Towards Ultra-Low Power Pulse Based Signal Processing

Jose C. Principe

Distinguished Professor

University of Florida

principe@cnel.ufl.edu

www.cnel.ufl.edu

Euro Micro 2015

Acknowledgements

Joint work with Drs. John Harris and Nima Maghari

Students: Gabriel Nallathambi
Kan Li
Alex Singh
Manu Rastogi
Christy Rogers
Du Chen
Das Wei

Supported by NIH and NSF

Digital Computing

- Church-Turing (1936) developed the theory of formal systems and propositional calculus that enabled computation
- Von Neumann (1945) proposed an architecture to implement it
- Number representations are supported by logic. This created our current model of computation, **which is digital.**
- We should appreciate the value of these theoretical and engineering achievements, and look at this systematic approach as an example to follow.
- Digital computing is ubiquitous and currently we do not have yet competitors (quantum computing).

From Real World Signals to Numbers: Sampling Theory

Definition: Sampling is a one to one mapping between algebraic spaces with a unique inverse (isomorphism)

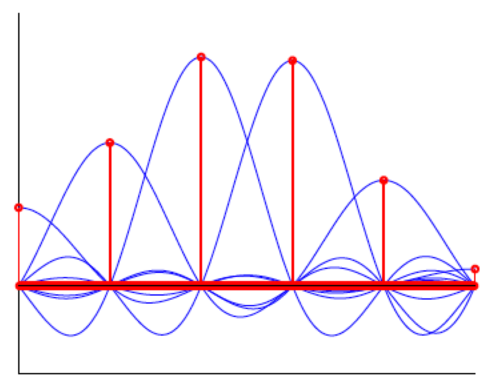
$$\boxed{\begin{array}{c}\text{Input Space}\\ \\ \text{Time Functions}\end{array}} \longrightarrow \boxed{\begin{array}{c}\text{Output Space}\\ \\ \text{Bit Streams}\end{array}} \longrightarrow \boxed{\begin{array}{c}\text{Reconstruction}\\ \text{to Input Space}\end{array}}$$

Band-limited	Point evaluation	Linear spaces
Shift-invariant	Local average	Frame methods
Sparse	Random projection	L1 minimization

Theorem

Whittaker-Kotel'nikov-Shannon
Every band limited signal $f(t)$, can be reconstructed from its uniform samples taken at the Nyquist rate:

$$f(t) = \sum_{k=-\infty}^{\infty} f\left(\frac{k\pi}{\Omega}\right) \frac{\sin(\Omega t - k\pi)}{\Omega t - k\pi}$$



Moore's Law and Computational Frameworks

• The demise of Moore's Law (1965/75) is pushing us to imagine computation beyond the digital model.

• With these two bottlenecks (theoretical and technological), perhaps it is a good time to think out of the box!

• I submit that **time based computation** has really good attributes (engineers master time measurements). But it is not fully explored nor theorized.

• Let us briefly review alternative computational frameworks.

Stochastic Computing

- Gaines (1967) proposed computing with probabilities instead of numbers (Von Neumann's stochastic logic).
- A quantity is represented by a clocked sequence of logic levels generated by a random process (Bernoulli sequence).
- He proposed 3 linear mappings of analog variables to digital variables probabilities (i.e. 0 for zero quantity, 1 for maximum range, and the probability in between).
- Finite state machines could operate with these probabilities to do computation.
- One of the problems is poor scaling with precision. Still interesting today because of low power.

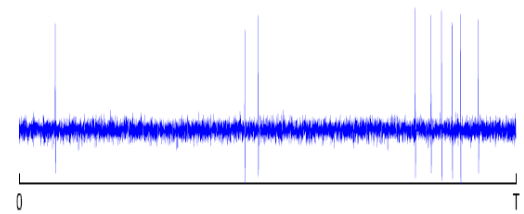
A. ALAGHI, J. HAYES, Survey of Stochastic Computing, ACM TECS, 2012

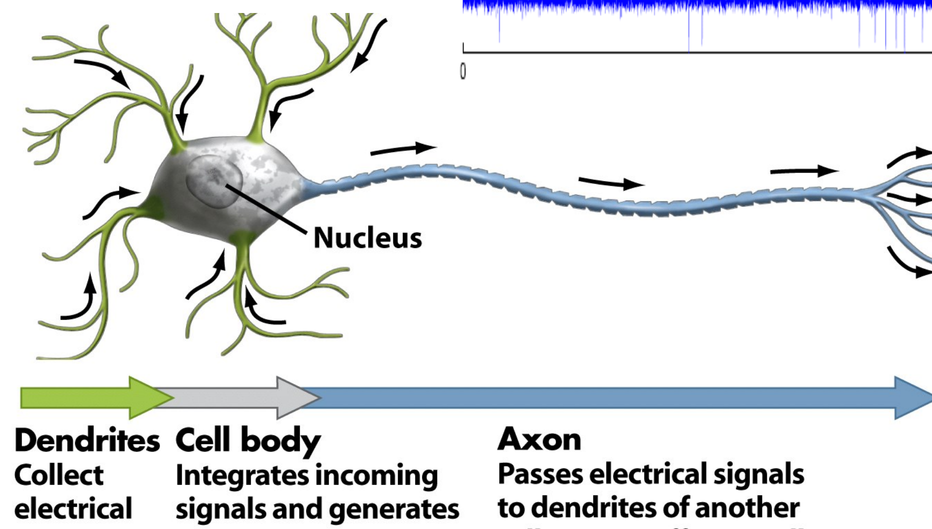
Non Numeric Computing: Analog

- The first man made computers were analog.
- Since the real world is analog it does not require conversion.
- Analog computation is fast (speed of electrons) and very appropriate to model dynamical systems <u>because it uses time to do computation.</u>
- Analog computing is also low power with current technologies.
- But it is plagued by fabrication variations, drift, noise, and non repeatability.
- All this limits analog computing scalability and analog computation is not general purpose at this point.

Non Numeric Computing: Human Brain

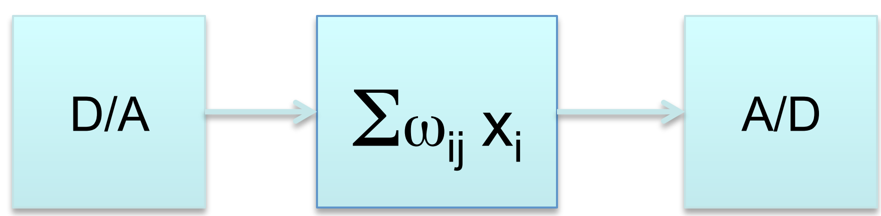
The brain is a spatio-temporal dynamical system, i.e. computation is done in time.





1. **Representation**
Neural spike trains are nonlinear encodings of vector space variables.
2. **Computation/Transformation**
Linear decoding of spike encodings can compute arbitrary vector functions.
3. **Dynamics**
Neural representations are control theoretic state variables in a nonlinear dynamical system

Eliasmith & Anderson, 2003



Neuromorphic Computing: Silicon Based

- W. Maass, 1999 showed that computation with pulses using Leaky Integrate and Fire (LIF) neural models is universal.
- It is an electronic implementation of brain like computation with the known rules of neural function.
- It is asynchronous, requires integration (or sum of products) so it is not fully implemented with time operators
- More recently many different neuromorphic architectures are following this approach (IBM TrueNorth, HRL SyNAPSE)

Neuromorphic Computing: Silicon Based

- The brain is a crowded noisy environment, and nature invented the spikes because spikes
  - Handle noise well
  - Use as little power as possible
- Spikes are great because we have exquisite precision in time measurements
- Neurons are operators on spikes, which still require analog processing or numeric computation with the current digital techniques.
- **Can we handle the neural operations differently?**

Time –Based Computation

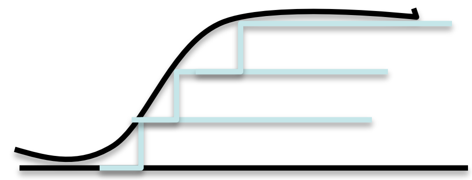
- Depart from the brain metaphor

  Do all computation with time domain operators

- Challenges
  - How to transform signals into pulses (samplers)
  - How to compute with pulses in time
  - Preserve Von Neumann programmability

Available Time Samplers

Asynchronous Delta-Sigma Modulators



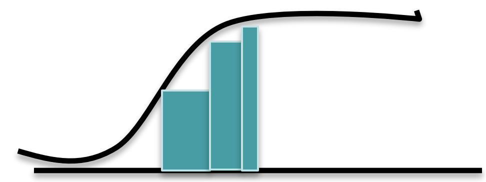
Input space → Output Space → Reconstruction

Band-limited — Inequality constraint — Low pass filter

Drawback: Oversampling, since information is encoded in the event rate

Integrate and Fire Sampler

A special case of an ASDM, but it operates at sub-Nyquist rates



Input space → Output Space → Reconstruction

Band-limited Shift-Invariant

Events at Integral level crossing

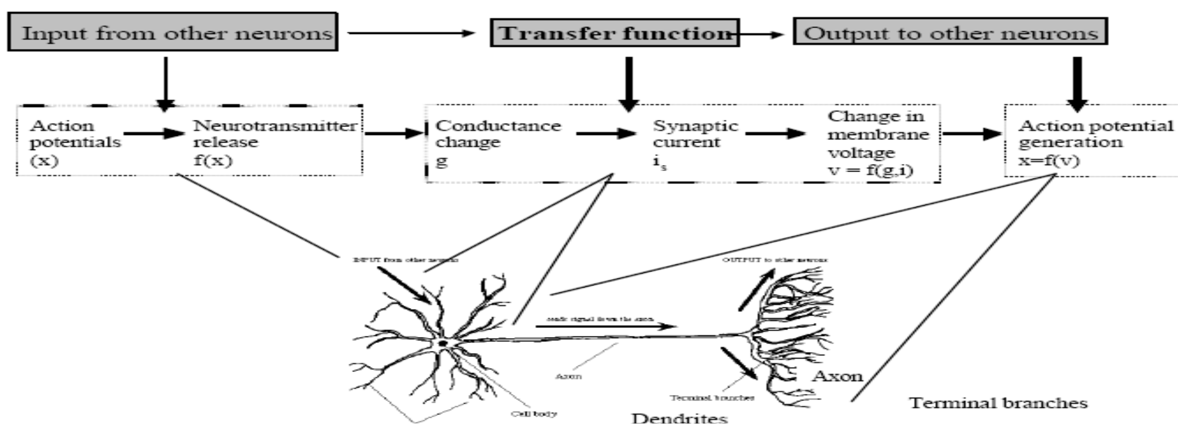
linear adaptive filters

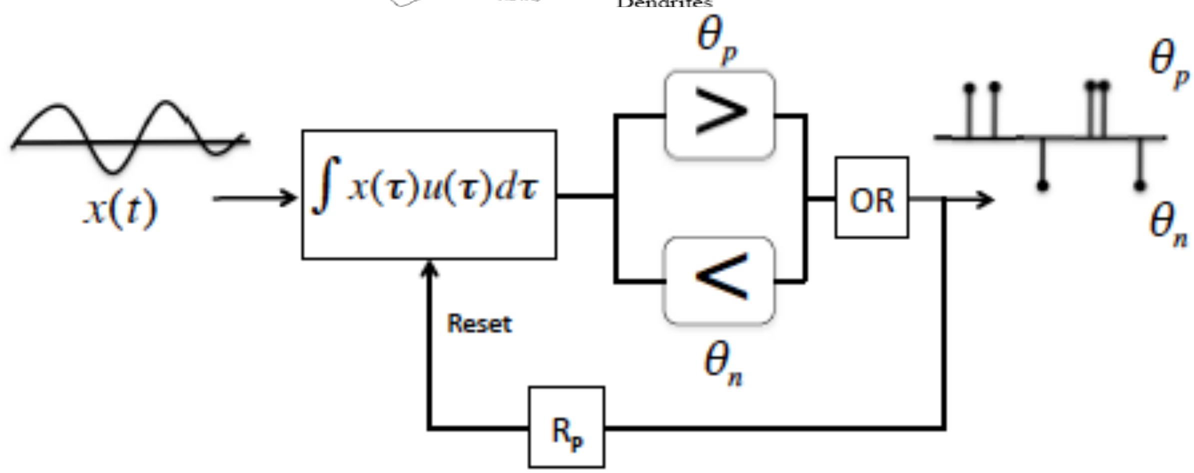
IFS approximates the input signal by the area under the curve (rectangles of fixed area). Information is encoded in the **precise timing** of the events.

Integrate and Fire Sampler

Inspired by how neurons work:

When the action potentials arrive at the synaptic input of a neuron, the potential field in the dendritic tree slowly rise until the neuron fires an action potential.

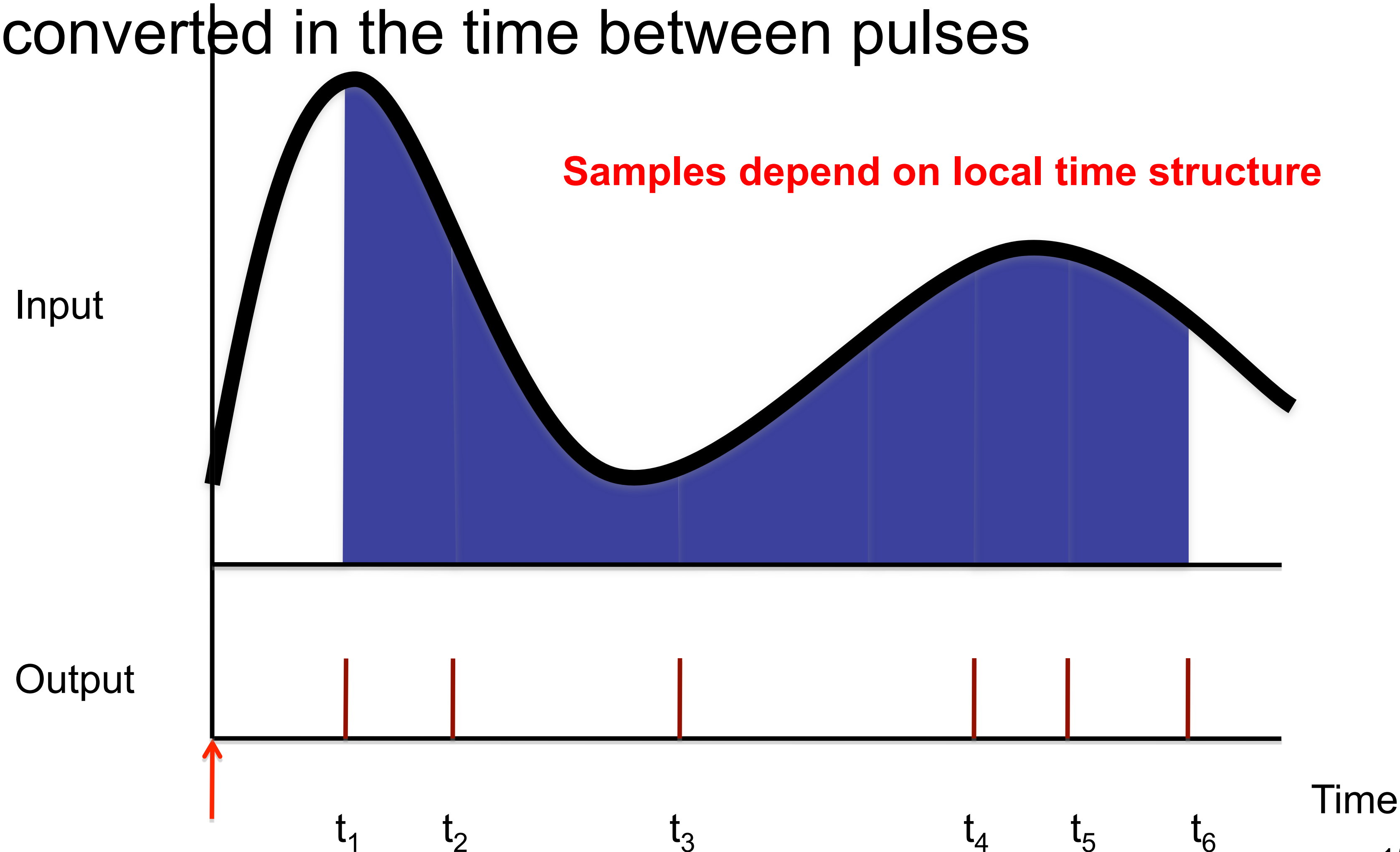




Amplifier with pulse coded output, (with Harris, Chen, and Wei), US Patent # 7324035, 2008.

Integrate and Fire Sampler

With a fixed size area constraint, amplitude is converted in the time between pulses



Integrate and Fire Theory

We introduce an auxiliary function, the membrane potential v(t)

$$f(t) = \frac{\partial v(t)}{\partial t} + \alpha v(t)$$

Membrane potential

$$v(t) := \int_{-\infty}^{t} f(x) e^{\alpha(x-t)} dx$$

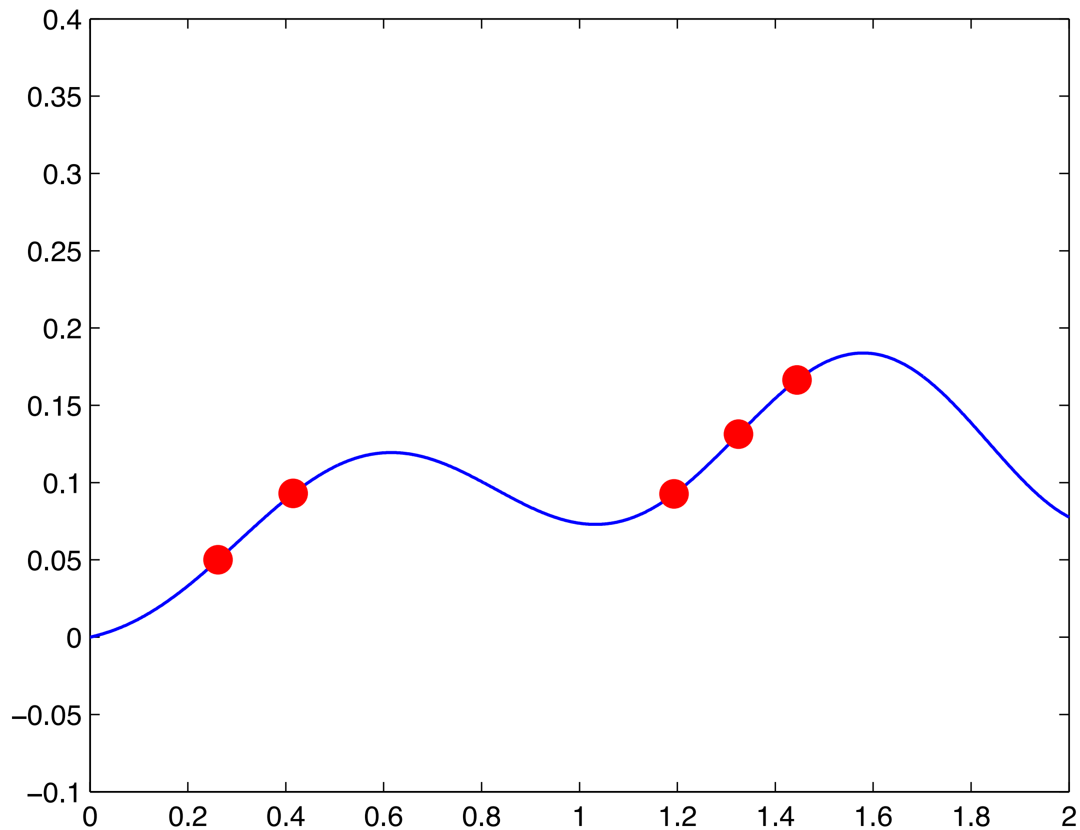
Condition to fire

$$\left| v(t_{j+1}) - e^{\alpha(t_j - t_{j+1})} v(t_j) \right| = \theta$$

Feichtinger H., Principe J., Romero L., Singh A.,Velasco G., “Approximate reconstruction of bandlimited functions for the integrate and fire sampler”, Advances in Computational Math, Volume 36 Issue 1, pp 67-78, 2012

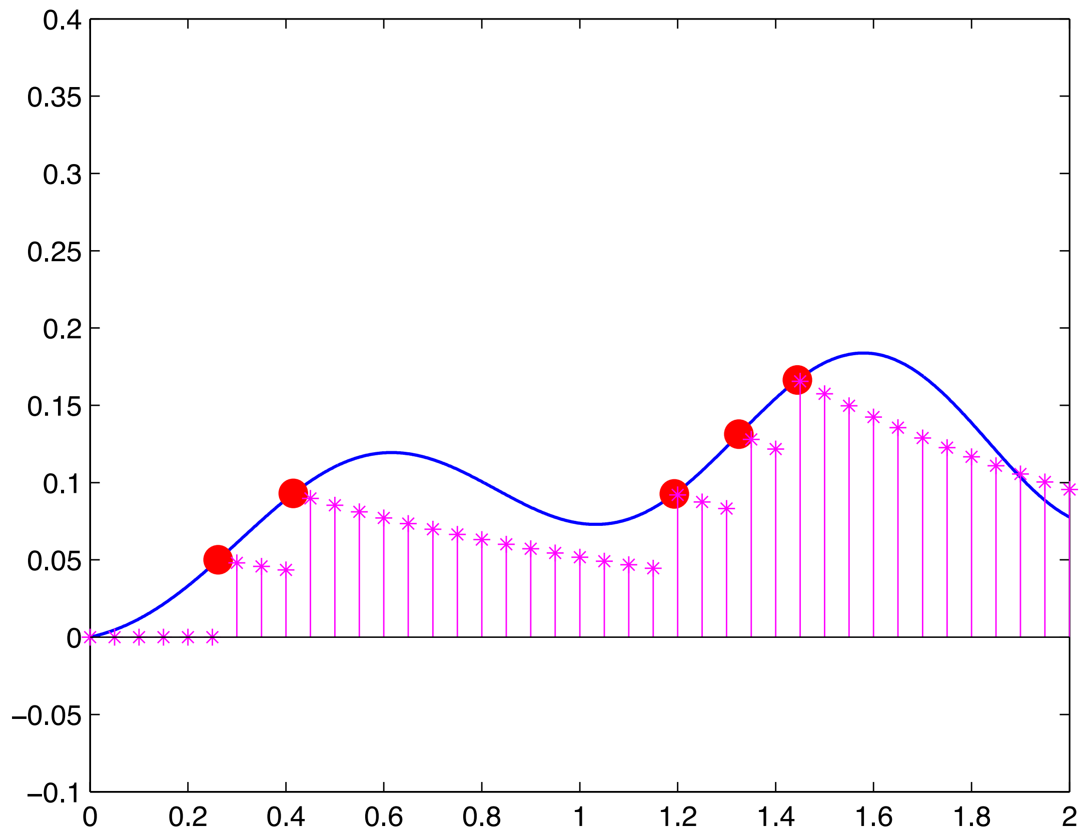
$$\left| v(t) - e^{\alpha(t_j - t)} v(t_j) \right| \leq \theta, \quad \text{for all } t \in [t_j, t_{j+1}]$$

$v(t)$



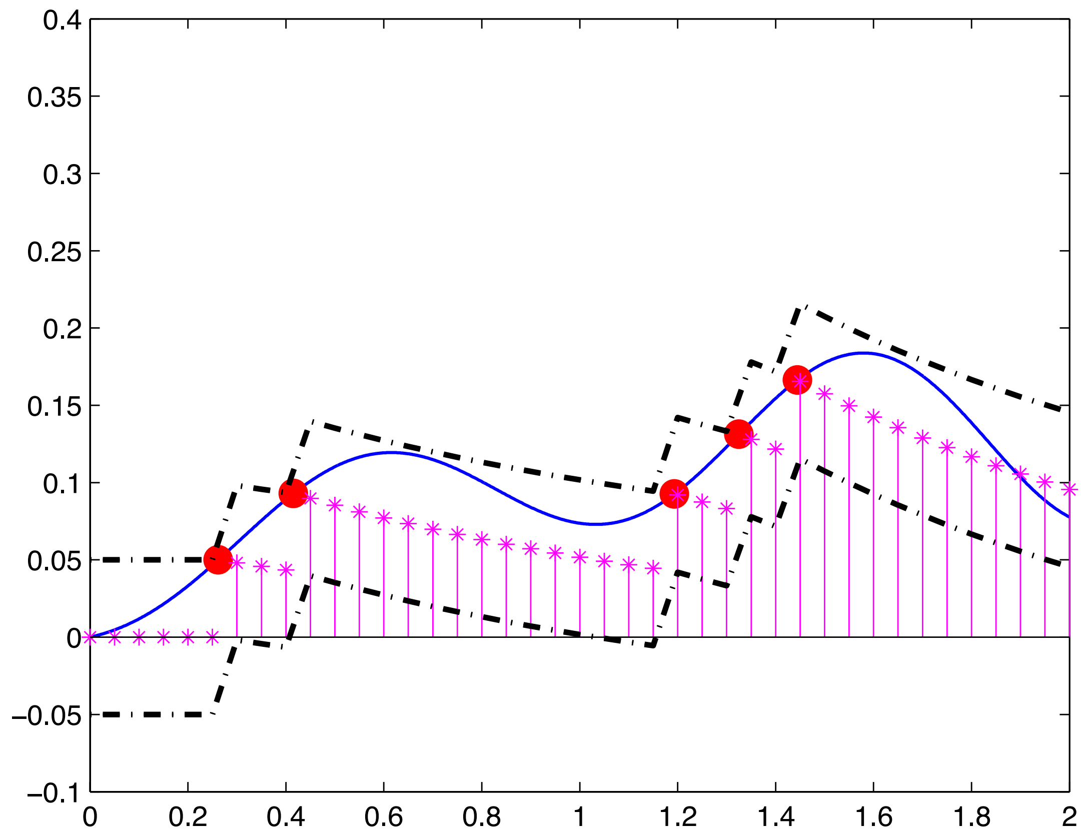
$$\left| v(t) - e^{\alpha(t_j - t)} v(t_j) \right| \leq \theta, \quad \text{for all } t \in [t_j, t_{j+1}]$$

$v(t)$

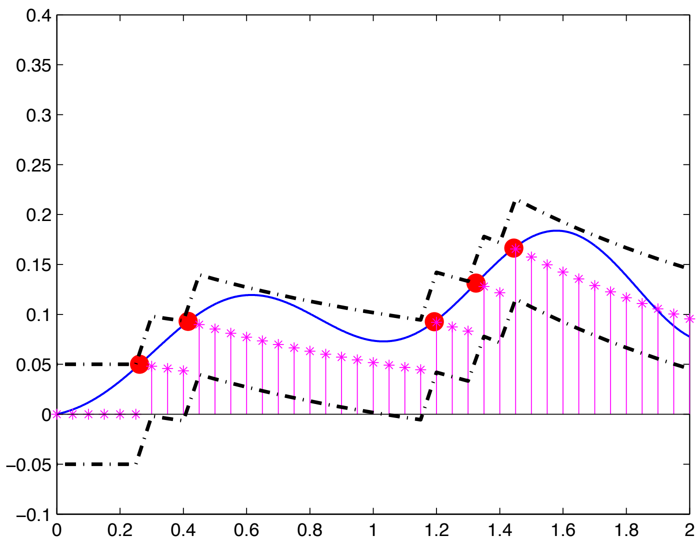


$$\left| v(t) - e^{\alpha(t_j - t)} v(t_j) \right| \leq \theta, \quad \text{for all } t \in [t_j, t_{j+1}]$$

$v(t)$



Integrate and Fire Theory



$$\tilde{v} = \sum_{k \in \mathbb{Z}} s_k \psi(\cdot - \beta k)$$

$$\tilde{f} = \sum_{k \in \mathbb{Z}} s_k \phi(\cdot - \beta k)$$

$$\|f - \tilde{f}\|_\infty \leq C\theta$$

For band limited signals we can bound the reconstruction error based only on the threshold, which also controls the accuracy

Integrate and Fire: Reconstruction

$$f(t) = \sum_{k=1}^{M} a_k \phi_k(t)$$

$$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_N \end{bmatrix} = \begin{bmatrix} \int_{t_1+\tau}^{t_2} \phi_1(\alpha)d\alpha & \dots & \int_{t_1+\tau}^{t_2} \phi_M(\alpha)d\alpha \\ \vdots & \ddots & \vdots \\ \int_{t_{N-1}+\tau}^{t_N} \phi_1(\alpha)d\alpha & \dots & \int_{t_{N-1}+\tau}^{t_N} \phi_M(\alpha)d\alpha \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_M \end{bmatrix}$$

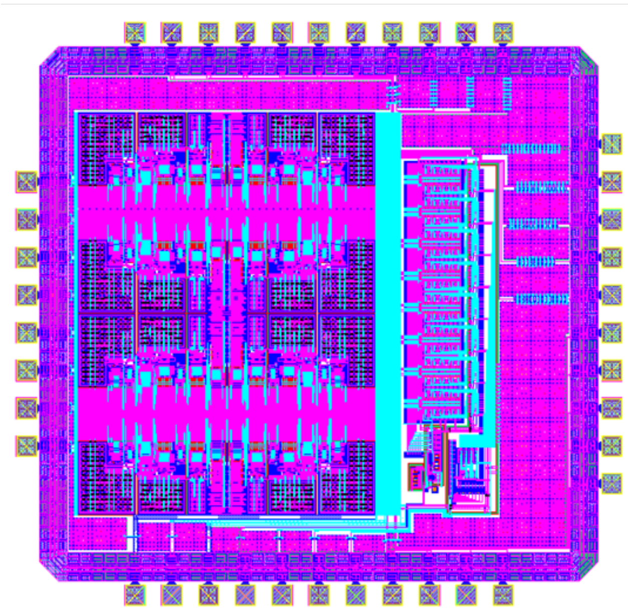
$$\vec{\theta} = S\vec{a}$$

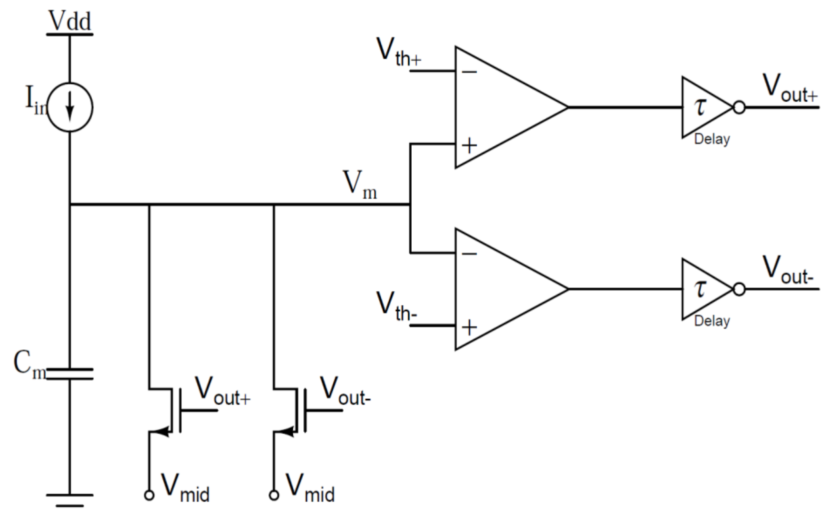
Reconstruction can be implemented in batch (blocks of data) or on-line using recursive least squares type of algorithms.
Basis Function: Splines or Fourier

How to Think About the IF Sampler in Practice

- IF sampler is different from ADCs because the number of pulses are **unequally distributed across the signal** (true time processing).
- This enables **sub-sampling** rates while preserving high reconstruction accuracy in high amplitude portions of the signal.
- Therefore it behaves like **compressive sampling** without imposing the constraint of sparseness.
- **There is no randomness intrinsic in this operation!**

Hardware Implementation





Biphasic IF circuit

8 channel IF chip for Neural implant
4.0 mm X 4.0 mm in CMOS 0.5 um tech.
Chip includes :

1. 8 Bio-Amplifers
2. 8 Voltage-to-current converters
3. 8 Bi-phasic IF
4. Telemetry DACs
5. Asynchronous Readout Circuit

Single channel IF has ~ 30 transistors.
With a layout box of 100 um X 100 um
In CMOS 0.5 um tech.

FOM (pJ/conv)= 0.6 in 0.6 μm

[1]M. Rastogi, V. Garg, and J.G. Harris, “Low power integrate and fire circuit for data conversion,” *2009 IEEE International Symposium on Circuits and Systems*, IEEE, 2009, pp. 2669-2672.

How to Compute with Pulses

Wish list:

- Avoid binary, synchronous machines!
- Avoid analog integration!

- Information is contained in the timing and sequence of pulses, so need to capture this structure by time operators
- Need to realize that all signals from the world are noisy.

How to Compute with Pulses

Two possible methodologies

- Syntactic Pattern Matching (automata)
  - Extract structure of the pulse trains using definitions (when available) or automatically from data (machine learning)
- Arithmetic
  - Define a Field on the space of pulse trains (time functions), instead of real or complex numbers.

Syntactic Pattern Matching

- The extraction of information should be done on the EXACT time structure of the pulse trains.
- The bipolar pulses generated by IFS have positive or negative polarity and hence have digital amplitude (-1/+1).
- Since IFS pulse trains are digital sequences, we can apply the theory of deterministic finite automata and formal grammars augmented with duration constraints.

Hopcroft, Ullman, Introduction to Automata Theory, Languages, and Computation, 1979

DFAs and Attribute Grammars

- A deterministic finite automaton (DFA) is a 5-tuple, consisting of a finite set of *states*, a finite set of input symbols called the *alphabet*, a *transition function*, a *start state*, and a set of *accept states* .
- An attribute grammar is a four-tuple

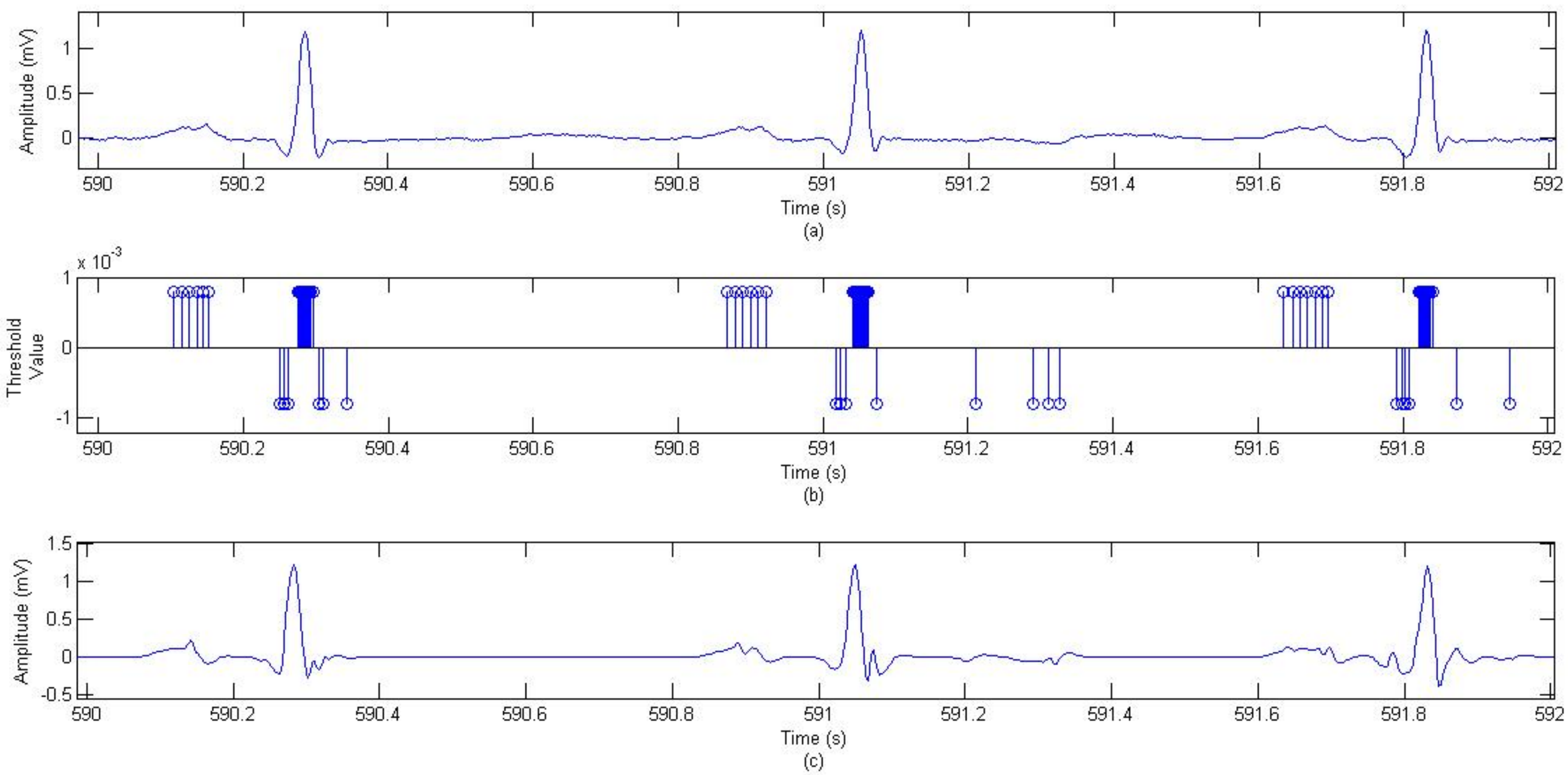
$$G(\underbrace{V_T}_{\text{terminals}},\underbrace{V_N}_{\text{Non-terminals}},\underbrace{P}_{\text{Production rules}},\underbrace{S}_{\text{Start symbol}})$$

with a finite set of attributes for each symbol

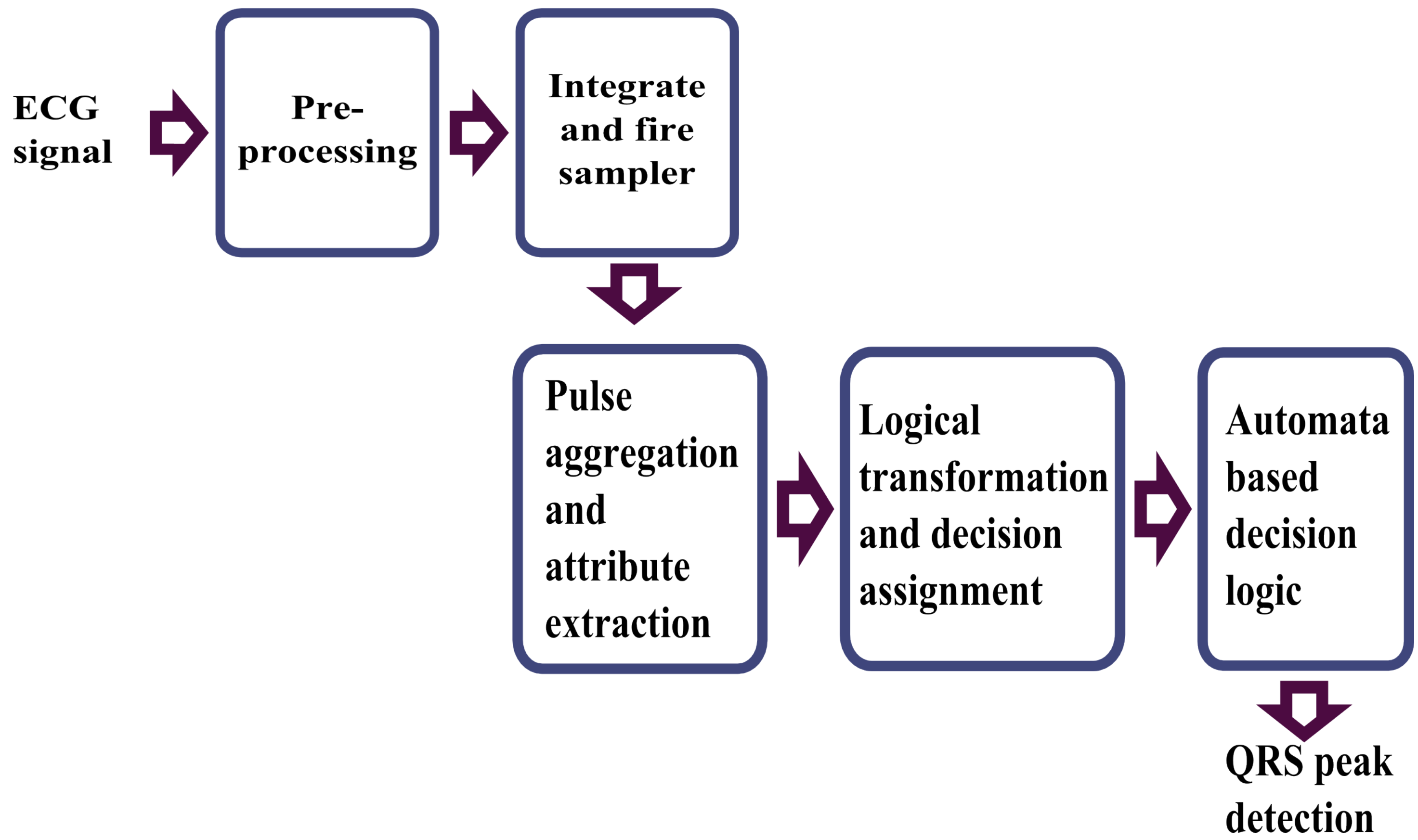
- Attribute grammars combine both syntactic and statistical approaches and incorporates language syntax and contextual semantics.

Application: ECG Beat Detection

30 pulses/sec (~8 bits) versus 100 Hz (8 bits)



SP Architecture



Nallathambi G., Principe J., “Integrate and Fire Pulse Train Automaton for QRS detection” IEEE Trans. Biomed Eng. , 2013.

Attribute Extraction

TIME Attributes

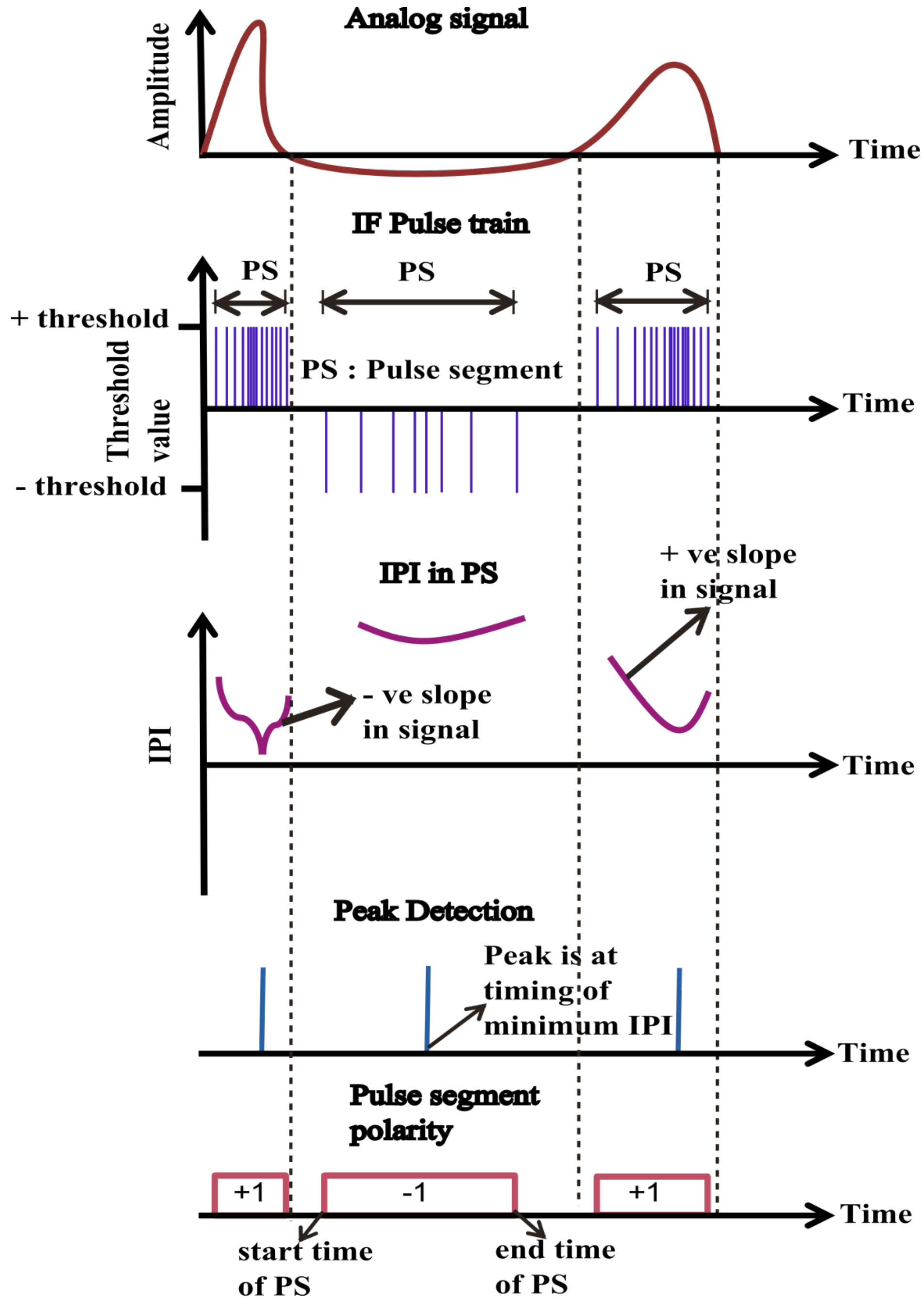
- **Pulse count**
- **Start time**
- **End time**
- **Minimum IPI**
- **Time mIPI**

All can be implemented by combinatory logic

Attribute vector

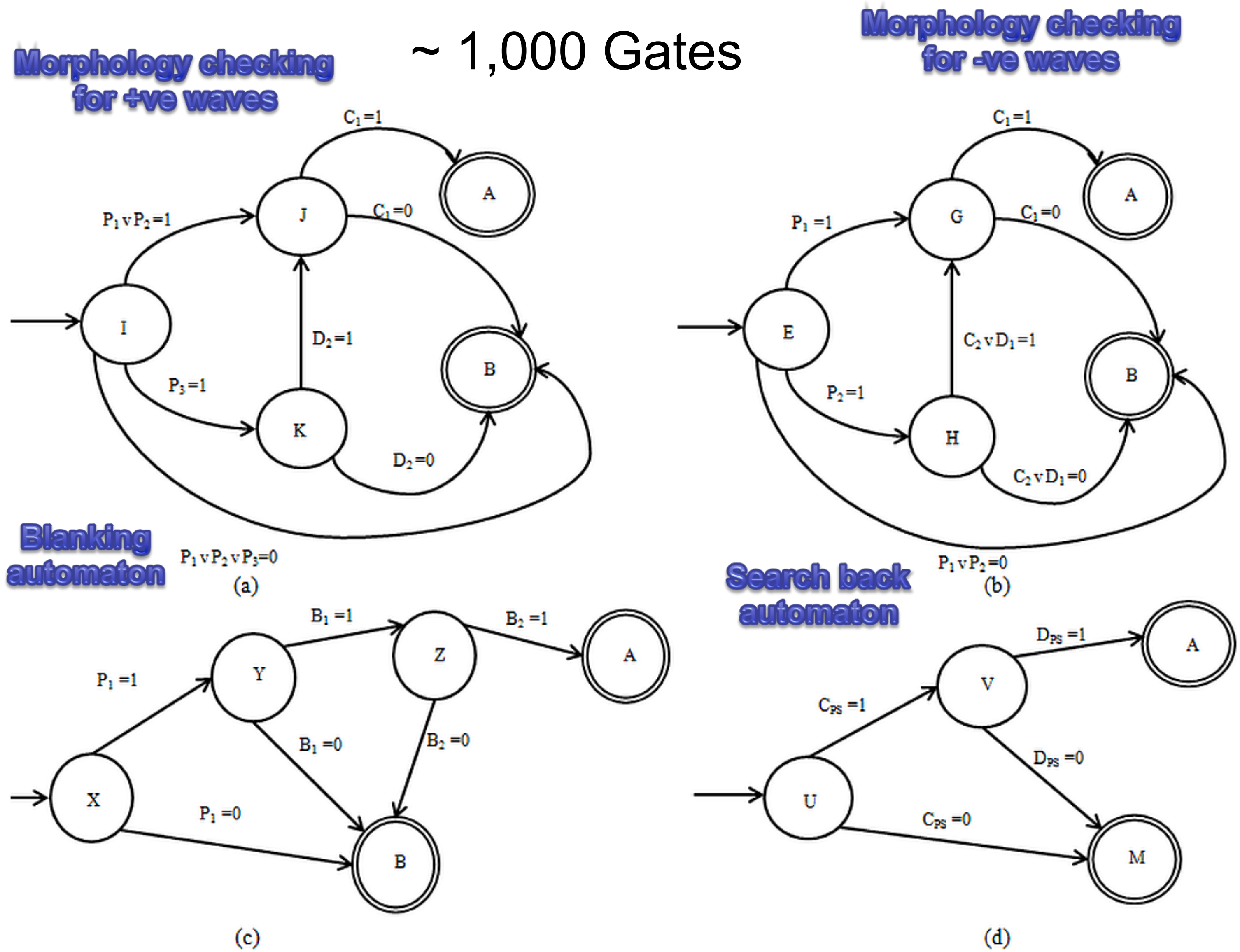
$$A(p^+) = A(p^-) =$$

$$= \{pc, st, et, miv, mit\}$$



Automata Based Decision Logic

~ 1,000 Gates



Comparison

Tested with MIT-BIH arrhythmia database

min[S_e, +P]	Algorithm
99.5%	**Proposed method**
	Hamilton and Tompkins [18]
	Afonso et al. [19]
	Bahoura et al. [20]
>99%	Inoue et al. [21]
	Li et al. [22]
	Poli et al. [23]
	Kohler et al. [24]
95%-99%	Sun et al. [39]
	Suppappola and Sun [40]

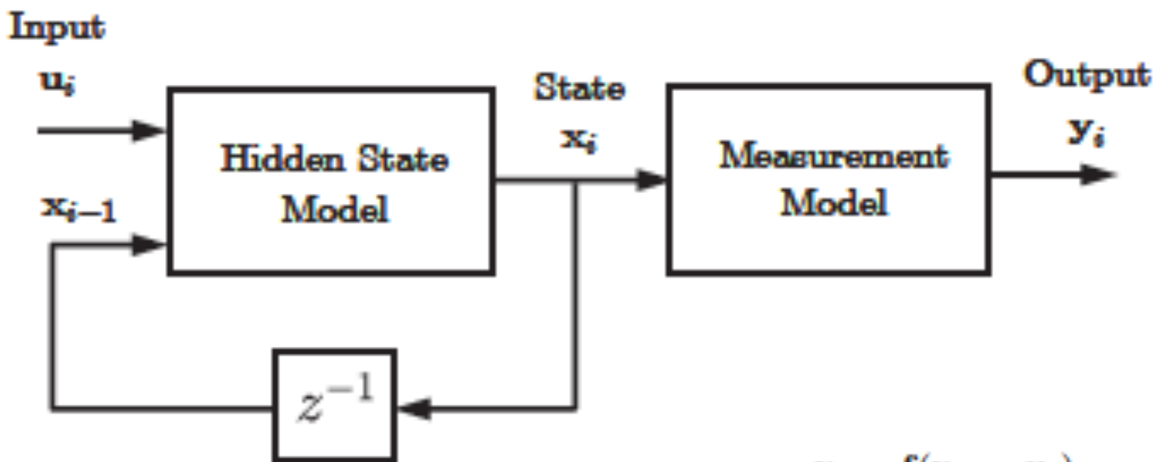
How to Learn Automata from Data?

So far the DFA and AG have been determined from the clinical ECG knowledge. So this is restrictive and requires human intervention.

Goal: Use ideas of kernel autoregressive filters (KAARMA) to learn the input structure through prediction, and then extract the grammars from the KAARMA

Li Kan, Principe J., “Kernel Adaptive Auto Regressive Moving Average Algorithm”, accepted IEEE Trans. Neural Networks and Learning Systems, 2015

State Models in RKHS



$$\mathbf{x}_i = \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i)$$
$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i)$$

where

$$\mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \triangleq \left[f^{(1)}(\mathbf{x}_{i-1}, \mathbf{u}_i), \cdots, f^{(n_x)}(\mathbf{x}_{i-1}, \mathbf{u}_i) \right]^T = \left[\mathbf{x}_i^{(1)}, \cdots, \mathbf{x}_i^{(n_x)} \right]^T$$

$$\mathbf{h}(\mathbf{x}_i) \triangleq \left[h^{(1)}(\mathbf{x}_i), \cdots, h^{(n_y)}(\mathbf{x}_i) \right]^T = \left[\mathbf{y}_i^{(1)}, \cdots, \mathbf{y}_i^{(n_y)} \right]^T$$

The advantage is that a linear model in RKHS is a nonlinear model in the input space.

Liu W., Principe J., Haykin S., "Kernel Adaptive Filtering: a Comprehensive Introduction", John Wiley, 2010.

State Models in RKHS- Our Approach

Rewrite the dynamical system equations as

$$\mathbf{s}_i \triangleq \begin{bmatrix} \mathbf{x}_i \\ \mathbf{y}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \\ \mathbf{h} \circ \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \end{bmatrix}$$

$$\mathbf{y}_i = \mathbf{s}_i^{(n_s - n_y + 1 : n_s)} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{n_y} \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ \mathbf{y}_i \end{bmatrix}$$

$$\mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i) = \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i)$$

$$\mathbf{x}_i = \mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i)$$
$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i) = \mathbf{h} \circ \mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i).$$

Map the augmented state s(.) and u(.) to two separate RKHS and then create a product kernel $\mathcal{H}_{su} \triangleq \mathcal{H}_s \otimes \mathcal{H}_u$ (tensor product)

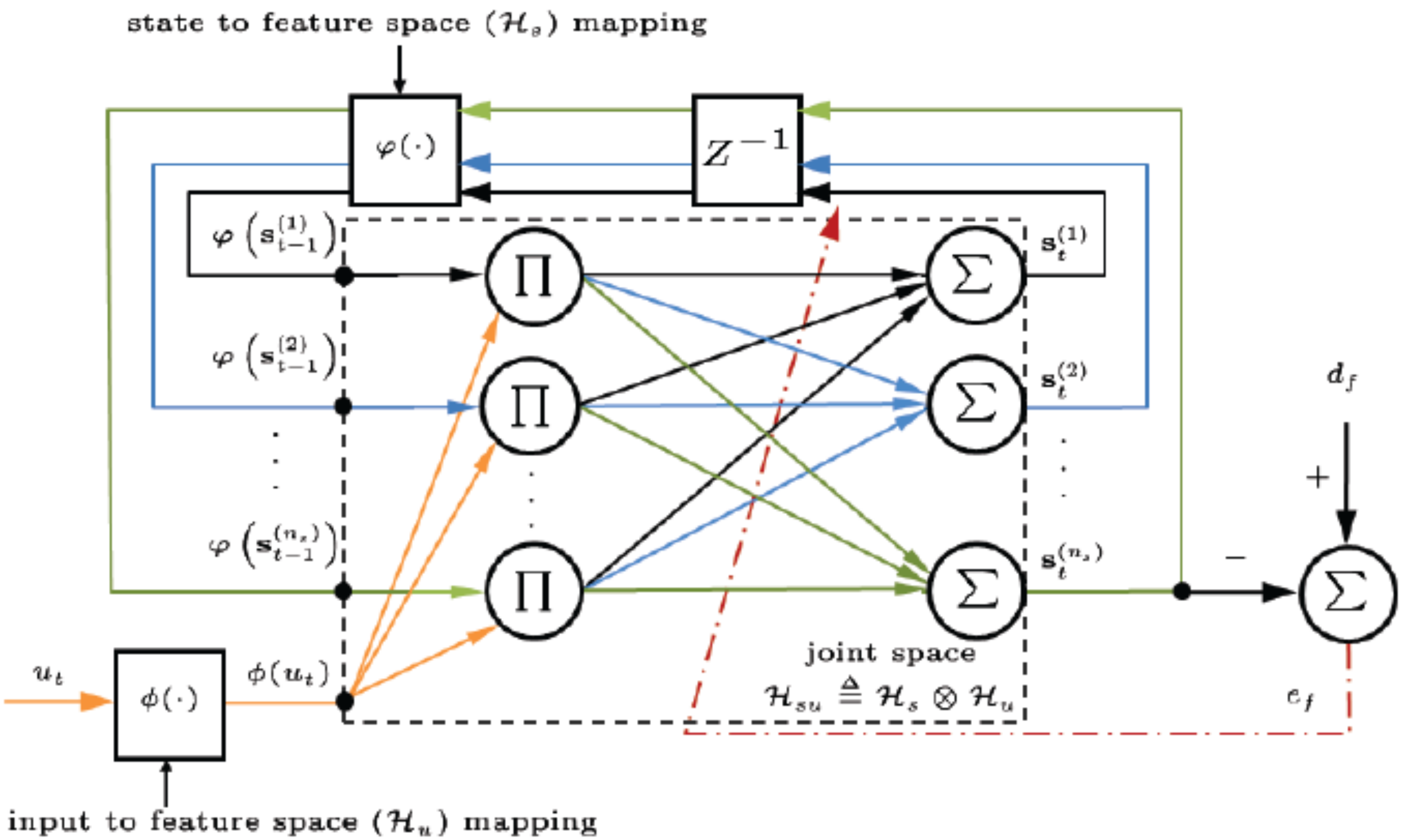
$$\mathbf{\Omega} \triangleq \mathbf{\Omega}_{\mathcal{H}_{su}} \triangleq \begin{bmatrix} \mathbf{g}(\cdot, \cdot) \\ \mathbf{h} \circ \mathbf{g}(\cdot, \cdot) \end{bmatrix}$$

$$\psi(\mathbf{s}_{i-1}, \mathbf{u}_i) \triangleq \varphi(\mathbf{s}_{i-1}) \otimes \phi(\mathbf{u}_i) \in \mathcal{H}_{su}.$$

$$\mathbf{s}_i = \mathbf{\Omega}^T \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)$$
$$\mathbf{y}_i = \mathbf{W}_m \mathbf{s}_i.$$

$$\begin{aligned} \langle \psi(\mathbf{s}, \mathbf{u}), \psi(\mathbf{s}', \mathbf{u}') \rangle_{\mathcal{H}_{su}} &= \mathcal{K}_{su}(\mathbf{s}, \mathbf{u}, \mathbf{s}', \mathbf{u}') \\ &= (\mathcal{K}_s \otimes \mathcal{K}_u)(\mathbf{s}, \mathbf{u}, \mathbf{s}', \mathbf{u}') \\ &= \mathcal{K}_s(\mathbf{s}, \mathbf{s}') \cdot \mathcal{K}_u(\mathbf{u}, \mathbf{u}'). \end{aligned}$$

State Models in RKHS



Parameters can be trained with Real time Recurrent Learning

Syntactic Pattern Recognition with KAARMA

- Problem: Given a set of positive and negative training sequences, describe the discriminating property of the two.

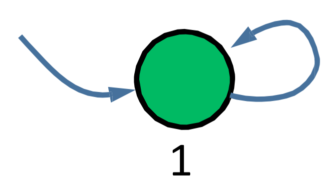
Positive Samples	**Negative** Samples	
1	10	
11	01	(Tomita regular grammar # 1)
111	00	
1111	011	
11111	110	
111111	11111110	

Solution:

English: Accept any binary string that does not contain ‘0’.

Regular Expression: 1*

or Deterministic Finite Automaton (*DFA*):



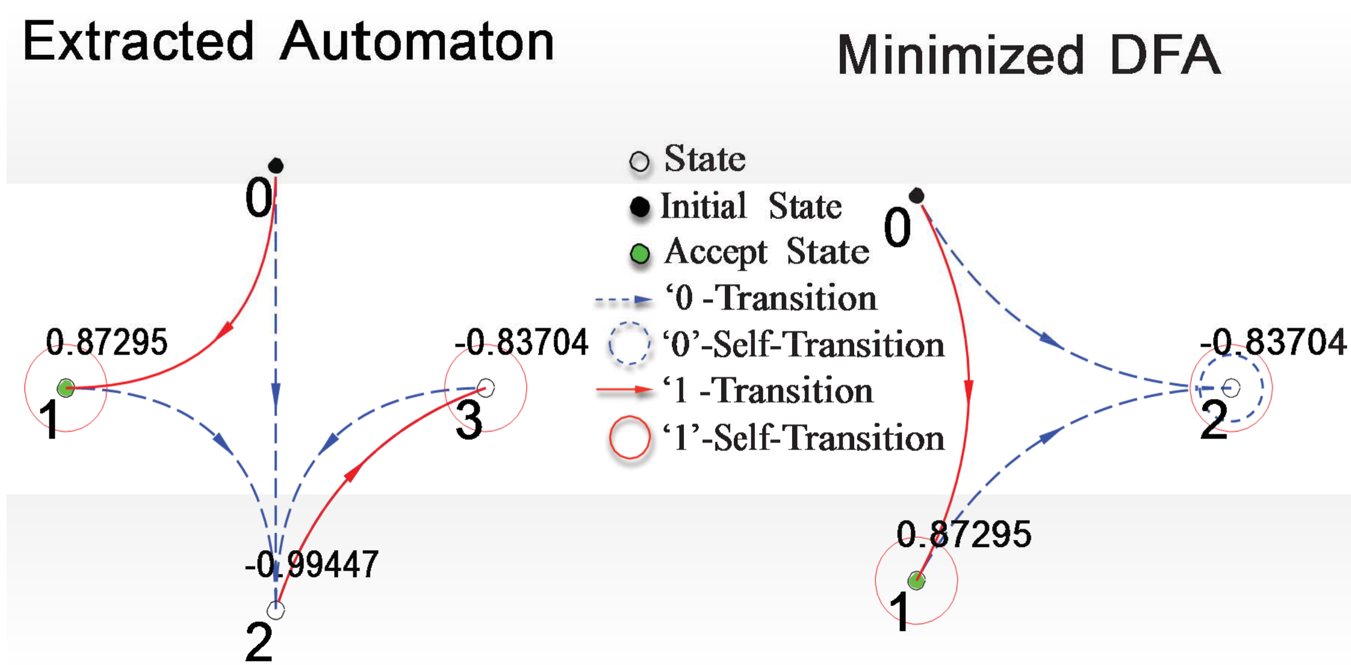
Tomita Grammars

No.	Description
1	1^*
2	$(10)^*$
3	No odd number of consecutive 0's after an odd number of consecutive 1's.
4	Any string with fewer than three consecutive 0's.
5	Any even length string with an even number of 1's.
6	Difference b/w number of 1's and 0's is a multiple of 3.
7	$0^*1^*0^*1^*$

- Training set consists of 1000 randomly generated binary strings, with lengths of 1-15 symbols (mean length is 7.758), and labeled according to grammar.
- The stimulus-response pairs are presented to the network sequentially: one bit at a time.
- At the conclusion of each string, the network weights are updated.

Tomita Grammar Extraction

- First the state of the KAARMA is bynarized (+/- 1)
- DFA is reduced using the Mealy procedure
- KAARMA generated DFA for Tomita grammar #1.



Results in Tomita Grammars

	Inference Engine	train size	test error	accuracy	network size	Extraction w. bynarized state	DFA size
Grammar 1	KAARMA	170	4	99.994	43.3	1.00	4.5
	RNN (Miller & Giles '93)	23000	1	99.999	9 (1st)	1.00	9.2
	RG (Schmidhuber & Hochreiter '96)	182	-	-	1 (A1)	-	-
Grammar 2	KAARMA	700	3	99.995	29.8	1.00	6.0
	RNN	77000	5	99.992	9 (2nd)	1.00	9.9
	RG	1511	-	-	3 (A1)	-	-
Grammar 4	KAARMA	900	1343	97.919	25	1.00	8.2
	RNN	46000	1240	98.078	9 (2nd)	0.81	12.3
	RG	13833	-	-	2 (A1)	-	-
Grammar 6	KAARMA	1160	2944	95.437	36.6	1.00	5.5
	RNN	49000	8725	86.475	9 (2nd)	0.67	10.5
Grammar 7	KAARMA	4400	4623	92.834	30.2	1.00	10.8
	RNN	121000	889	98.622	9 (2nd)	0.86	10.7

Summary

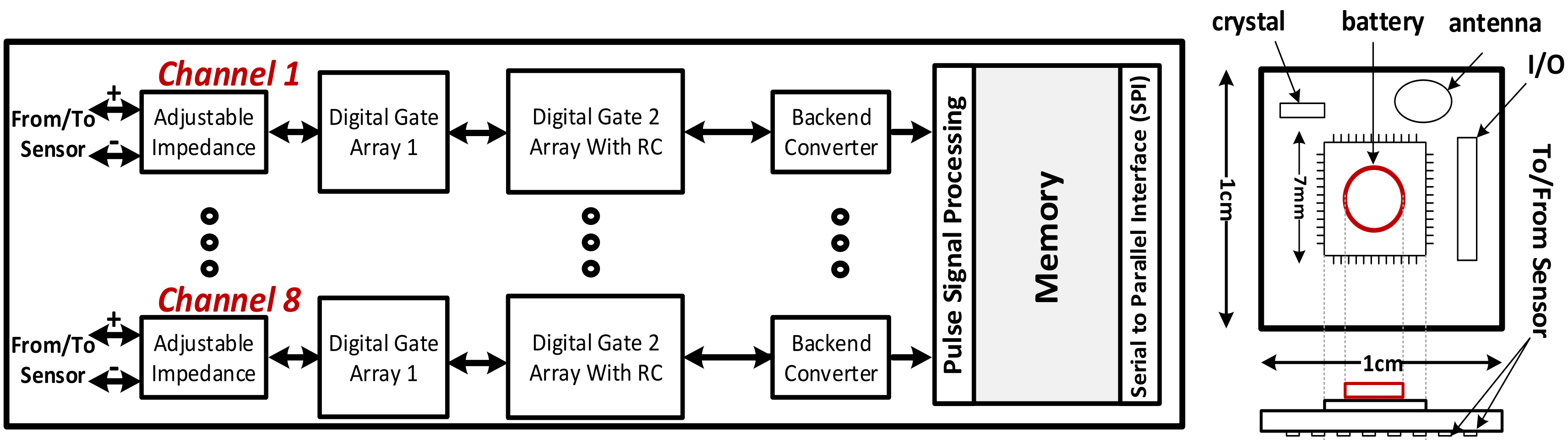
These and other tests show that the KAARMA is a powerful method to extract temporal patterns directly from data.

For real world applications we still need to implement the attribute grammars to handle noise.

The corresponding DFAs can then be implemented directly in small programmable gate arrays or customized VLSI chips for each application with minimal hardware resources.

The Future:
Fully Reprogrammable & Synthesizable Analog / Digital Circuits

Goal: to implement the ECG detector in ultra low power logic using < 5 μWatts.



(a)

(b)

Full use of digital gates, even for analog amp V_{dd}=0.4 v.

Pulse Domain Arithmetic

Any finite bandwidth signal can be decomposed as

$$f(t)|_{t=nT} = \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} f(\lambda)\delta(\lambda - nT)d\lambda$$

In practice the delta function is replaced by short pulses of Δt duration

$$f(t)|_{t=nT} \approx \sum_{n=-\infty}^{\infty} \int_{nT}^{nT+\Delta t} f(\lambda)d\lambda$$

Suppose we constrain the area to θ

$$\int_{t_n}^{t_n+\Delta t_n} f(\lambda)d\lambda = \theta$$

which is what the IFS does. Don't loose information about f(t) if we put out a time marker when the area constraint is reached (the pulse), then the time between two consecutive pulses is θ.

$$f(t_n) \cdot \Delta t_n = \theta.$$

Pulse Based Arithmetic Units, Patent pending #23308560, Aug 2015

Arithmetic with IFS Pulse Trains

- Goals:

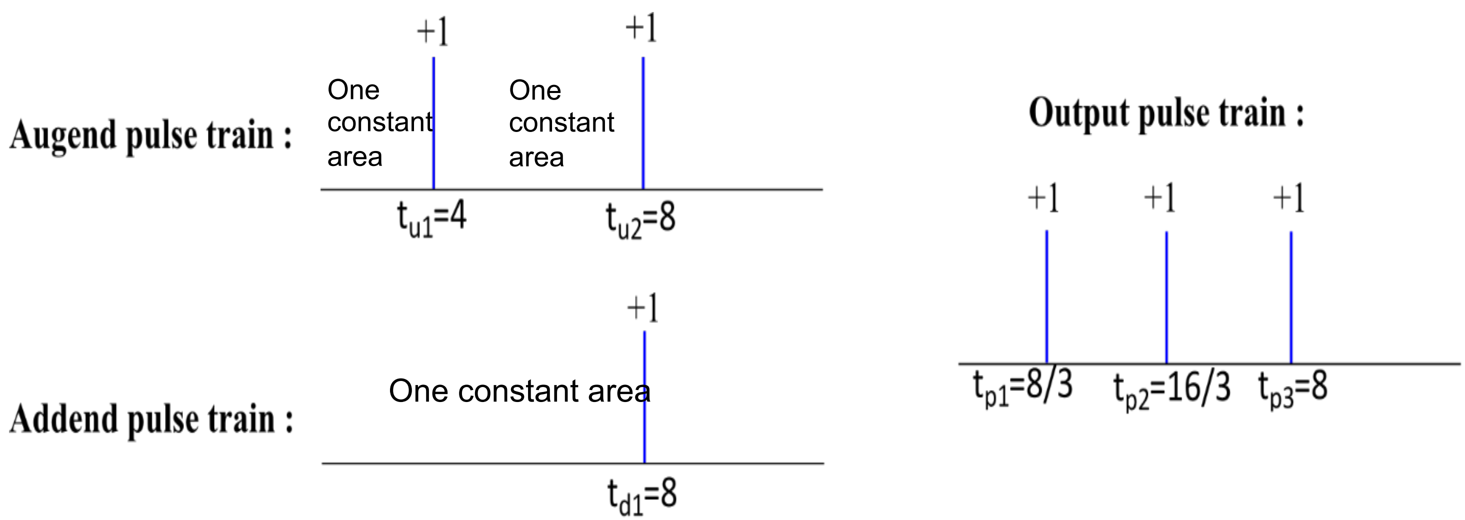
  1. Algebraically process the information in analog signals by converting them to IFS pulse trains with amplitudes of +/- 1.

  2. Perform addition and multiplication with IFS pulse trains to mimic the operations of instantaneous addition and multiplication on the analog signals
- Information will be exclusively in the time domain
  - Inputs – Pulse trains
  - Output – Pulse train

Guidelines for Pulse Domain Arithmetic

- Known:
  - Time between two pulses satisfies the area constraint.
- To perform arithmetic:
  - Assume all pulse trains are generated by the same IFS parameters.
  - Relate pulse differences to areas to find out when to include pulses in the time line resulting from the binary operation of addition/multiplication.
  - Because pulses occurring in two signals are asynchronous, it is also necessary to quantify carryovers between subsequent evaluations.

Pulse Domain Addition - Illustration



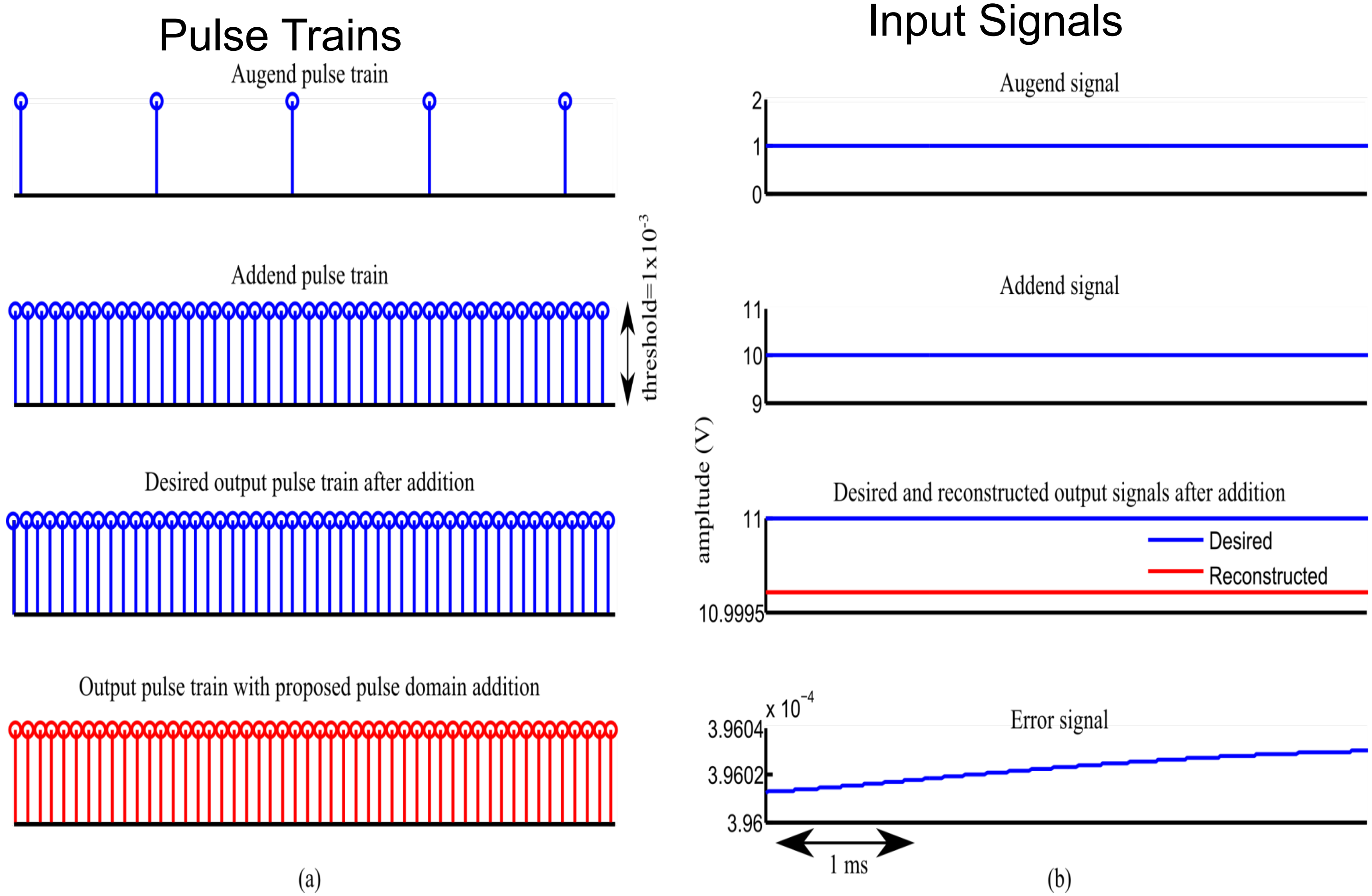
Pulse domain addition scheme

Time	Addend Area	Augend Area	Resultant sum area	Output pulse timing	Output pulse polarity
0 to t_{d1}=8	1	2	3	8/3, 16/3, 8	+1,+1,+1

Pulse Domain Addition – Algorithm

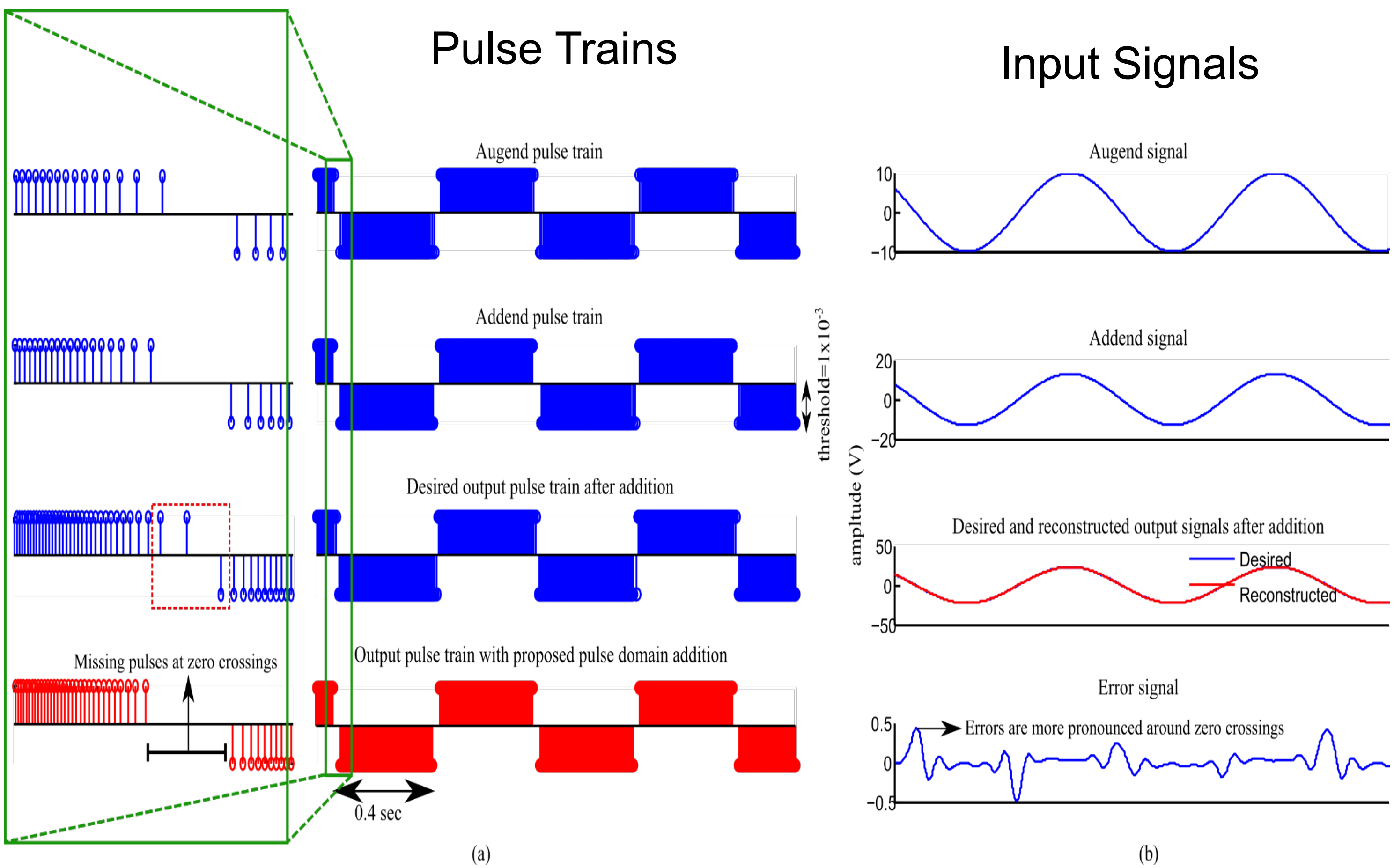
1. Find the number of constant areas resulting from augend and addend at a given pulse interval.
2. The floor function of the total number of constant areas defines the # of pulses of the output pulse train which represent the same constant area.
3. The fractional part in step 2 gives the carryover area which is added in the next pulse interval.

Results – Addition of Periodic Pulse Trains



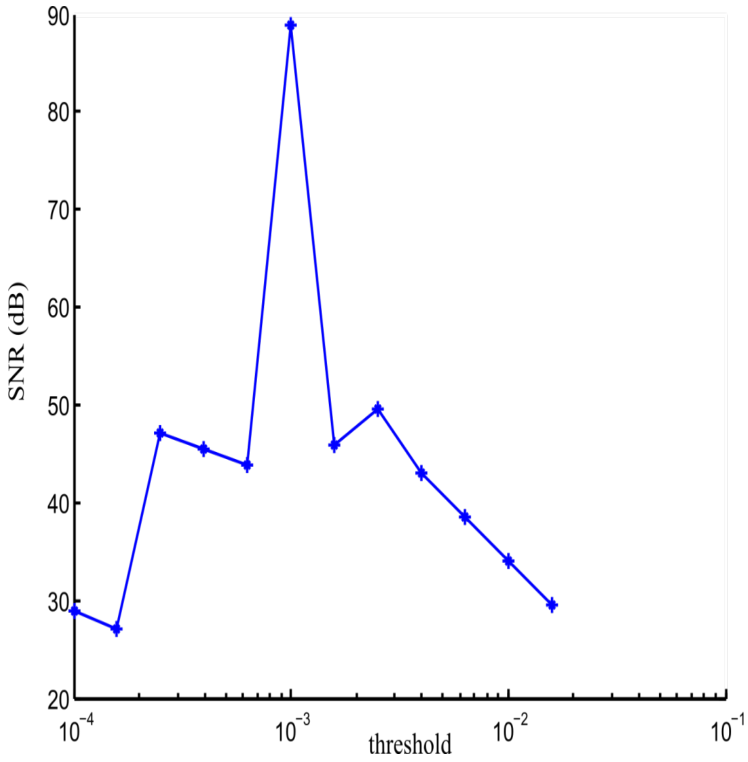
SNR is 88.87dB (simulation with 100 MHz time stamping and 1 MHz counters)

Results – Addition of Aperiodic Pulse Trains



SNR is 42.2dB (simulation with 100 MHz time stamping and 1 MHz counters)

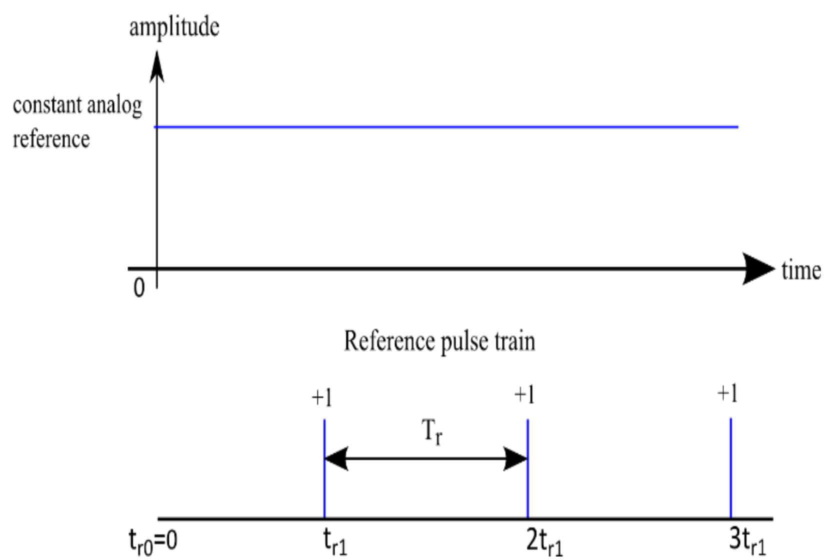
SNR of Addition is Under the Control of the User



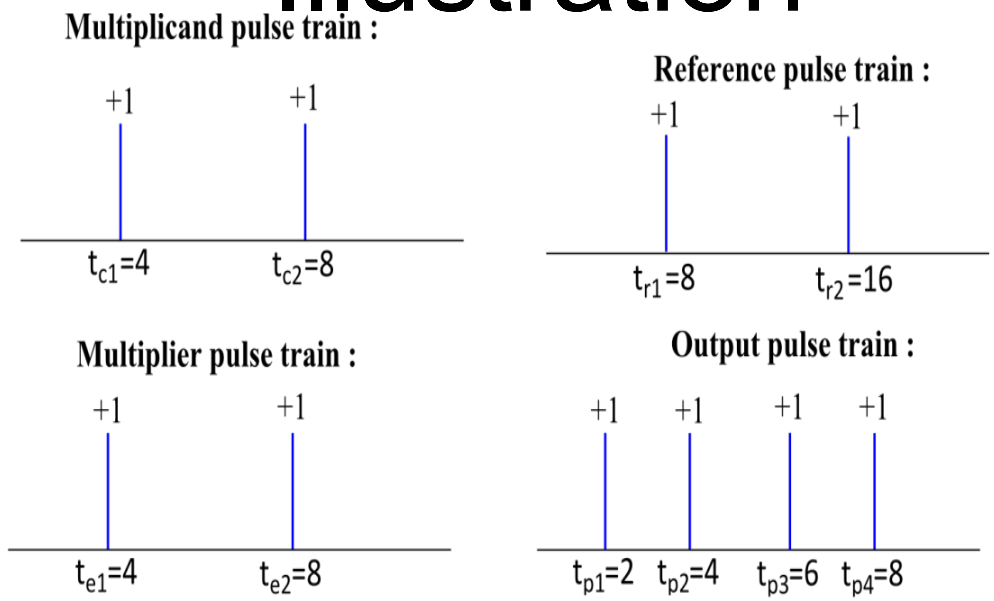
The IFS threshold can be adjusted appropriately to get the desired SNR.

Pulse Domain Multiplication

- To perform multiplication, we need to identity a pulse train reference - corresponding to a reference of one under the analog curve
  - This determines the contraction/expansion of timing in the output pulse train.



Pulse Domain Multiplication - Illustration



Pulse domain multiplication scheme

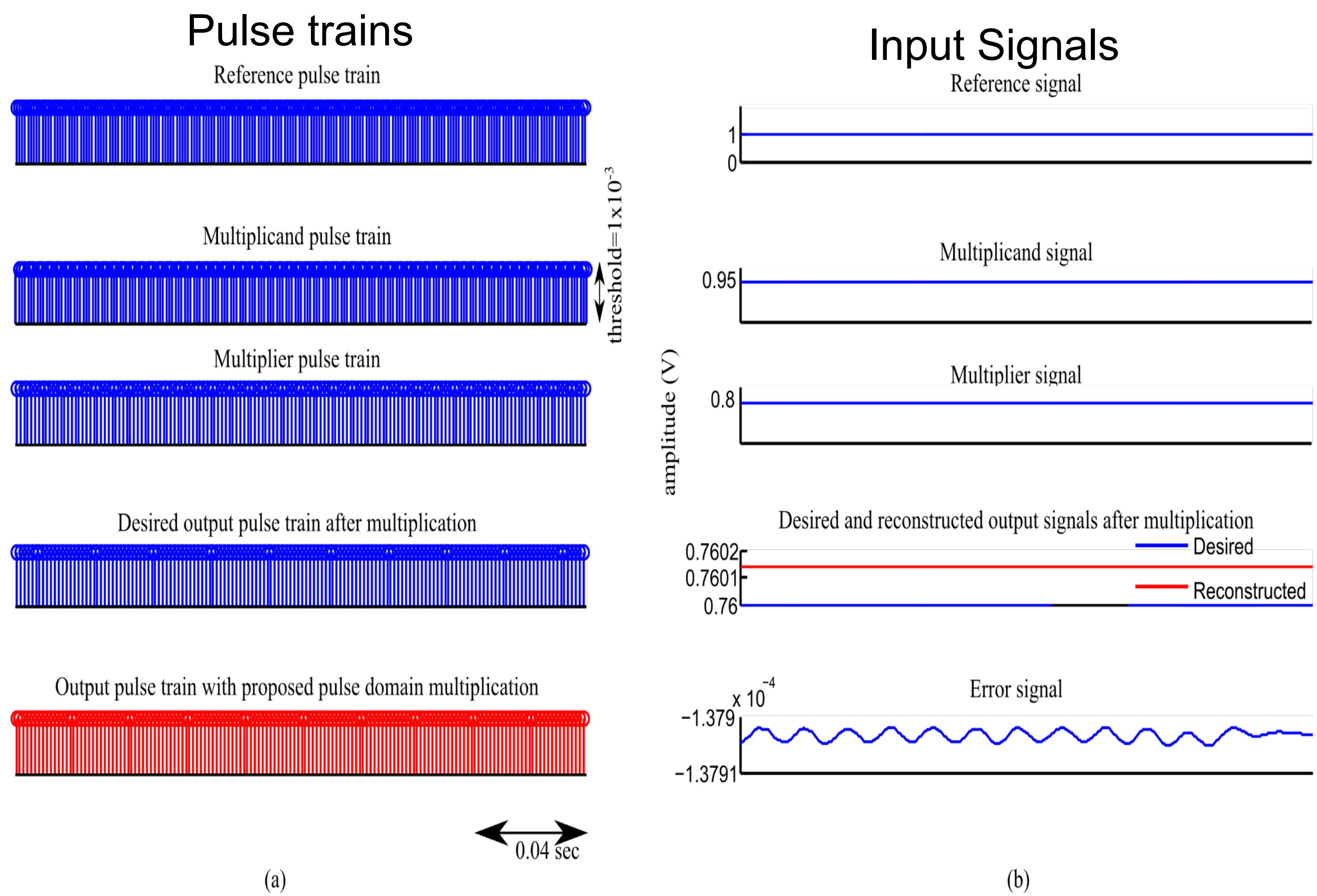
Relative multiplier area= Reference IPI/ Multiplier IPI

Time	Relative multiplier area	Multiplicand Area	Resultant product area	Number of output pulses	Output pulse timing	Output pulse polarity
0 to t_{e1}= 4	8/4=2	1	2*1=2	2	2, 4	+1, +1
t_{e1}=4 to t_{e2}= 8	$(\frac{8}{8-4})$=2	1	2*1=2	2	6, 8	+1, +1

Pulse Domain Multiplication – Algorithm

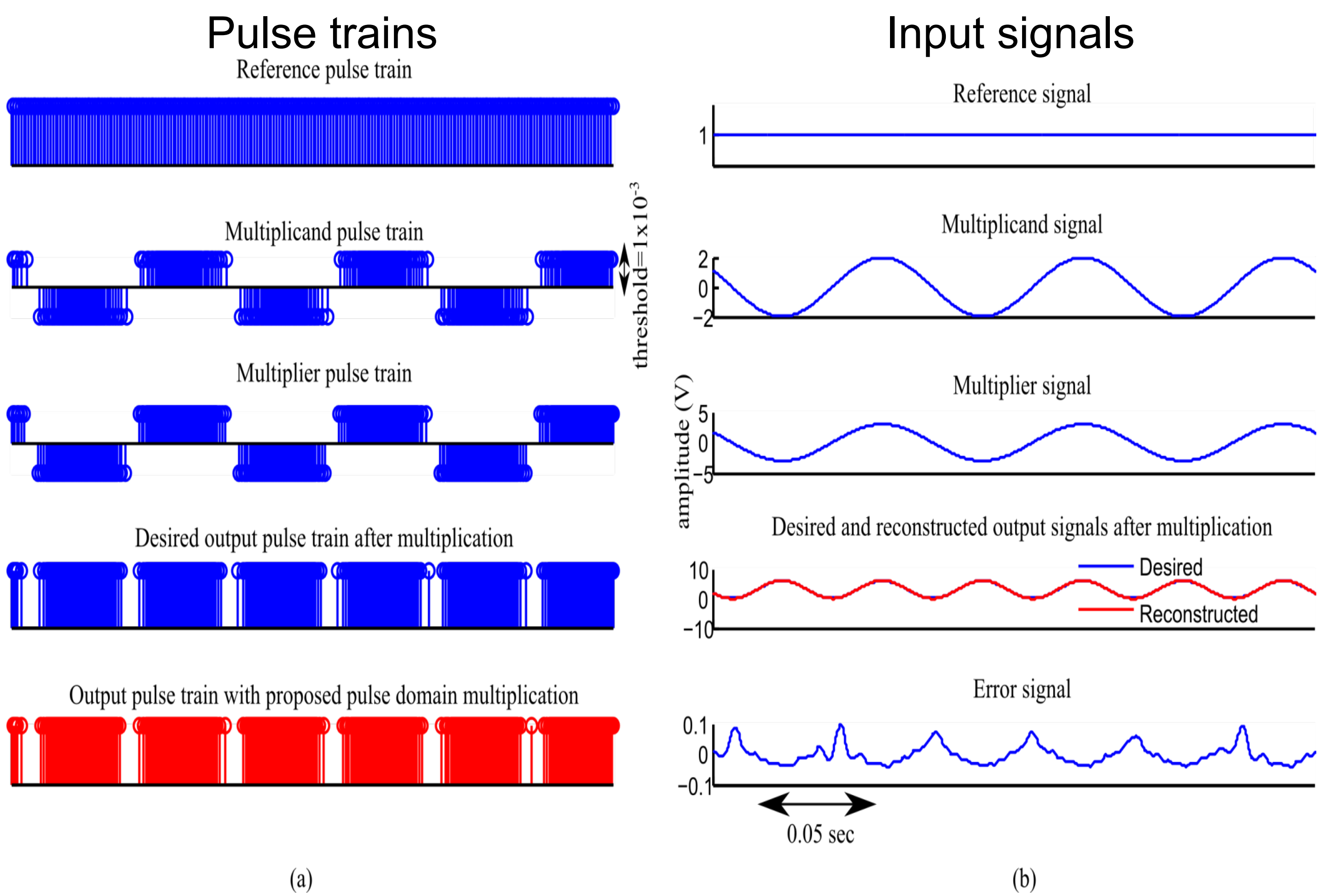
1. Find the number of constant areas resulting from multiplier by dividing the reference pulse train by the multiplier pulse interval.
2. Find the number of constant areas resulting from multiplicand at the given pulse interval.
3. The floor function of the product of the number of constant areas of step 1 and step 2 gives the output pulse train which represent the same constant area.
4. The fractional part of step 3 gives the carryover area which is added in the next pulse interval.

Results – Multiplication of Periodic Pulse Trains



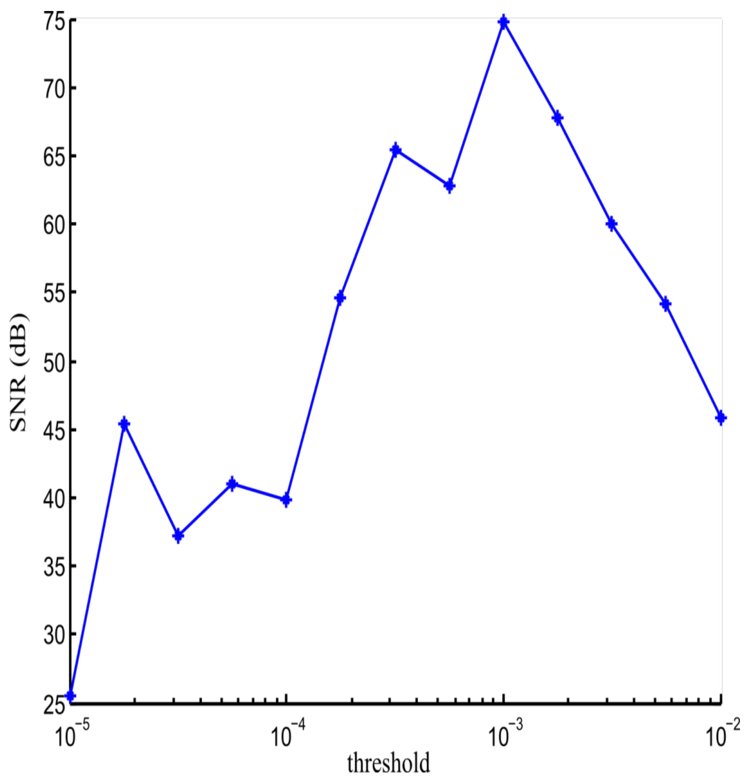
SNR is 74.82dB (simulation with 100 MHz time stamping and 1 MHz counters)

Results – Multiplication of Aperiodic Pulse Trains



SNR is 41.12 dB (simulation with 100 MHz time stamping and 1 MHz counters)

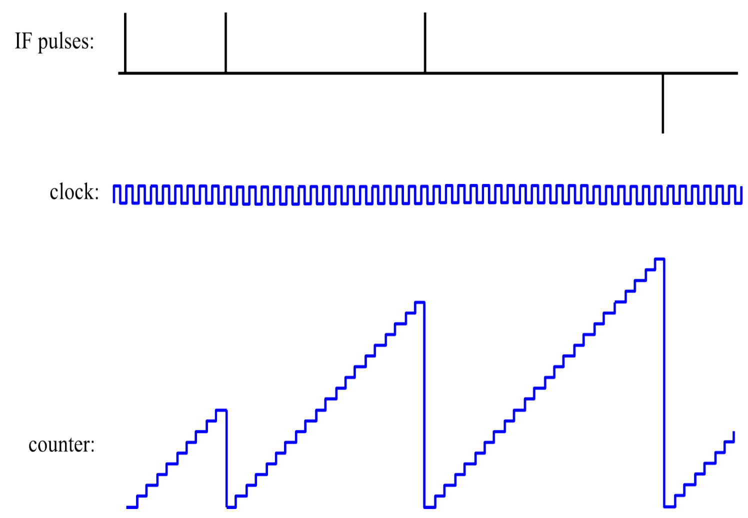
SNR of Multiplication is Under User Control



The threshold can be adjusted appropriately to get the desired SNR.

Current Work

- We have theoretically proved that pulse train algebra forms a Field.
- This allows inner products with pulse trains which is the foundation of signal processing.
- We are also developing low power architectures for processing with IFC pulse trains.
  - The main building block of the architecture will be counters.
  - It enables the quantification of information in time.



Conclusions

- Pulse trains created by the IFS do represent analog signals with an accuracy given by the threshold. So they can substitute ADCs for digital signal processing.
- It is possible to quantify properties of time signals using automata provided the user can infer the rules to achieve the goals.
- KAARMA and the binarization of its state appears as an automatic way of learning the automata structure directly from data.
- These automata can be implemented in very simple systems with the advantage of ultra low power due to dedicated architectures and ultra low Vdd.

Conclusions

- We also developed addition and multiplication in the pulse domain for general purpose computation with pulse trains.
- Right now this is a curiosity that expands signal processing in the analog domain using operators, instead of converting time into amplitude as most of the analog signal processing.
- If we can implement these rules in ultra low power Arithmetic Units this opens the door to a revolution in signal processing.
- If you are interested in this approach, please contact me.

principe@cnel.ufl.edu