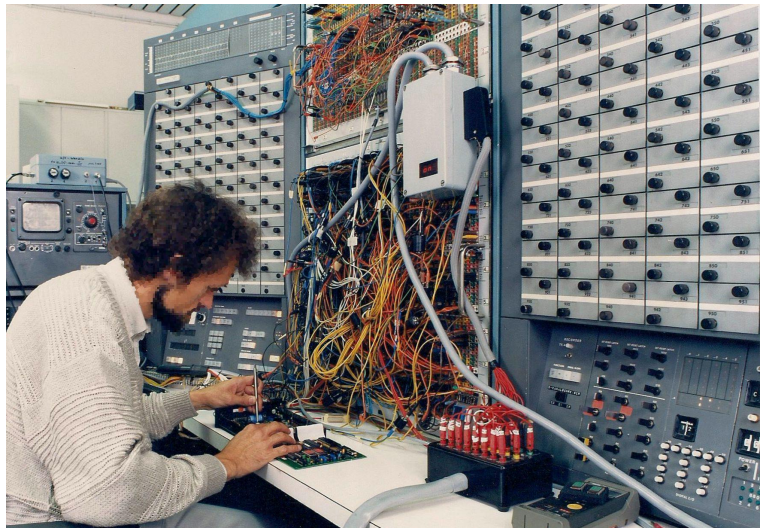
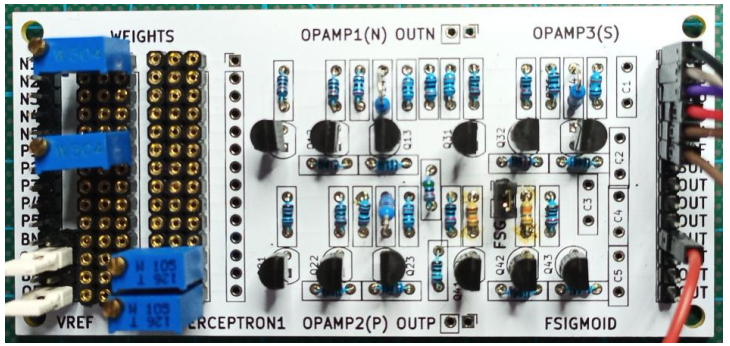
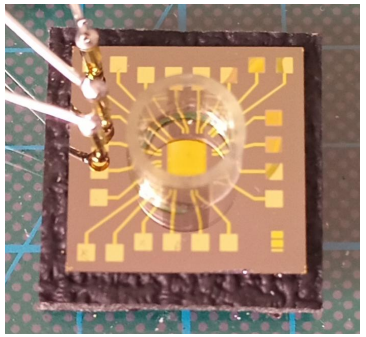
STEFAN BOSSE[1,2*]

[1] Institute of Computer Science
Researchgroup Practical Computer Science
University of Koblenz

[2] Department of Mechanical Engineering
Lehrstuhl für Materialkunde und Werkstoffprüfung
University of Siegen







DESIGN OF ANALOG COMPUTERS: BUILDING BLOCKS AND ADVANCED METHODS FOR TINY ANALOG MACHINE LEARNING WITH SURROGATE MODELING

Stefan Bosse

Sysint 2025 Conference

university of koblenz
Computer Science

CONTENT

01 **Introduction**
Motivation: In-sensor computation and organic electronics

02 **Analog Computers**
Electronic circuits for numerical computations - from history to challenges and Imits

03 **Analog ANN (AANN)**
Analog Artificial Neural Networks - Design and Training Methodologies

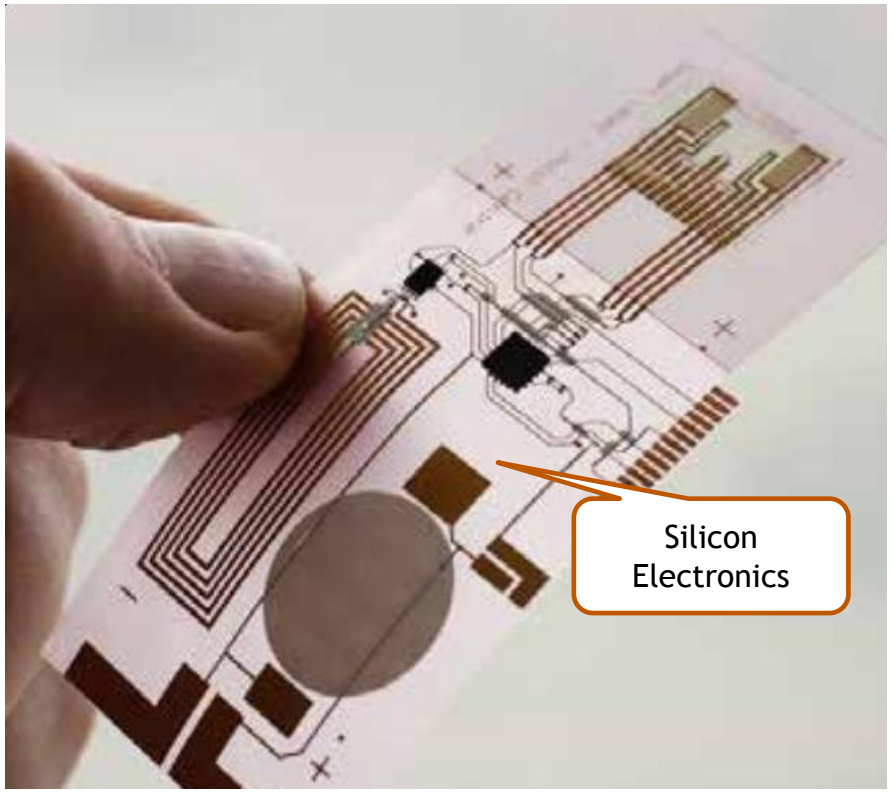
04 **Surrogate Models**
Using Electronic Simulation and Surrogate ML Models for Training and Test of AANN

05 **Discussion**
Workflow - Experiments - Examples

06 **Conclusions and Outlook**
Issues and pitfalls
Lessons learned

INTRODCUTION

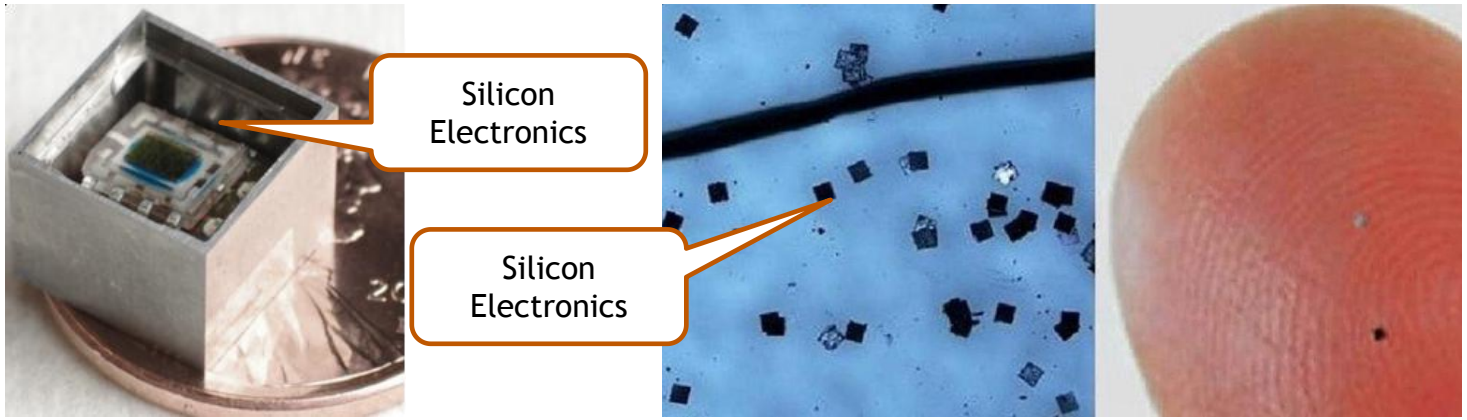
- (Distributed) **Sensor networks** are deployed in a wide range of applications and environments. A sensor network is a distributed system given by a connected graph of communicating sensor nodes. **Each sensor node measures physical properties of its local environment.**
- **Sensor density increased exponentially** and sensors are integrated systems („Smart Sensors“).



FlexSmell Project with silicon electronics on flexible substrates

Assumption: Data in sensor networks is inherently distributed and must be processed locally on sensor node level: In-sensor Computation.

INTRODCUTION



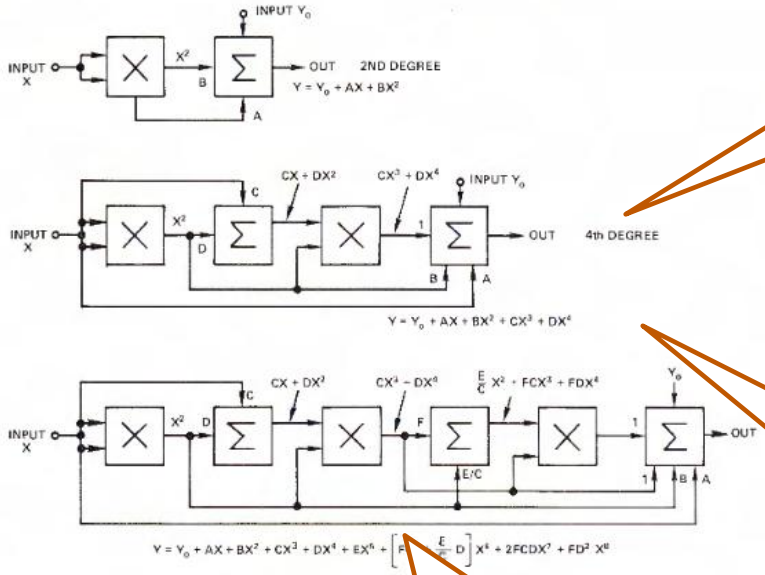
(Left) MEMS for Distributed Wireless Sensor Networks, Warnecke et al. (Right) Smart Dust - Hitachi

- Digital computing based on the **binary number system** is the standard for any numerical computation since about 60 years. Digital computers are capable to perform highly complex **numerical computations**, with only a **small set of instructions** using high-level programming languages and compilers.
- With ongoing miniaturization, **computation is integrated in sensors and devices** towards material-integrated sensor networks (in-sensor computation). But the miniaturization towards the 1 mm^3 scale **reduces computational power and memory capacity** significantly.

Hypothesis: Analog (electronic) circuits can perform numerical computations such as Artificial Neural Networks with lower ressources and electrical energy than digital circuits (and eventually faster).

WHY ANALOG PROCESSING?

Sum-of-Product (SOP):
Digital: > 1000 Transistors
Analog: < 10 Transistors



Dynamic/Resolution f(x):
Digital: Discrete, 8-32 Bits
Analog: Cont. > Noise

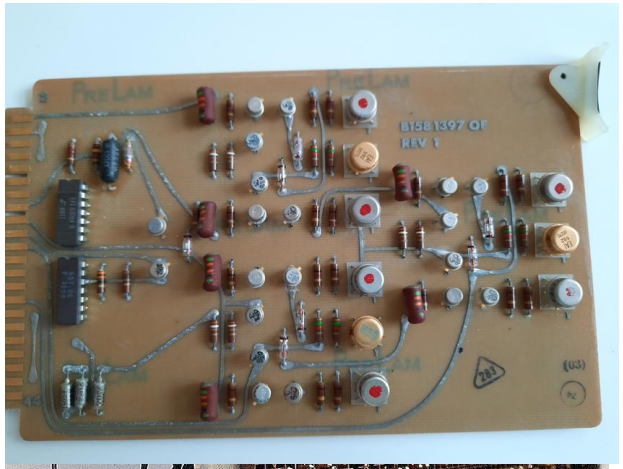
Printed Organic Electronic:
10-100 Transistors / mm^2
Digital: Not suitable
Analog: Suitable

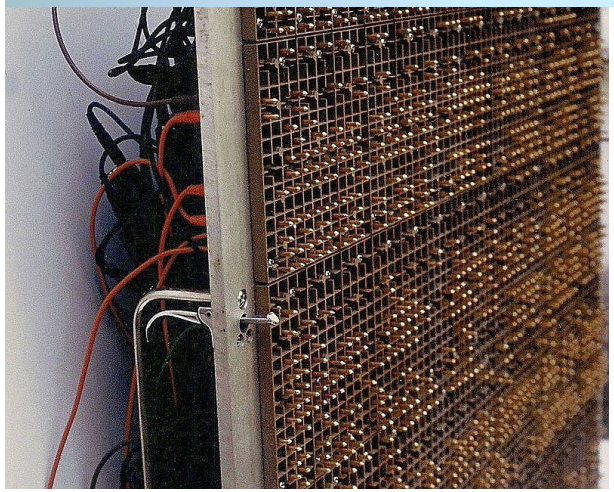
Start-up time:
Digital: 1 s (µC)
Analog: 0

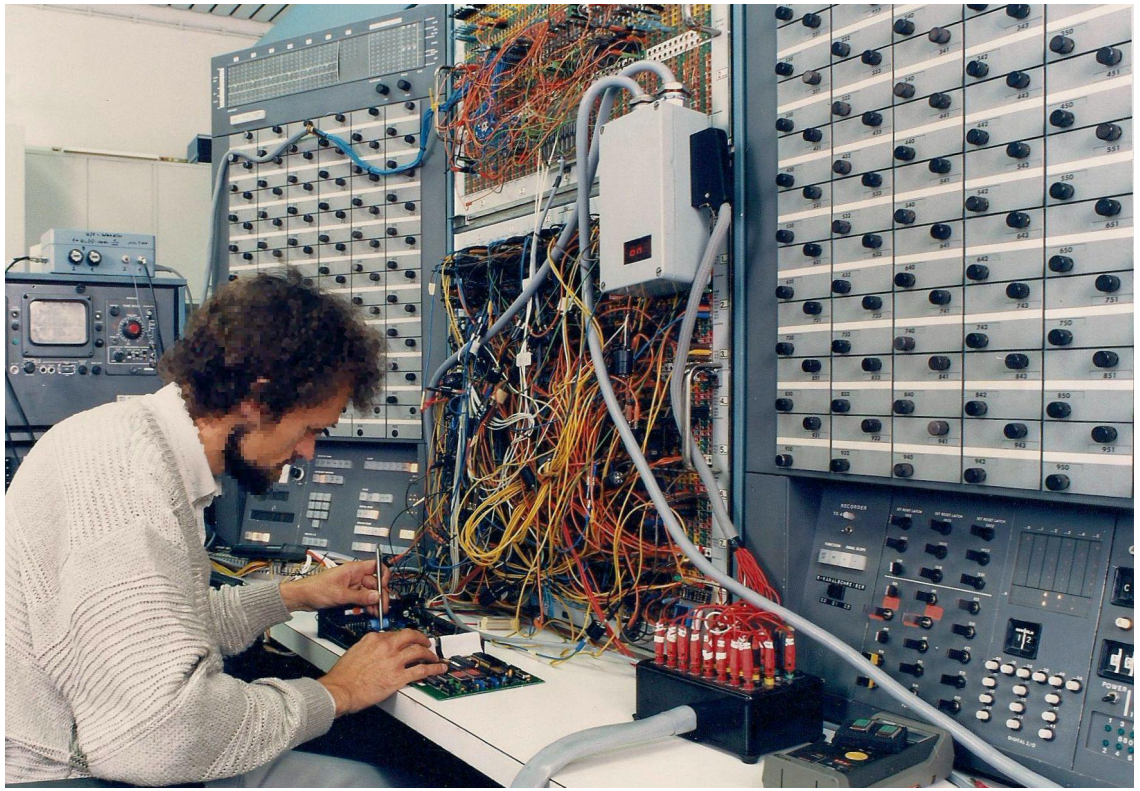
Solving Diff. Eq.:
Digital: 1 s, 1 MB
Analog: 1 ms, 6 Capacitors

ANALOG COMPUTERS: THE HIGH-PERFORMANCE CLASSICS...

- **Digital: Discrete Value Distribution**
- **Analog: Continuous Value Distribution (t,s)**
- **Basic Model: Ideal Operational Amplifier**
- **Functional Composition: Wire Interconnect**
- **Parameters: Variable Resistors**
- **Technology: Silicon Electronics**







Analog Computer EAI8800 (1986) with Hardware-in-the-loop!

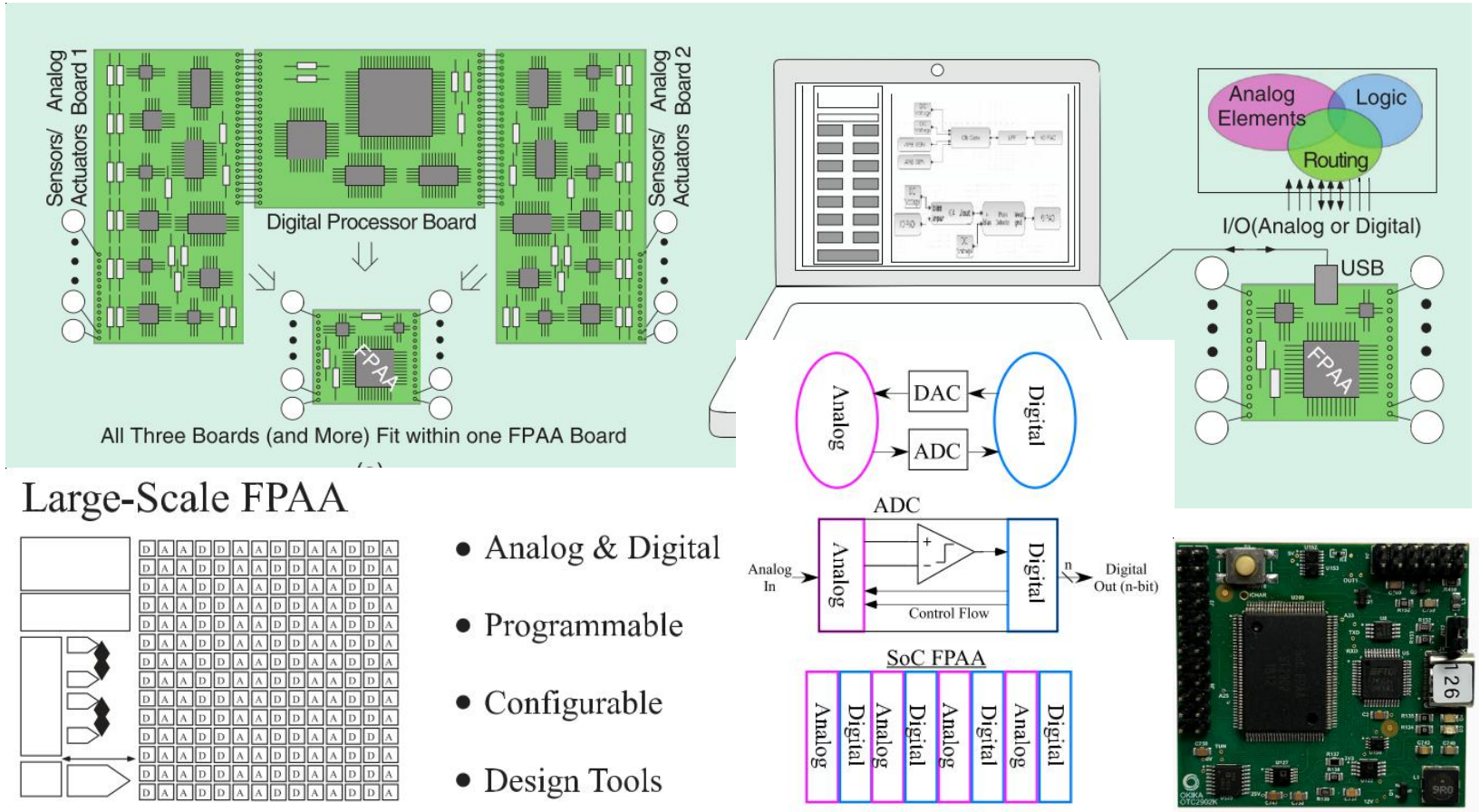
ANALOG COMPUTERS: THE ANCIENT!

- **Digital: Discrete Value Distribution**
- **Analog: Continuous Value Distribution (t,s)**
- **Basic Model: Ideal Operational Amplifier**
- **Functional Composition: Wire Interconnect**
- **Parameters: Variable Resistors**
- **Technology: Silicon Electronics**

Analog Computer EAI8800 (1993) with rust!

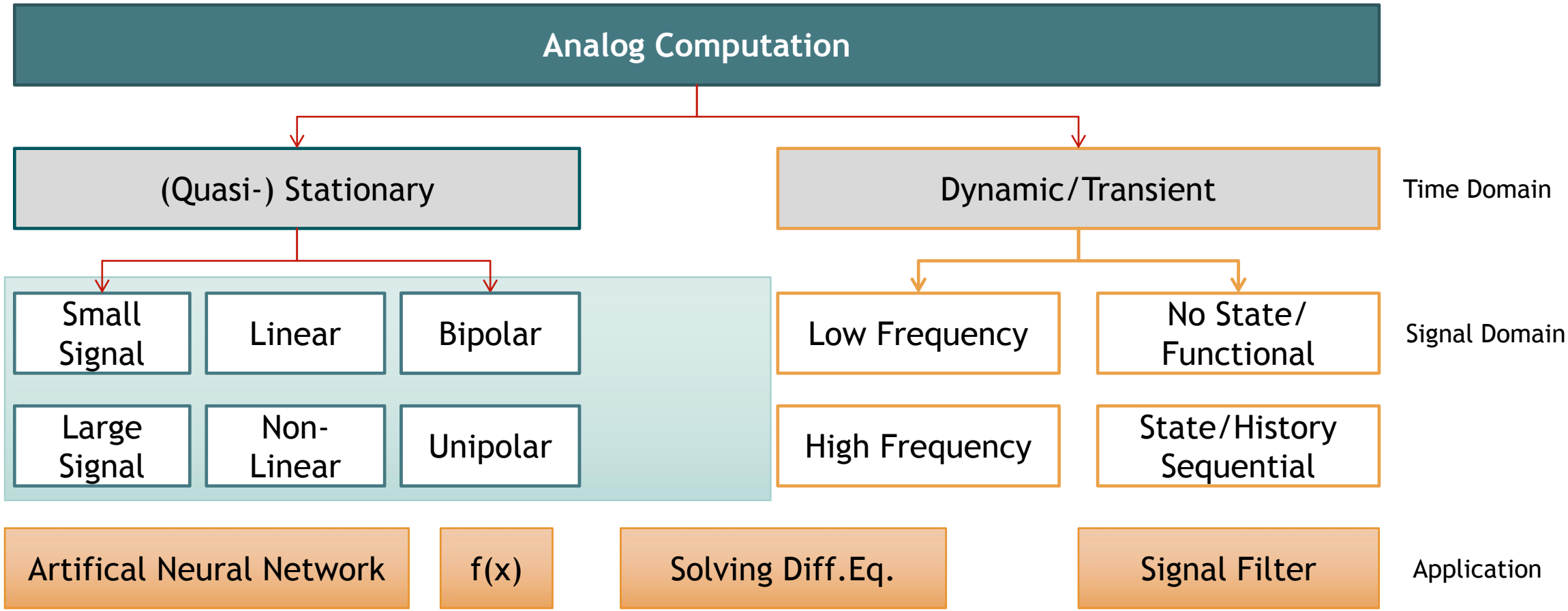
ANALOG COMPUTERS: PROGRAMMABLE CHIPS - NEW AGE?

- **Mixed-Analog-Digital: Semi- Continuous Value Distribution (t‘,s‘)**
- **Basic Model: Mixed A/D, OPAMP, Transistors, AD/DA Conversion**
- **Functional Composition: Switched Matrix**
- **Parameters: Digital Resistors, Switched Capacitors**
- **Technology: Silicon Electronics**

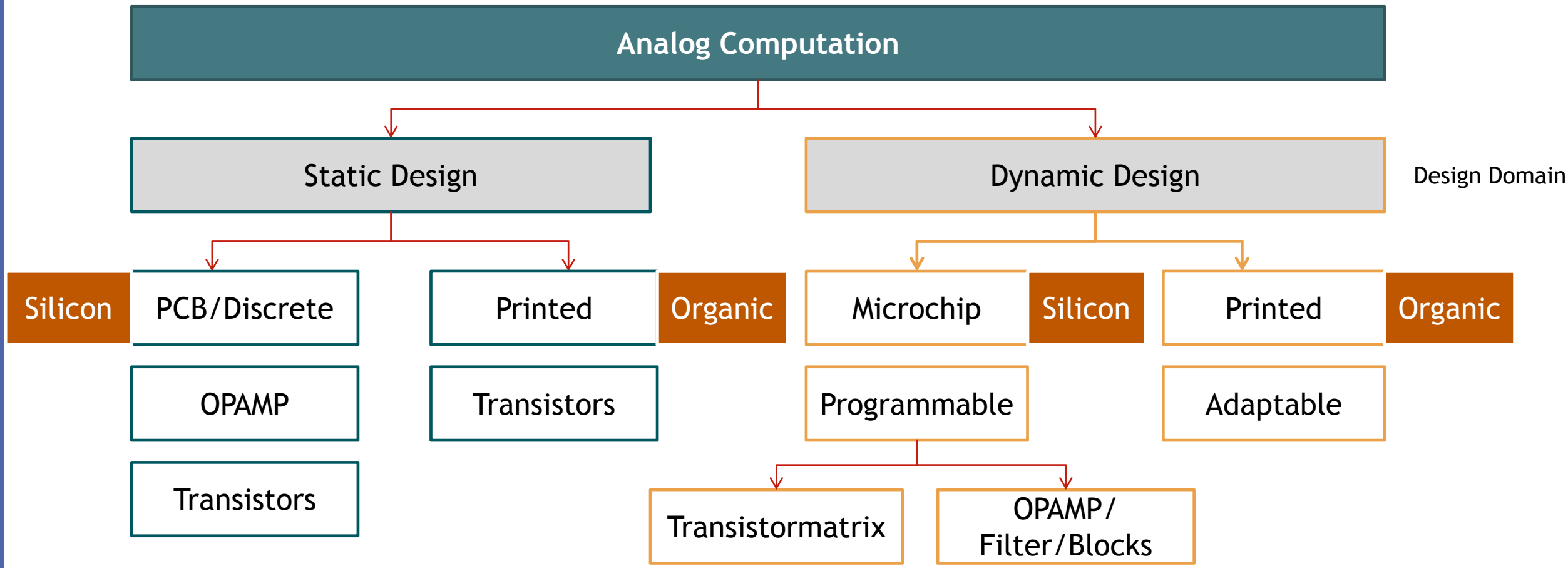


Field Programmable Analog Array Computer FPAA (2003-2024) on a chip! [Hasler et al..Okika dev.]

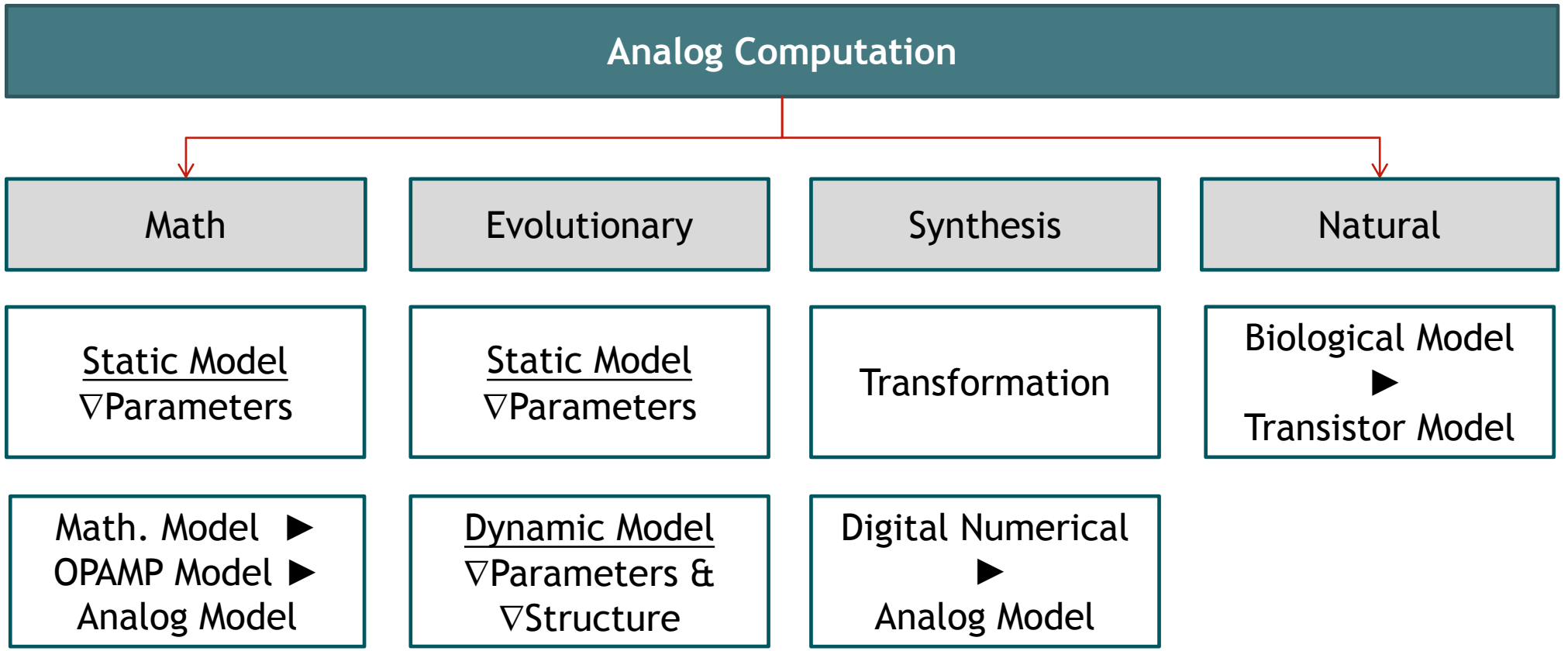
ANALOG COMPUTATIONAL SYSTEMS: TAXONOMY



ANALOG COMPUTATIONAL SYSTEMS: TAXONOMY

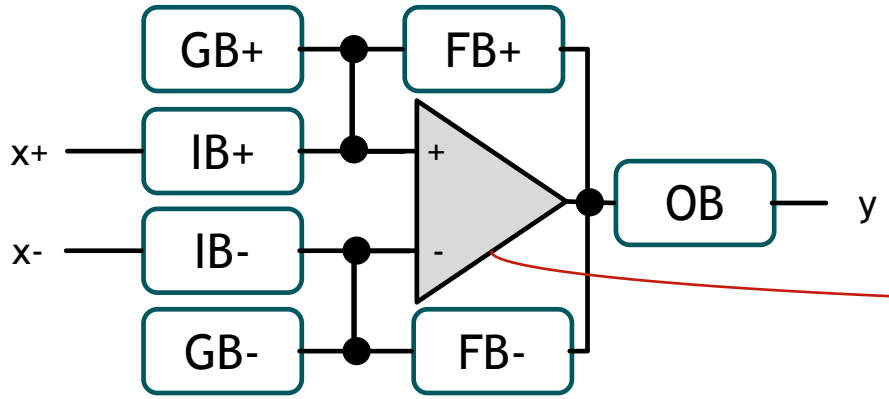


ANALOG COMPUTATIONAL SYSTEMS: TAXONOMY

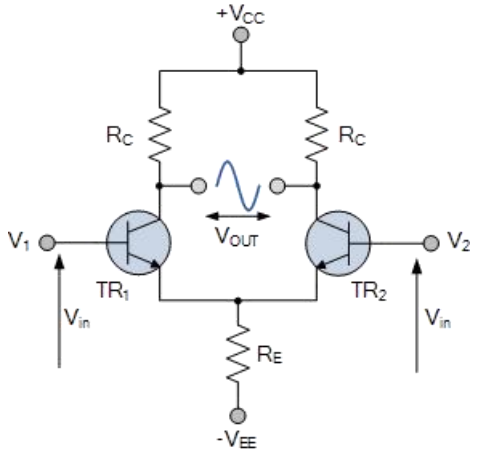


ANALOG COMPUTATIONAL SYSTEMS: OPAMP MODEL

- **The Operational Amplifier (OPAMP) is the basic cell of any accurate analog computational system!.**
- An OPAMP is a difference amplifier with an inverting and a non-inverting input i_+ and i_-, respectively. There is one output *o*.
- An OPAMP can implement a(ny) (perhaps time-dependent) function $y=f(x)$ by adding up to 7 functional blocks (e.e., resistive) defining the transfer function of the entire circuit:

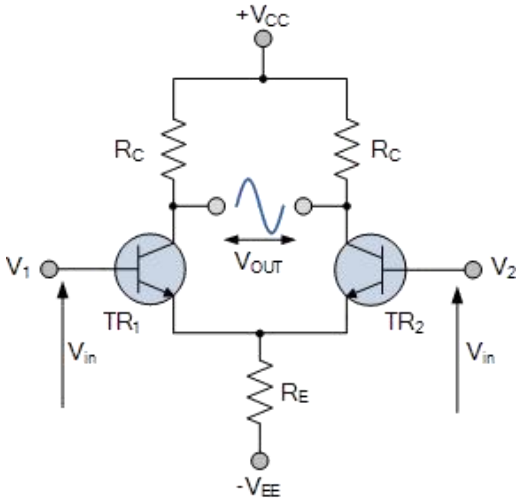


IB: Input Block
OB: Output Block
GB: Ground Block
FB: Feedback Block

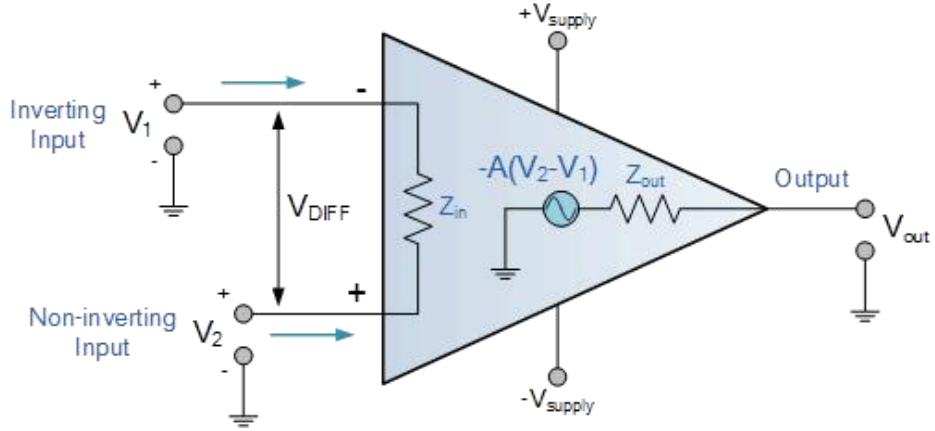


ANALOG COMPUTATIONAL SYSTEMS: OPAMP MODEL

- The mathematical model == ideal OPAMP features an infinite (or realistic very large) open-loop gain (G_0), no offset/bias or any other technical deviation, infinite common-mode rejection ratio (CMRR), zero noise, and **infinite output voltage range**.
- It is a difference amplifier, i.e.:



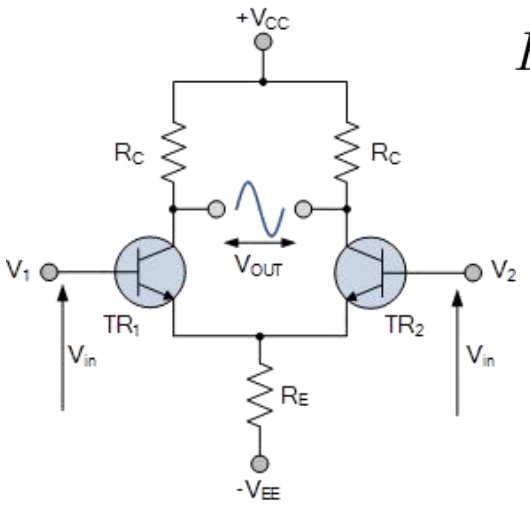
$$V_{out} = \Delta V G_0, \Delta V = V_+ - V_-$$

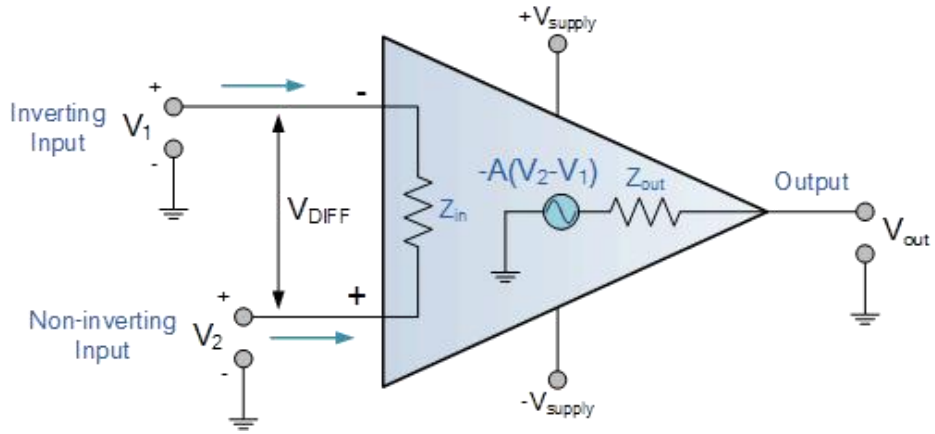


ANALOG COMPUTATIONAL SYSTEMS: OPAMP MODEL

- **Real transistor circuits show significant deviation from the ideal mathematical model!**
- The (static) error *E* is introduced by a function Γ, which depends on a large set of parameters:

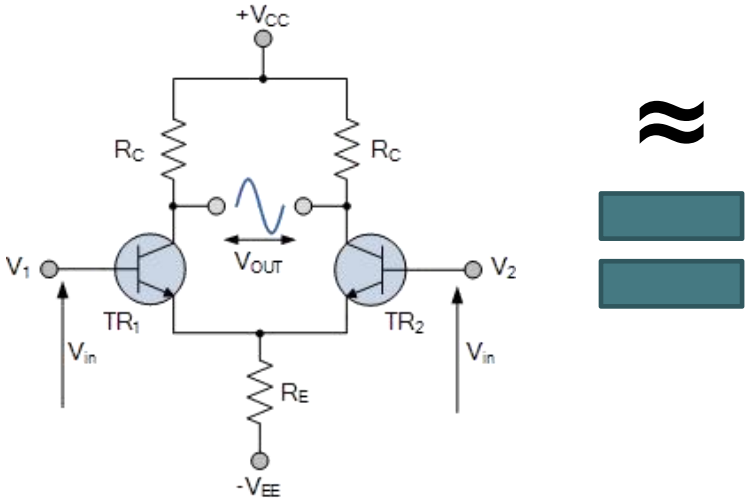
$$V'_{out} = \Gamma(V_+, V_-, I_+, I_-, R_1, R_2, R_L, T_1, ...)$$
$$E(...) = V_{out} - V'_{out}$$



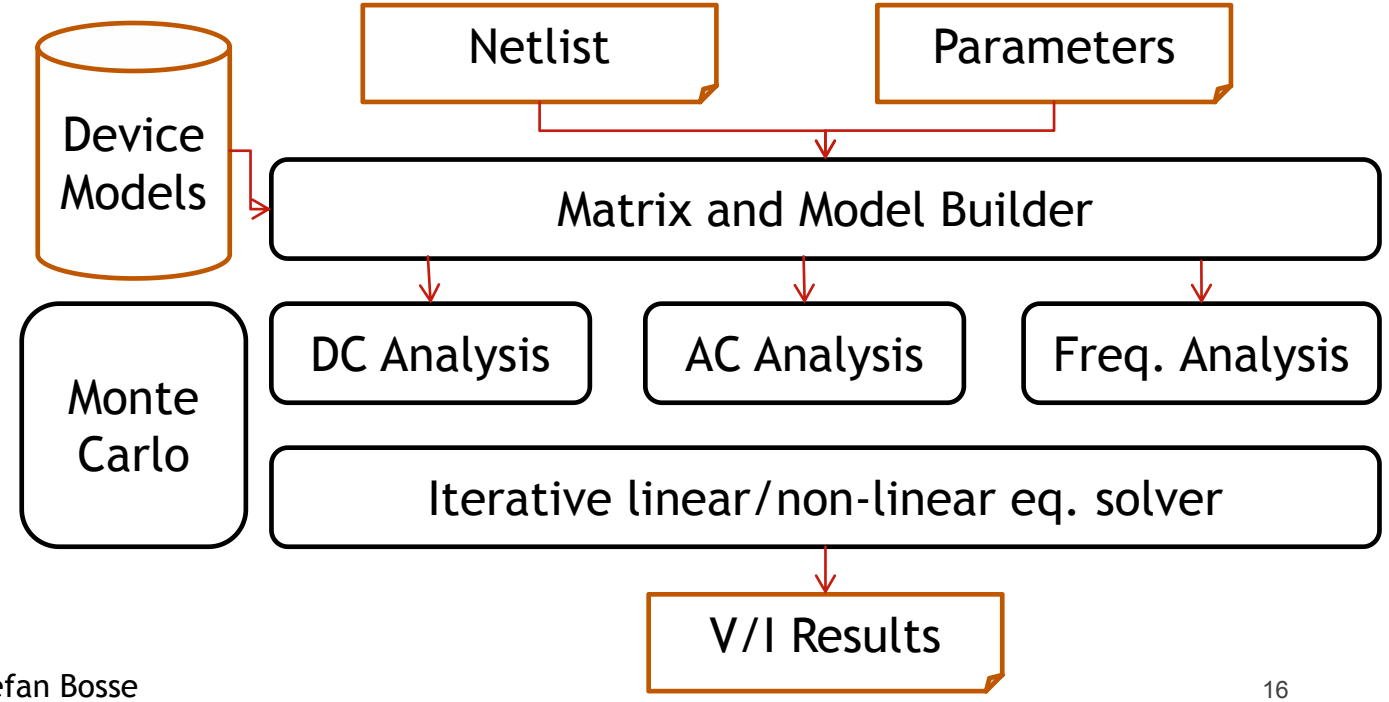


ANALOG COMPUTATIONAL SYSTEMS: OPAMP MODEL

- **Real transistor circuits show significant deviation from the ideal mathematical model!**
- The (static) error *E* is introduced by a function Γ, which depends on a large set of parameters.
- **We need accuracte simulation to design analog circuits from given computational models!**

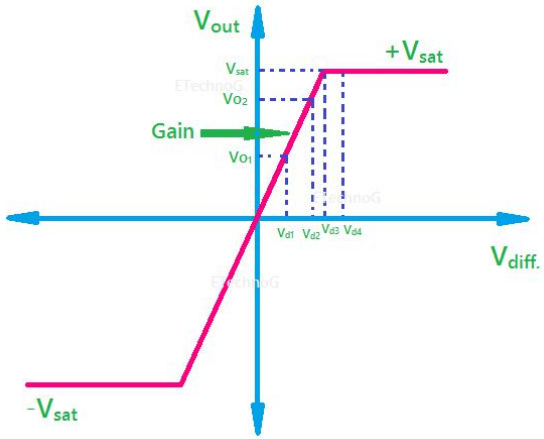


≈



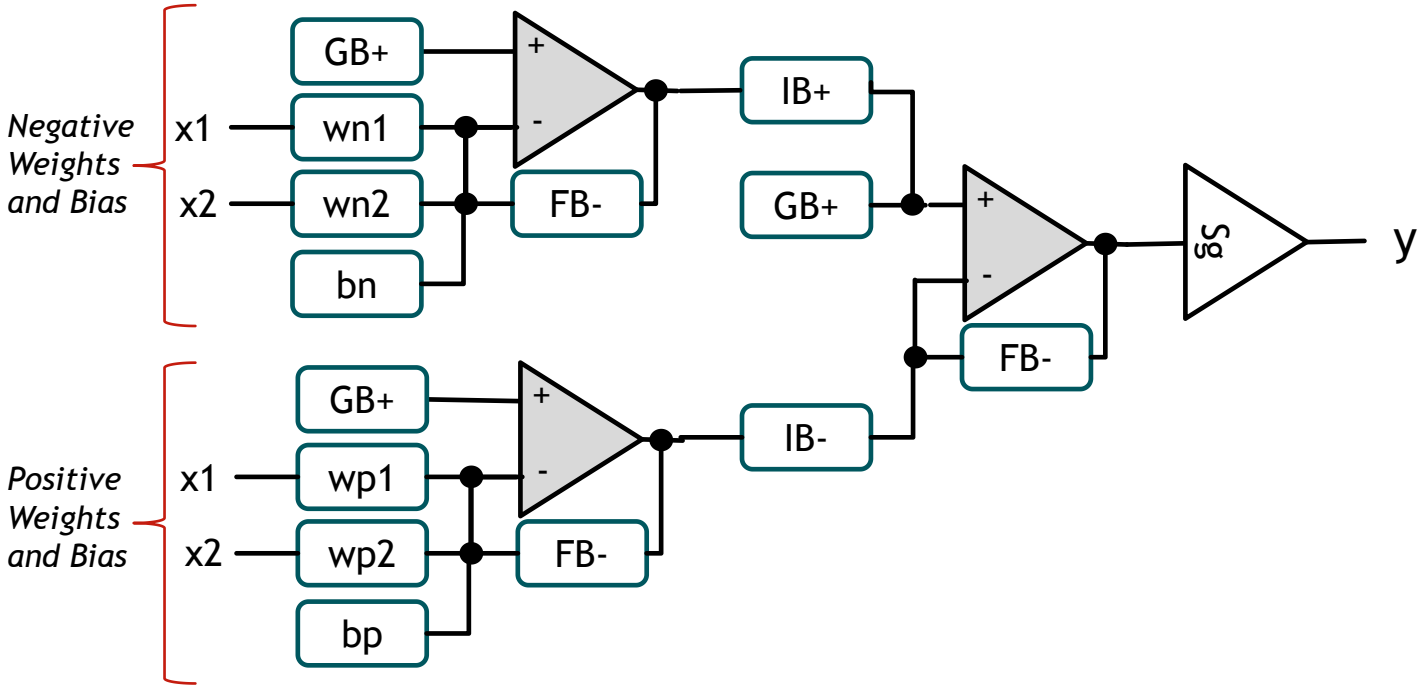
ANALOG COMPUTATIONAL SYSTEMS: OPAMP MODEL

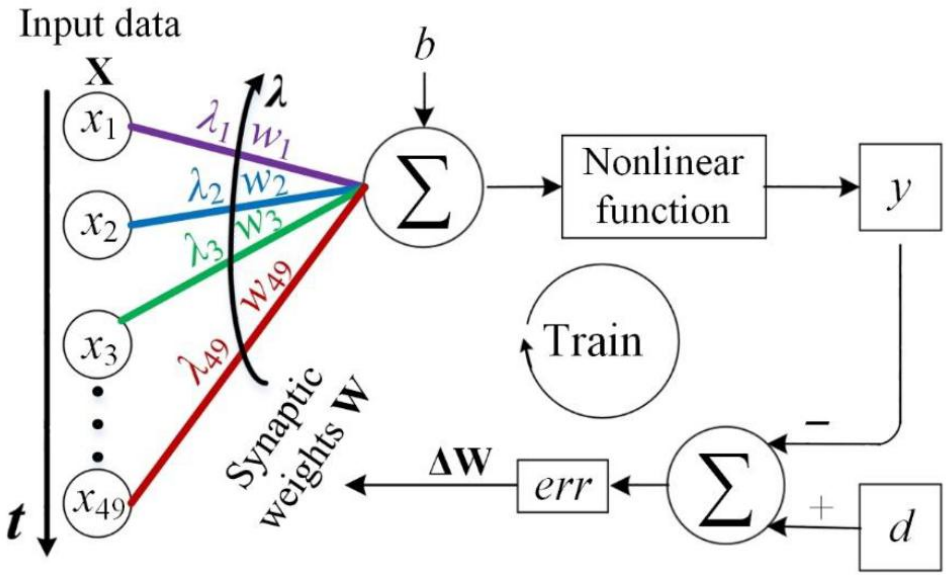
- **Real transistor circuits show significant deviation from the ideal mathematical model!**
- A real OPAMP requires about 10-20 transistors, **a lower transistor count increases the error**:
  - Output offset $V_{out} \neq 0$, although $\Delta V = 0$
  - Drift
  - Limited Gain (Amplification) G
  - Non-linearity
  - Temperature dependency (including spatial gradients)
  - Asymmetric input and output transfer functions
  - Limited output range V_{out} (Saturation, Clipping)



ANALOG COMPUTATIONAL SYSTEMS: ANALOG ANN

- **We implement an analog artificial neuron (AAN) with OPAMP architecture.**
- The neuron is composed of three simple OPAMP blocks and a sigmoid block with a **bipolar path architecture [different paths for negative and positive parameters]**:

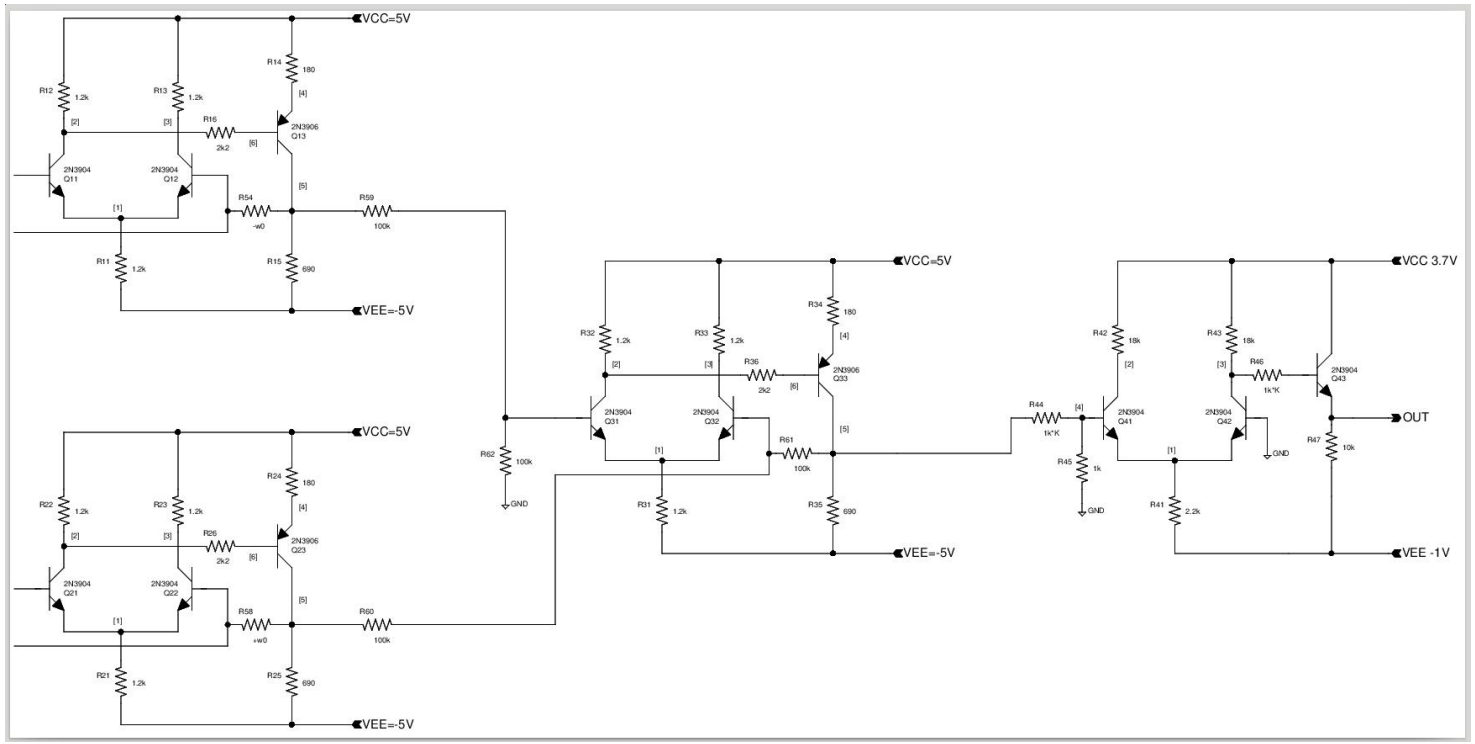


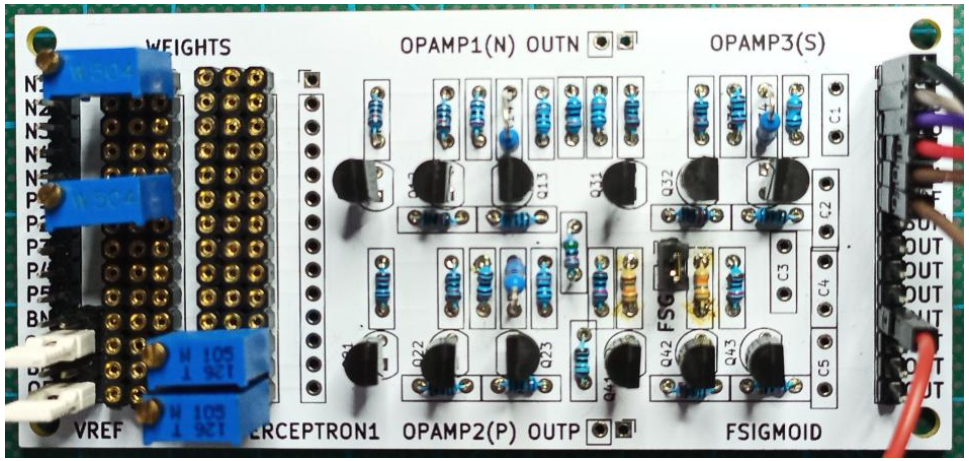


Mathematical Model of a neuron, Xu, 2020

ANALOG COMPUTATIONAL SYSTEMS: ANALOG ANN

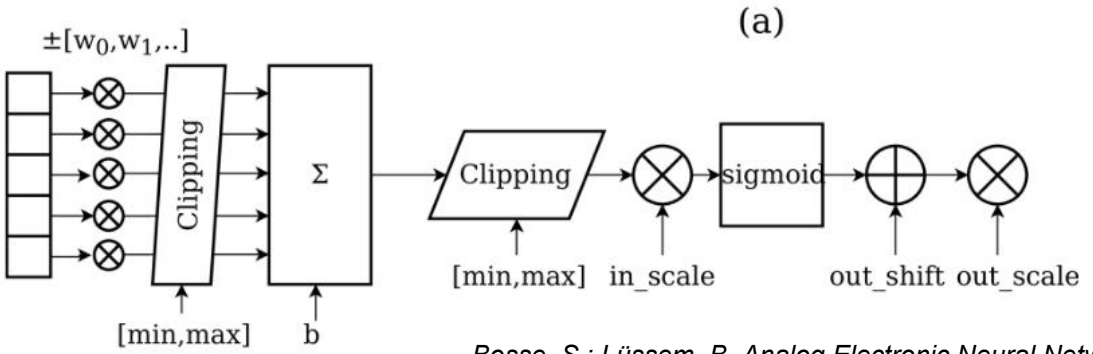
- **We implement an analog artificial neuron (AAN) with a simple transistor circuit.**
- The neuron is composed of three simple OPAMP blocks and a sigmoid block with a bipolar path architecture using 12 bipolar transistors and 27 resistors (excluding weights):

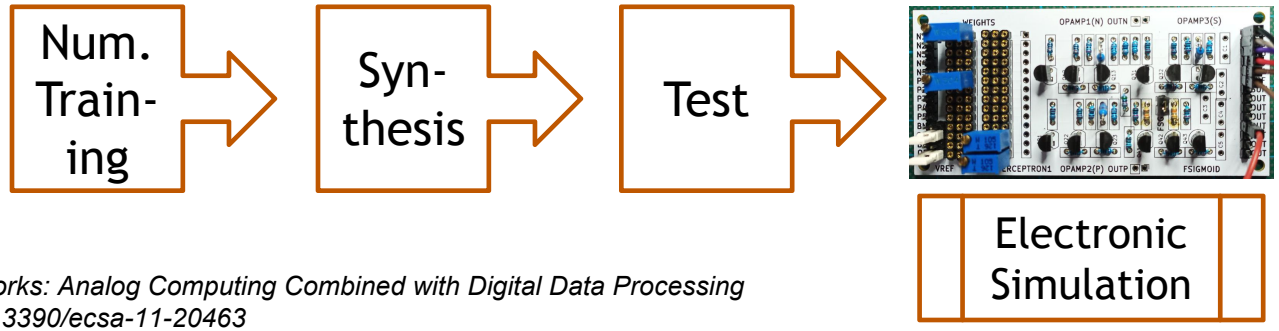




TRAINING OF ANALOG ANN: DIGITAL MODEL APPROACH

- **Starting point: A standard mathemarical/numerical model of a neuron is used consisting of two functions: Linear sum-of-products (SOP) and non-linear activation function (A).**
- Weights are considered as amplification factors for the OPAMP blocks.
- Semi-realistic limitations by **model clipping**: Limited output of SOP blocks (e.g., ±10), limited amplification (e.g. w_{max}=10); input and output scaling; modified sigmoid function
- Classical training algorithms based on **analytical gradients** can be used: sgd/adam/adagrad.





Bosse, S.; Lüssem, B. Analog Electronic Neural Networks: Analog Computing Combined with Digital Data Processing Revisited. Eng. Proc. 2024, 82, 102. https://doi.org/10.3390/ecsa-11-20463

TRAINING OF ANALOG ANN: ANALOG MODEL APPROACH

- **Next step: Training with analog model function f(x,w): x→y (digital twin) of the electronic circuit containing SOP and A functions! SOP and A can be non-linear!**
- **Bipolar architecture**: Separated paths for negative and positive weight/bias parameters trained simultaneously.
- Realistic limitations by **analog model** through:
  1. Real Measurements
  2. Electronic Simulation
  3. **Surrogate Model (SM)** derived from data (Real/Simu), e.g., FC-ANN (tanh activation functions)
- Training: **gradient descent** (gd) with numerically computed gradient of d$f(\boldsymbol{x},\boldsymbol{w})$/d$w_i$

$$w_i = w_i - \alpha e(f,x,y)\frac{\Delta f(x)}{\Delta w_i}, w_i = w_i - \alpha \sum_{j=1}^{batchsize} e(f,x_j,y_j)\frac{\Delta f(x_j)}{\Delta w_i}$$

w: weight parameter, e: backpropagated error, α: update rate

Normally SOP and A are computed separately! Here the entire f(SOP,A) is computed

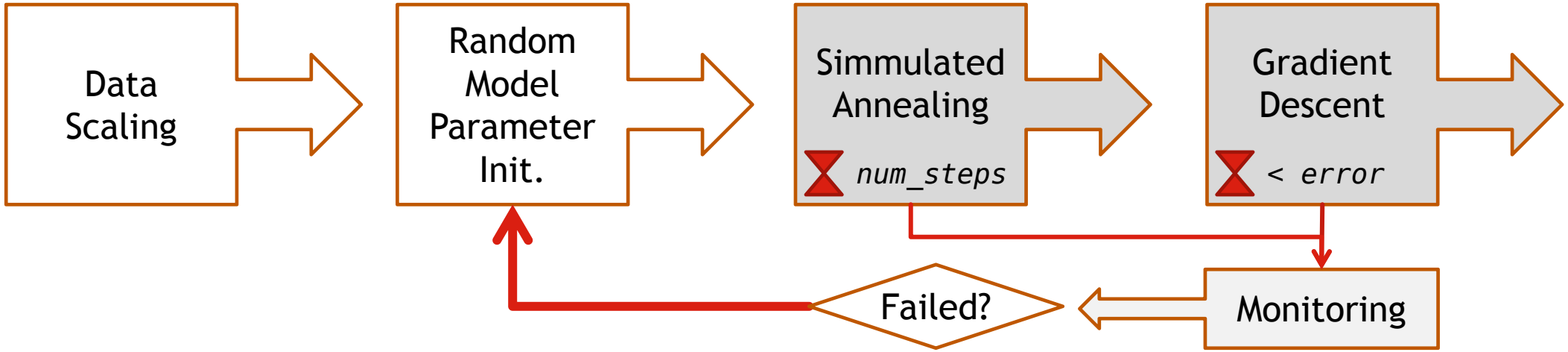
TRAINING OF ANALOG ANN: ANALOG MODEL APPROACH

- **Next step: Pre-Conditioning of parameter space using Simulated Annealing before error gradient backpropagation is performed!**

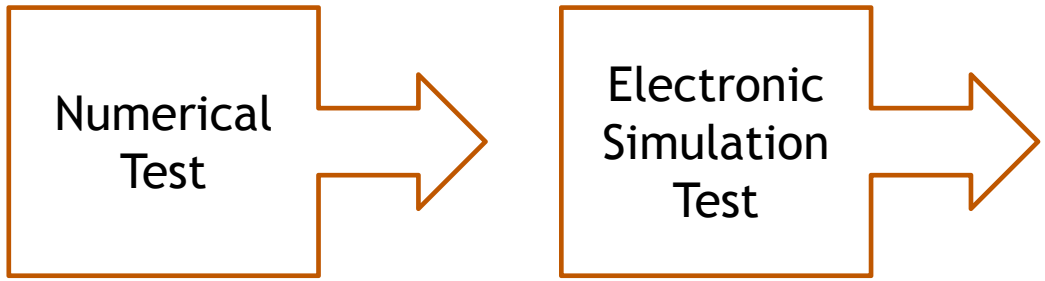
```
function SA(data, params, num_steps=1000, noise=0.01, cooling=0.999) {
  initial_params=optimal_params=best_params=params; temp=1.0
  new_loss = best_loss = loss(data,params)
  for(i=1,num_steps) {
    temp=temp*cooling
    new_params = params.map(p => p+gaussianRandom(0, noise))
    new_loss = loss(data,new_params)
    if (new_loss < best_loss || random()*temp > best_loss/new_loss) {
       params = new_params
       if (new_loss < best_loss) { optimal_params = params; best_loss = new_loss }
    }
  }
}
```

TRAINING OF ANALOG ANN: ANALOG MODEL APPROACH

1. Workflow: Training

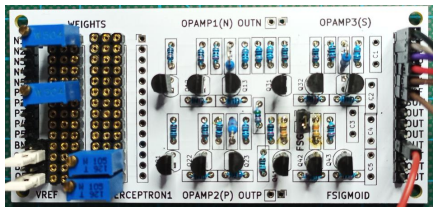


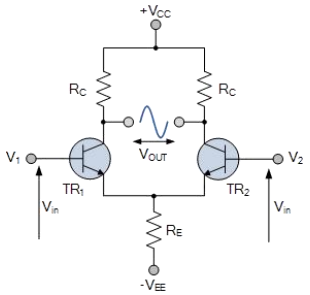
2. Workflow: Test

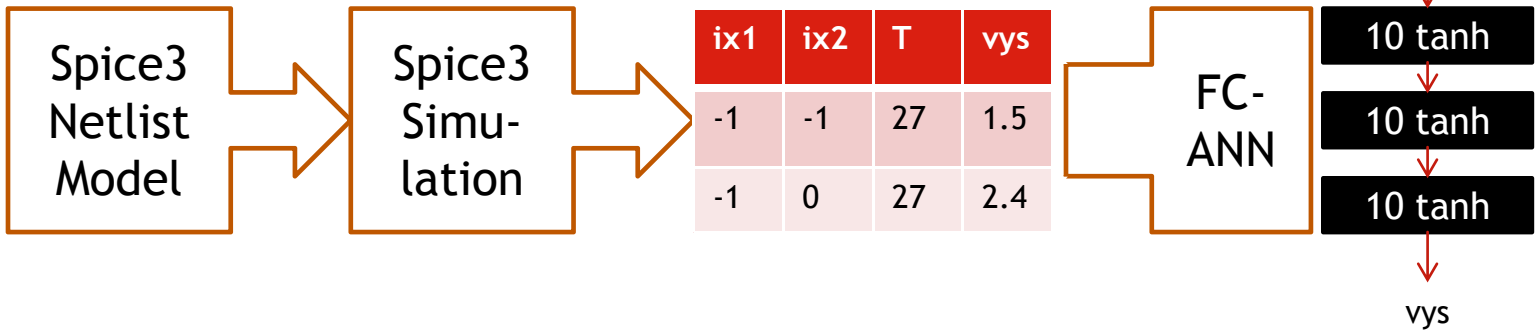


TRAINING OF ANALOG ANN: SURROGATE MODEL APPROACH

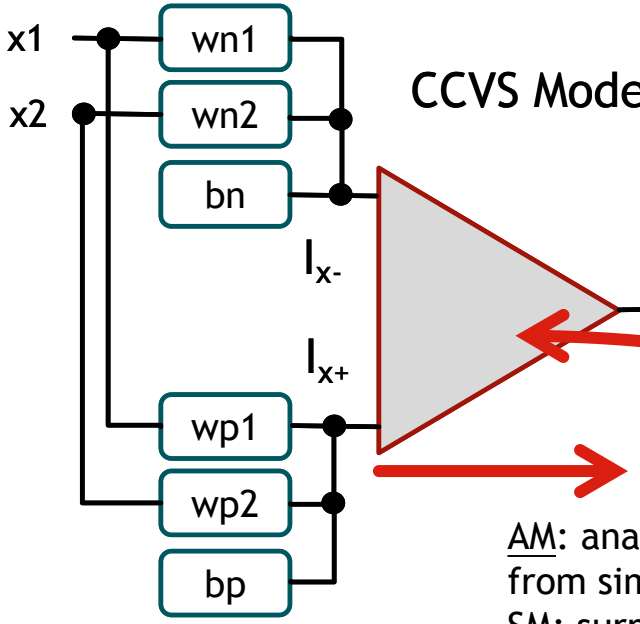
- **The Analog Electronic Model (AM) is one monolithic parameterized function f(x,w): x → y describing input to output relation used for forward and backward computations of the AANN.**
- The mapping **x → y** can be derived by:
  1. Measurement in real parametrizbale electronic circuits (hardware-in-the-loop): Slow! Too slow;
  2. By electronic simulation: Not so slow but still too slow;
  3. **By a data-driven surrogate model SM/S-Model (e.g., a neural network, too): Faster, let‘s try it!**
  4. By an analytical function: Fastest, but impossible (or too simplified).







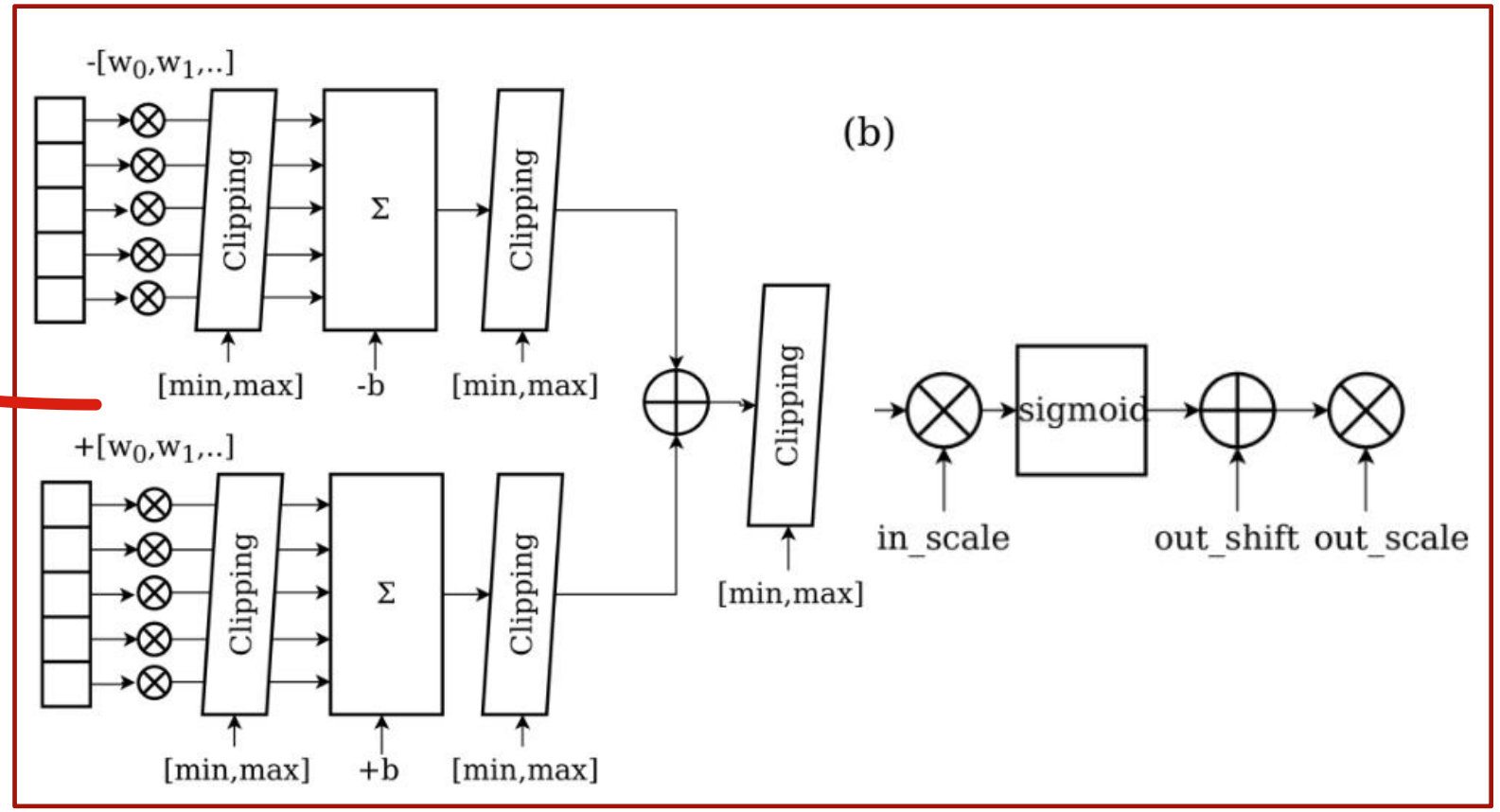
TRAINING OF ANALOG ANN: ANALOG MODEL APPROACH



$$R_i = R_0 / w_i$$

$$I_x = -\sum_{i=1}^{n} \frac{x_i}{R_i}$$

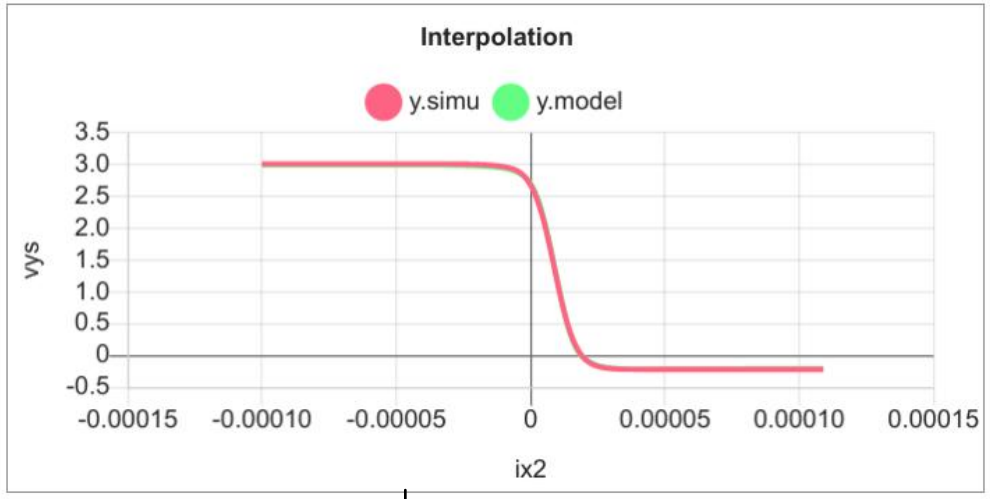
AM: analog model from simulation
SM: surrogate model, f(AM): x→y
CCVS: Current-controlled Voltage Source



Monolithic f(SOP,A,w,b)

TRANSFER FUNCTION S-MODEL VS. CIRCUIT

- **S-Model analysis within interpolation range: Promising results ... we are done?**

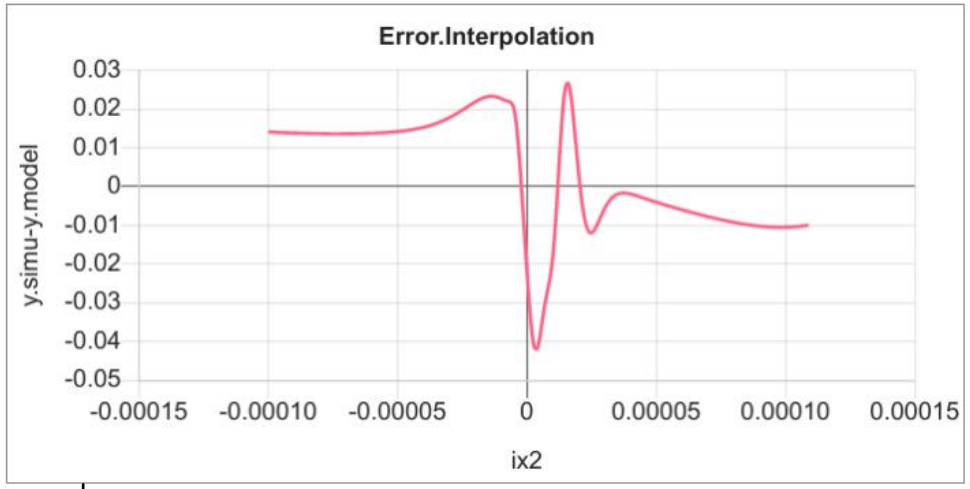


Circuit

- Output voltage versa input current (ix1=fixed)
- Monotonic behaviour

S-Model

- First overview: Monotonic behavior
- Interpolation: Input values within training range

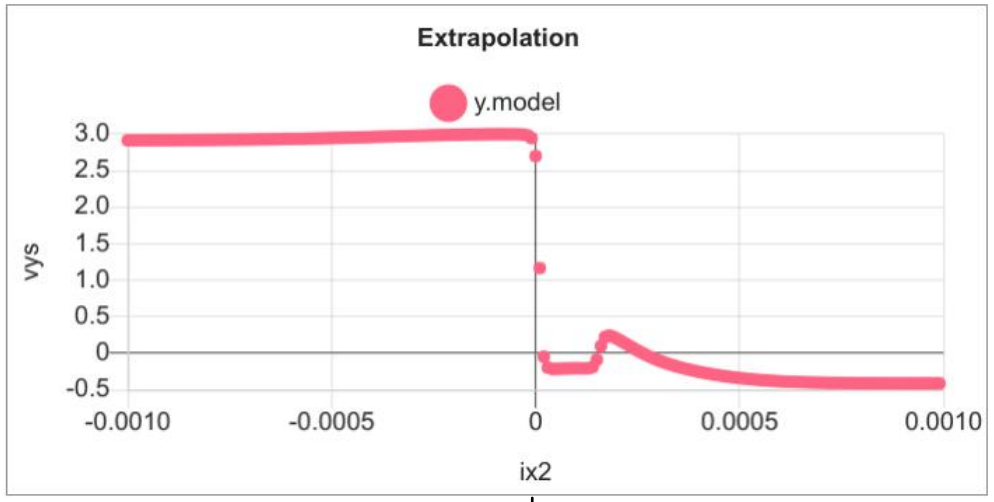


S-Model

- Low prediction error compared with circuit data, but error oscillates in the steep region

TRANSFER FUNCTION S-MODEL VS. CIRCUIT

- **What is wrong here? The reality gap of data-driven models ... never extrapolate**



Circuit

- Output voltage versa input current (ix1=fixed)
- Monotonic behavior

S-Model

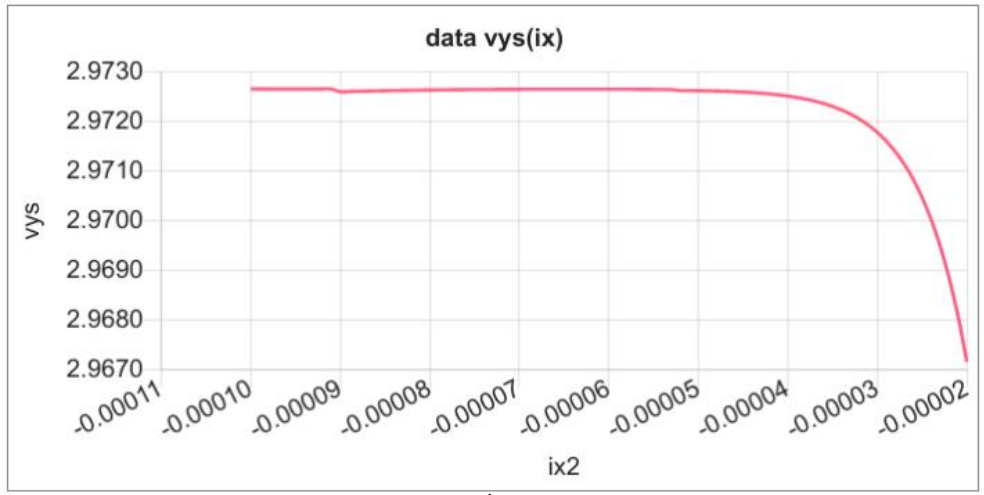
- Non-Monotinic behavior!
- Extrapolation: Input values outside training range

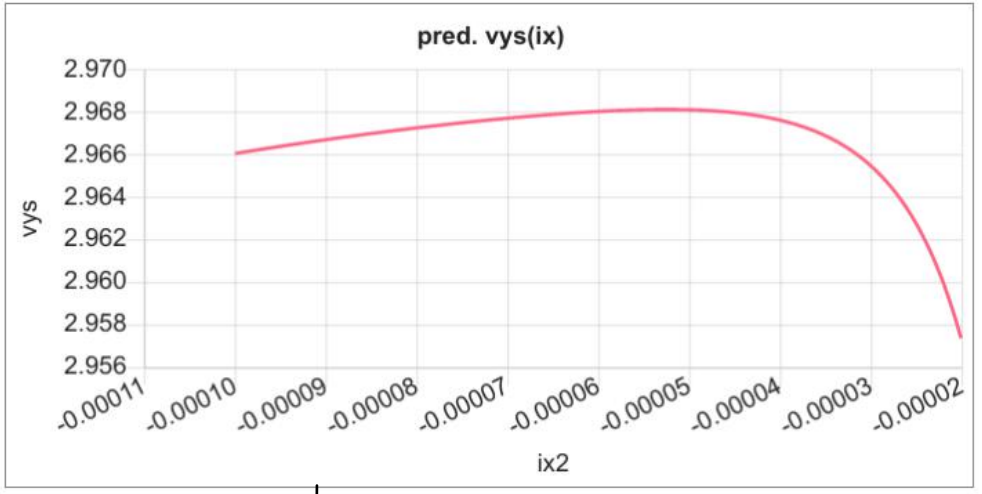
S-Model

- High prediction error compared with circuit data outside the training space in the right branch
- Left branch seems still valid, really?

TRANSFER FUNCTION S-MODEL VS. CIRCUIT

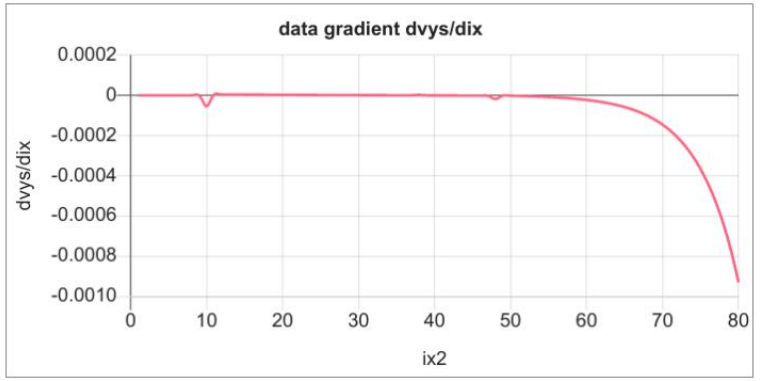
- **What is wrong here? The reality gap of data-driven models ... the details!**





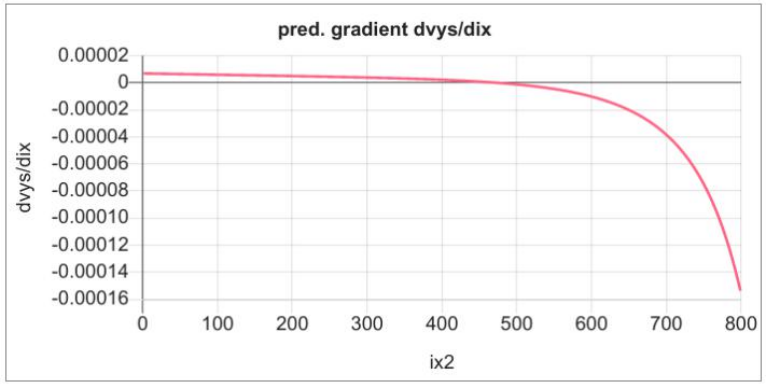
Circuit

- Output voltage versa input current (ix1=fixed)
- Left branch
- Falling monotonic behavior
- But gradient shows „notches“

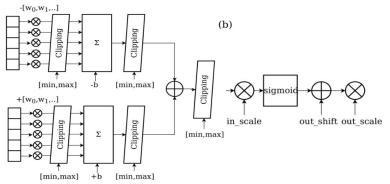


S-Model

- Non-monotonic behavior
- Zero crossing of gradient
- **Non-convex optimization problem**



EXPERIMENTS AND RESULTS



- **Bipolar clipped mathematical model (baseline model), Gradient Desc. (+ Simulated Annealing)**

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Logic OR Gate

- [2,1] neurons
- Convergence: < 20 Epochs, 100% success rate
- Class. Error: 0%
- Bipolar separated parameters: yes
- Linear problem

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Logic EXOR Gate

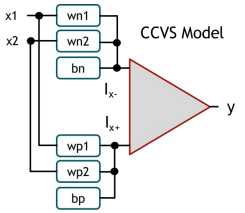
- [2,1] neurons
- Convergence: < 50 Epochs, > 70% success rate
- Class. Error: 0%
- Bipolar separated parameters: yes
- Non-linear problem

w	l	pw	pl	Cls
3	3	2	1.5	Set
4	2	1	2.4	Ver
3	4	2	2	Sac
2	3	1	2	Ver

IRIS Benchmark

- [3,3] neurons
- Convergence: < 200 Epochs, > 90% success rate
- Class. Error: 2-4%
- Bipolar separated parameters: yes
- Linear and non-linear problem
- SA-only training: success
- 1 µs forward, 2 µs backward

EXPERIMENTS AND RESULTS



- **Bipolar analog el. surrogate model (from simulated data), Gradient Desc.+ Simulated Annealing**

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Logic OR Gate

- [2,1] neurons
- Convergence: < 50 Epochs, > 90% success rate
- Class. Error: 0%
- Linear problem

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Logic EXOR Gate

- [2,1] neurons
- Convergence: < 100 Epochs, > 50% success rate
- Class. Error: 0%
- Non-linear problem

w	l	pw	pl	Cls
3	3	2	1.5	Set
4	2	1	2.4	Ver
3	4	2	2	Sac
2	3	1	2	Ver

IRIS Benchmark

- [3,3] neurons
- Convergence: > 30 < 500 Epochs, 20% success rate (w/o SA: < 10%)
- Class. Error: 2-4%
- Bipolar separated parameters: no
- Linear and non-linear problem
- SA-only training: failed
- 100 μs forward, 800 μs backward

CONCLUSIONS

Analog ANN

- Bipolar Difference OP Amplifier Architecture
- Transistor circuit: 12 Transistors
- SOP with non-linearity, Offset
- SOP with limited weights (<40)
- Training: Indirect with Bipolar clipped numerical model with post-synthesis or direct analog electronic surrogate model

Surrogate Model

- Derived from electronic simulation of OPAMP circuit
- Current Controlled Voltage Source Model. Input: Positive and negative SOP currents, Output: Voltage of Sigmoid approximation circuit
- Low approx. error, but deviation at the boundaries
- High error beyond trained parameter space

ML Training and Results

- Training with Error Gradient Descent Backpropagation
- Parameter space initializiation with random process
- Parameter space pre-conditioning with Simmulated Annealing
- Low convergence, trainining instability, but low class. error
- Iterative monitored training process with fallback on low progress

THANK YOU

Stefan Bosse

sbosse@uni-koblenz.de

www.edu-9.de

university of koblenz
Computer Science

Praktische Informatik