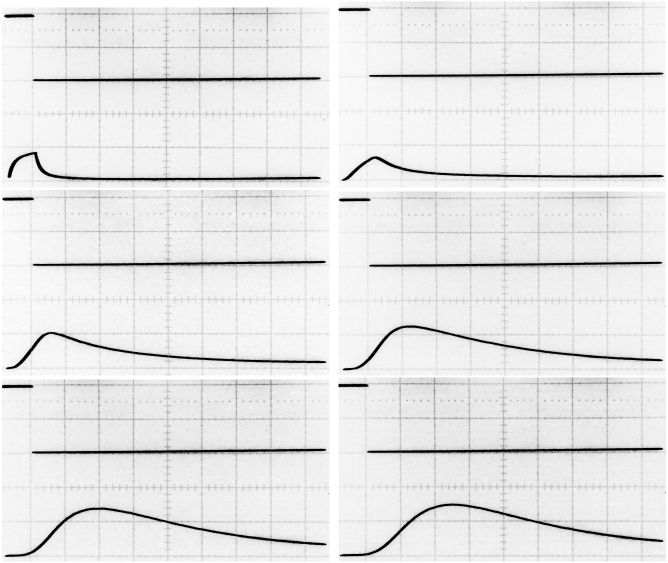


# A passive analogue

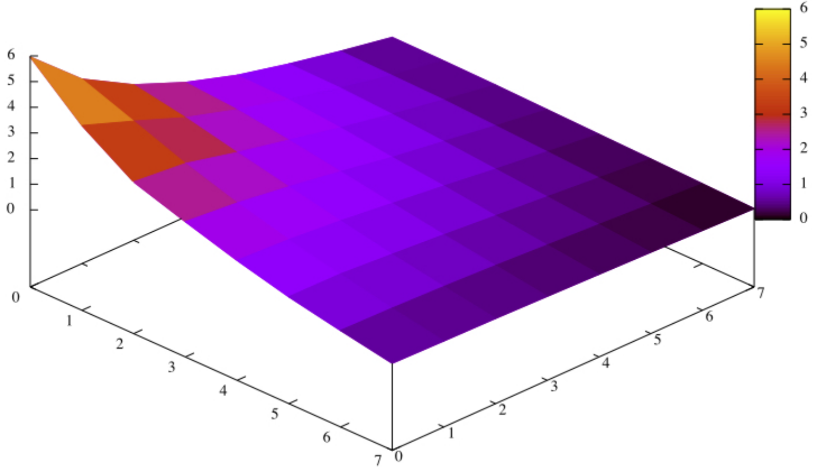


## A passive analogue

Response along the diagonal elements to an input pulse at $u_{0,0}$:



Steady state solution:



An analog computer setup for the two-dimensional heat-equation starts with

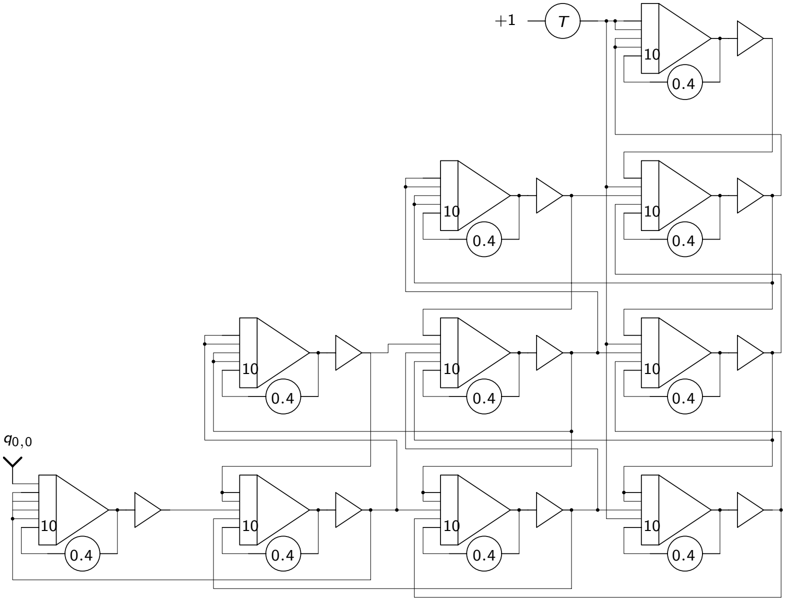
$$\dot{u} = \alpha \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right).$$

Since an analog computer can only integrate with respect to time, an obvious aproach would be to replace the right-hand side of the above equation by a difference term:

$$\dot{u}_{i,j} = \alpha \left( u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j} \right) + q_{i,j}.$$

This can now be treated as shown before. In the following one octant of a quadratic sheet is modelled (taking a symmetry argument into account):

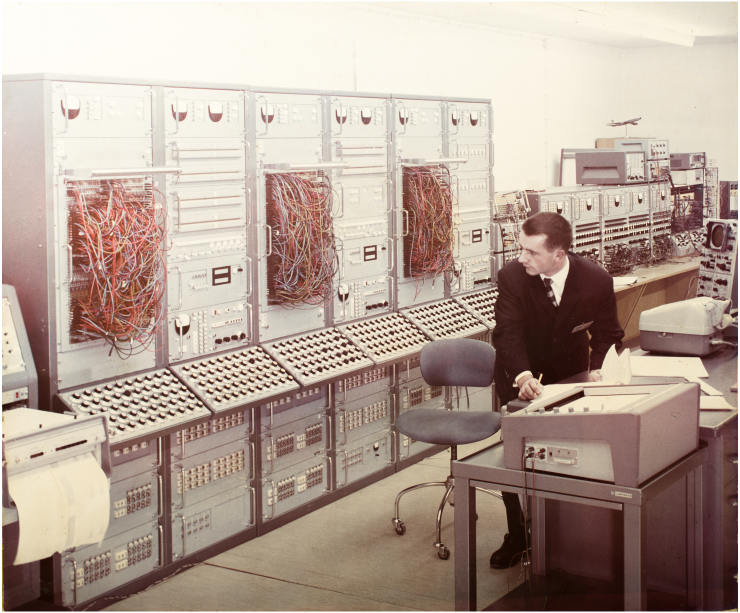
# An active analogue



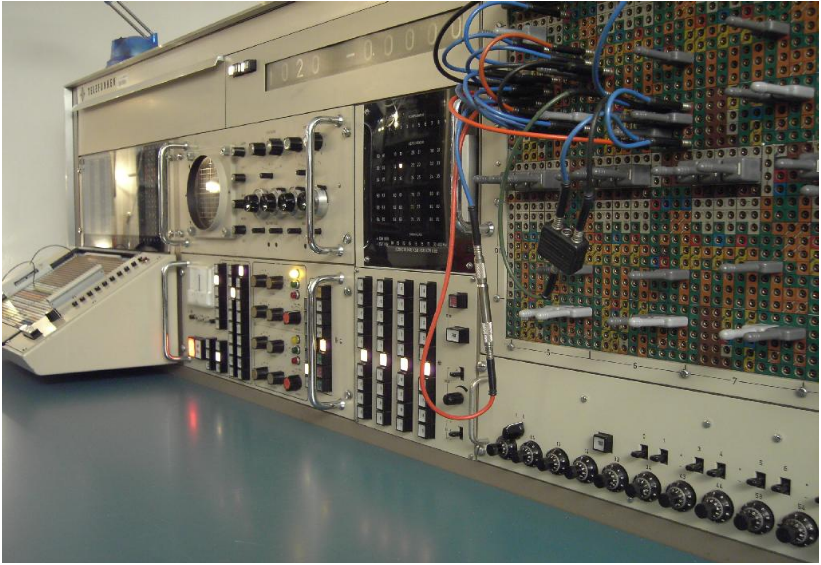
# Historic analog computers

- Analog computers were dominant machines from the late 1940s to about the mid-1970s.
- They were (and still are) much faster than stored-program digital computers and were used extensively in aerospace technology, vehicle design, chemistry, medicine etc.
- Analog computers were superseded by stored-program digital computers mainly due to maintenance issues, price and programmability

# Telefunken RA800, 1960

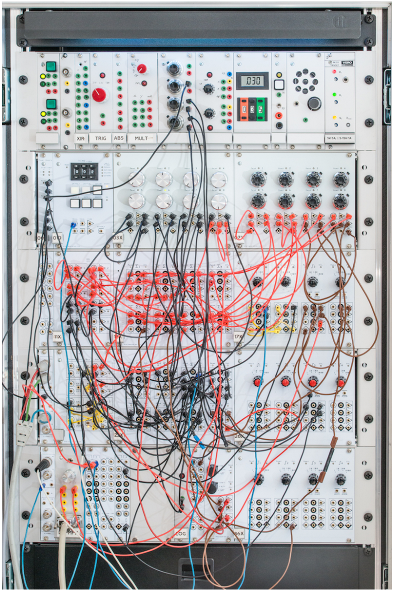


# Telefunken RA770, 1966

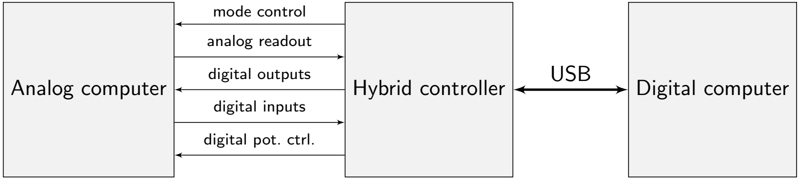


# VW car simulator, 1970s

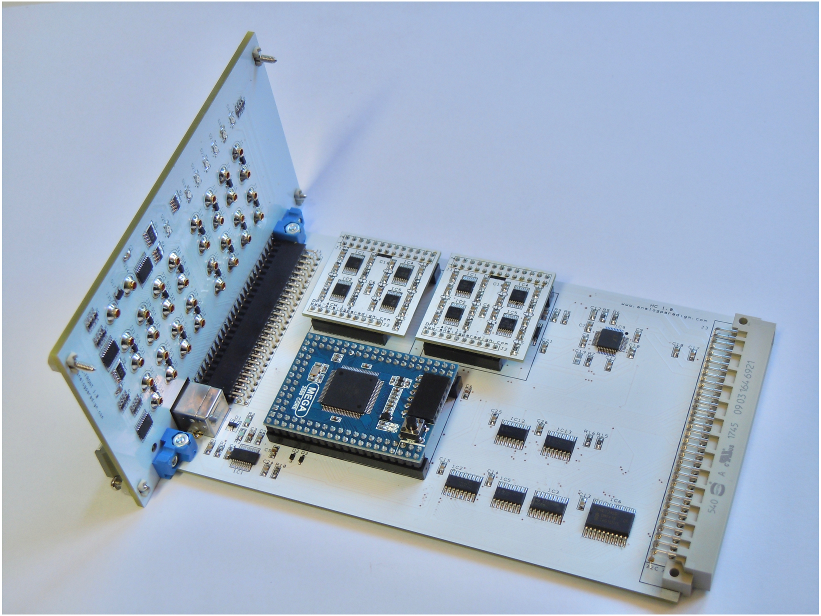
# Present analog computers


XIR
TRIG
ABS
MULT

# Hybrid controller



# Hybrid controller



Future analog computers

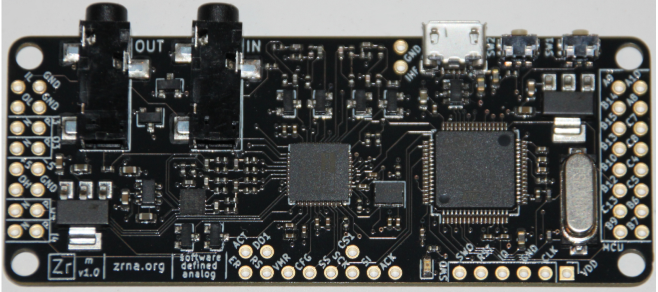
…on chip…

Traditional analog computers have some severe drawbacks that must be overcome for future applications:

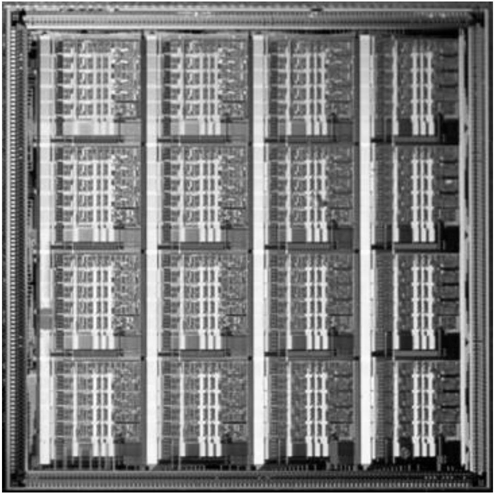
- Its discrete implementation using operational amplifiers and passive electronic components is no longer practical and does not scale well.
- Setting up a traditional analog computer is way too cumbersome:
  - The patch panel programming takes too much time.
  - Changing programs is a mechanical process (removable patch panels).
  - Setting up coefficients and function generators is time consuming and error prone.

There are already some modern developments such as these:

Anadigm manufactures FPAAs (using switched capacitors for the central interconnect structure) mainly aimed at the signal processing market. A typical application example is shown here (`zrna.org`):

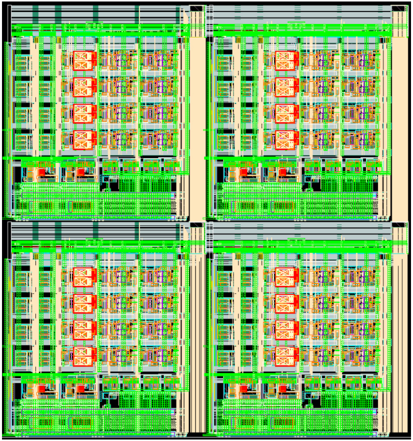


First fully reconfigurable analog computer on chip – mainly used for stochastic differential equations:[3]



[3] See [Cowan(2005)].

More advanced analog computer on chip:[4]



(Picture source: http://yipenghuang.com/wp-content/uploads/2016/07/v2_chip.png, retrieved 2019-09-19.

[4] See [Guo et al.(2016)].

## Problems

- These and other chips are for quite special purposes (signal pre-/postprocessing, artificial intelligence etc.).
- The most general chips are academic projects with no intentions to become production ready hardware.
- Software support is minimal to say the least...
- The computing elements often exhibit rather low bandwidth (about 20 kHz).
- Static and dynamic errors are pretty large.

# Applications

What are typical applications of an analog computer on chip (either used standalone or as part of a hybrid computer setup)?

- High performance computing
- Artificial intelligence (implementation/training of ANNs etc.) and machine learning
- Medical applications (cardiac pacemakers, brain pacemakers, insulin sensing/pump control) – the power consumption could be sufficiently low to power some implants with energy harvesting thus getting rid of clumsy batteries which limit life span and have to be recharged etc.
- Industrial control systems (basically stateless, not hackable in a traditional sense)
- Signal pre-/postprocessing for mobile devices and the like
- Monte-Carlo simulations, financial mathematics

# Future developments

We, a team with strong academic and industrial background, are currently in the process of founding a company devoted to the development of a true general-purpose recofigurable analog computer on a chip.

At the moment, we

- develop novel computing elements for CMOS integration offering high precision at high bandwidths.
- investigate interconnect structures (research is done on a reconfigurable Analog Paradigm Model-1 analog computer in order to determine the required switch structure for a reconfigurable analog computer on chip).
- develop a basic software infrastructure for modern hybrid computing.

What challenges have to be solved?

- Tight integration with the digital computer to form a hybrid computer is not easy.
- This requires real-time operation on the side of the digital computer.
- Numerical algorithms to refine (rough) solutions obtained by means of an analog computer are necessary.
- Ideally, the programmer must not be aware of the intricacies of analog computer programming. This requires
    - a hardware definition language for the analog computer domain.
    - an accompanying compiler yielding the bitstreams required to configure the analog computer on chip.

Thank you for your interest!

Happy analog computing![5]

[5] The author would like to thank Dr. Chris Giles for proofreading this set of slides.

# Bibliography

GLENN EDWARD RUSSELL COWAN, *A VLSI Analog Computer / Math Co-processor for a Digital Computer*, Columbia University, 2005

N. GUO, Y. HUANG, T. MAI, S. PATIL, C. CAO, M. SEOK, S. SETHUMADHAVAN, and Y. TSIVIDIS, "Energy-Efficient Hybrid Analog/Digital Approximate Computation in Continuous Time", in *IEEE Journal of Solid-State Circuits*, vol. 51, no. 7, pp. 1514-1524, July 2016

Thos. D. Truitt, A. E. Rogers, *Basics of Analog Computers*, John F. Rider Publisher, Inc., New York, December 1960