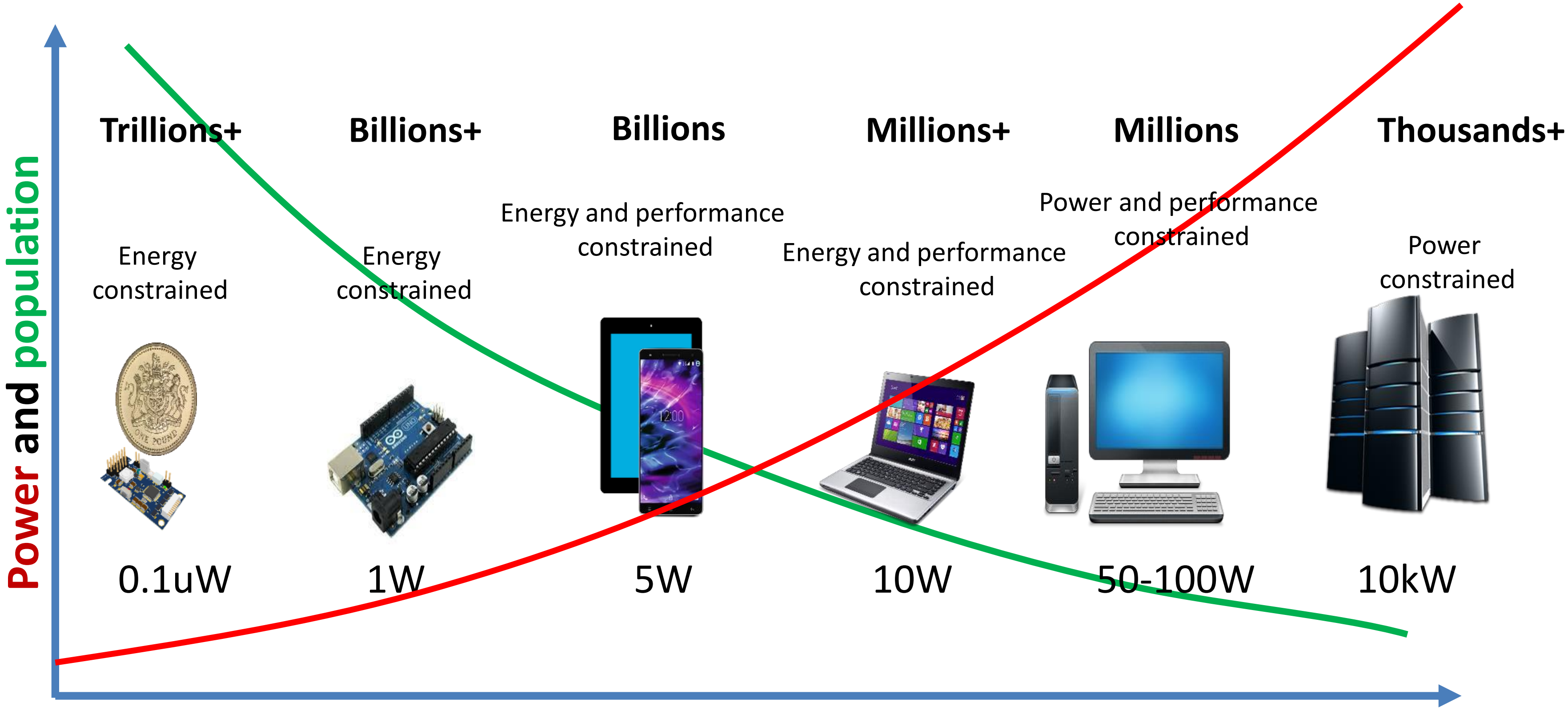
Energy drives logic

What kind of hardware do we need for pervasive AI?

Alex Yakovlev
Head of microSystems Group
School of Engineering
Newcastle University
https://www.ncl.ac.uk/engineering/research/eee/microsystems/
http://async.org.uk/

CAIR AI Workshop 7 February 2019

Swarm of devices – Future of ICT



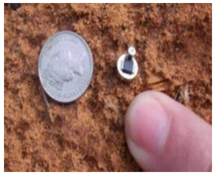
Power and population

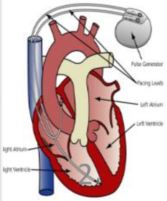
Trillions+ | Billions+ | Billions | Millions+ | Millions | Thousands+

Energy constrained | Energy constrained | Energy and performance constrained | Energy and performance constrained | Power and performance constrained | Power constrained

0.1uW | 1W | 5W | 10W | 50-100W | 10kW

Trillions of ubiquitous systems (sensors, probes, monitors, actuators, controllers) are being deployed to operate in myriad of places (organisation, human, body part, household, offices, pets) using harvested energy or micro-batteries

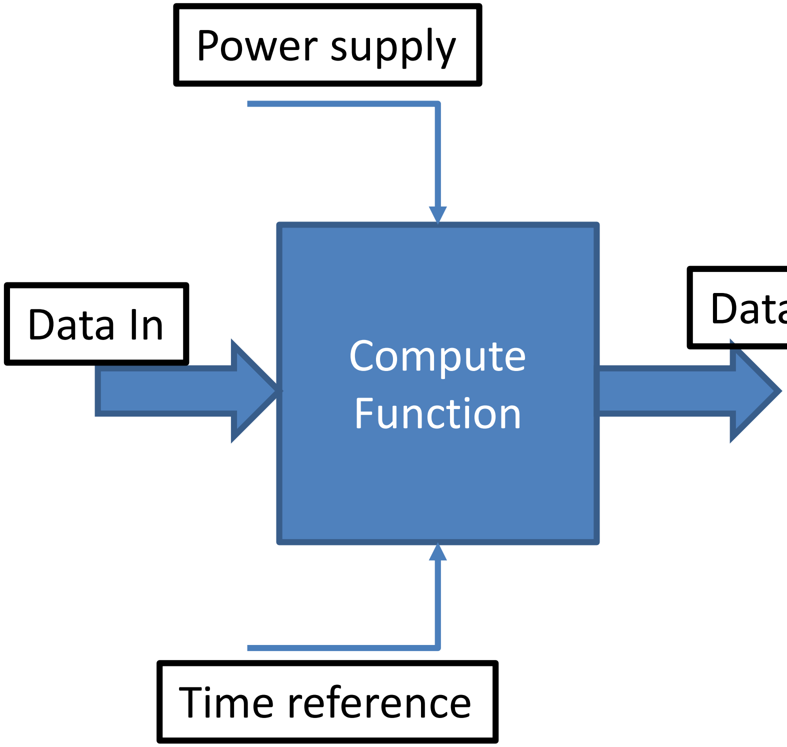




Granularity of intelligence

- Pervasive Intelligence requires reconsidering many balances:
  - Between software and hardware
  - Between power and compute
  - Between analog and digital
  - Between design and fabrication and maintenance
  - …
- Granularity of time and energy

Granularity of intelligence



Granulation phenomenon:

Granularity of power
Granularity of time
Granularity of data
Granularity of function

Questions:

- Can we granulate intelligence to minimum?
- What is the smallest level at which we can make cyber-systems **learn** – in terms of power, time, data and function?

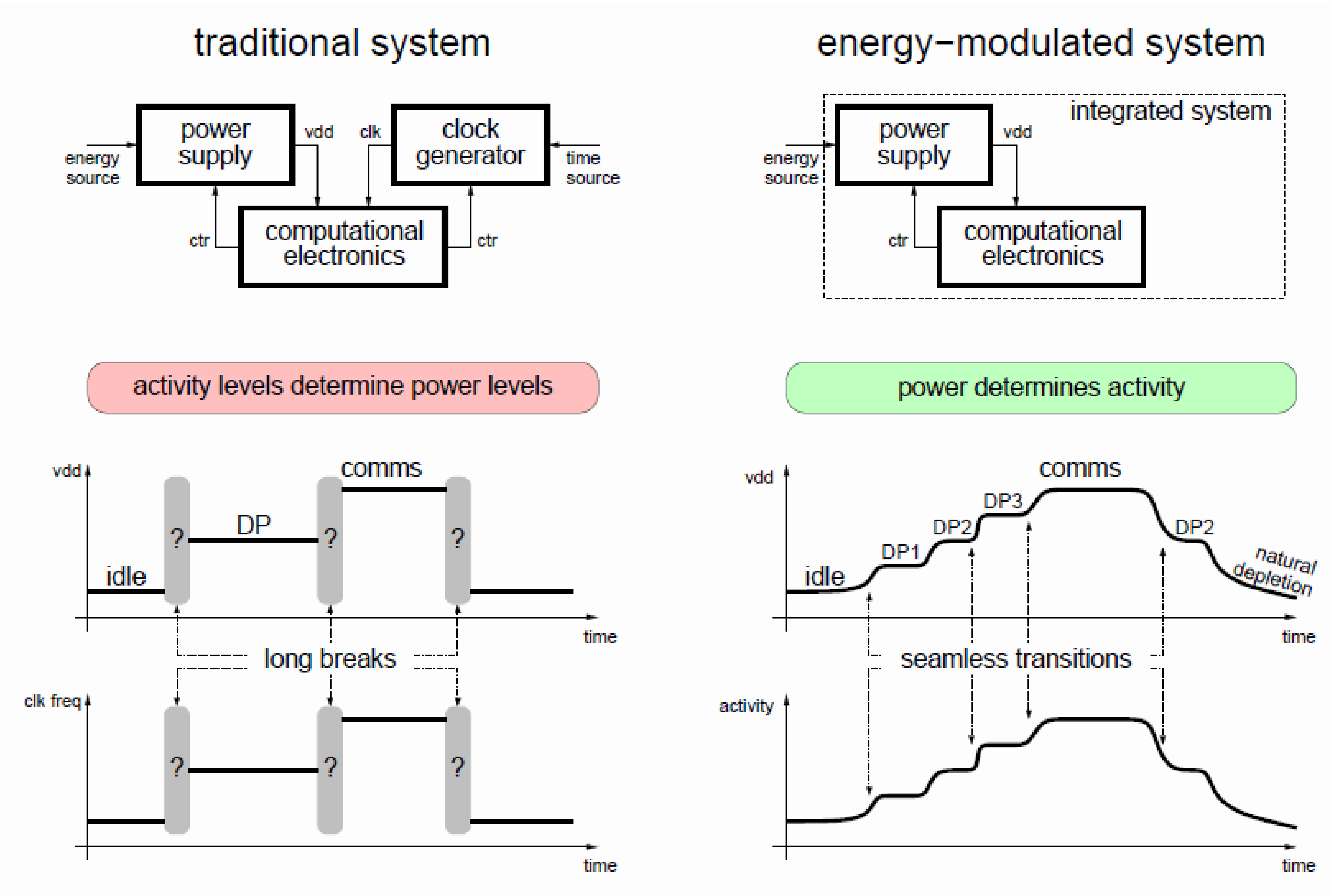
Grand challenge for pervasive hardware AI:

To enable electronic components with an ability to learn and compute in real-life environments with **real-power** and in **real-time**

Research Hypothesis:

We should design systems that are energy-modulated and self-timed, with maximally distributed learning capabilities

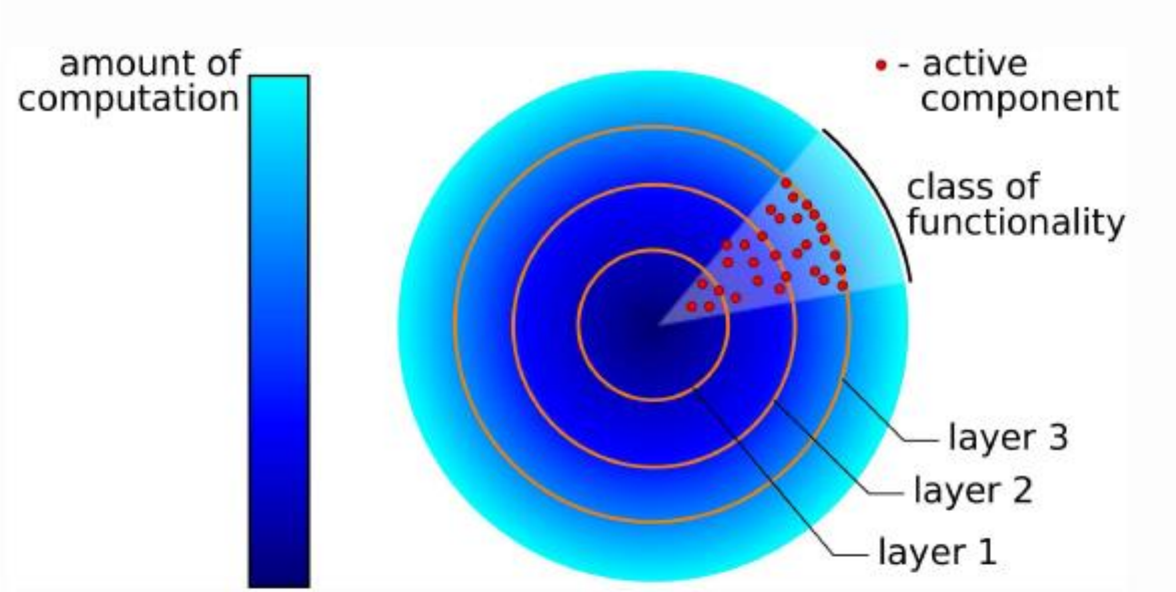
Energy-modulated computing



Vision for Future

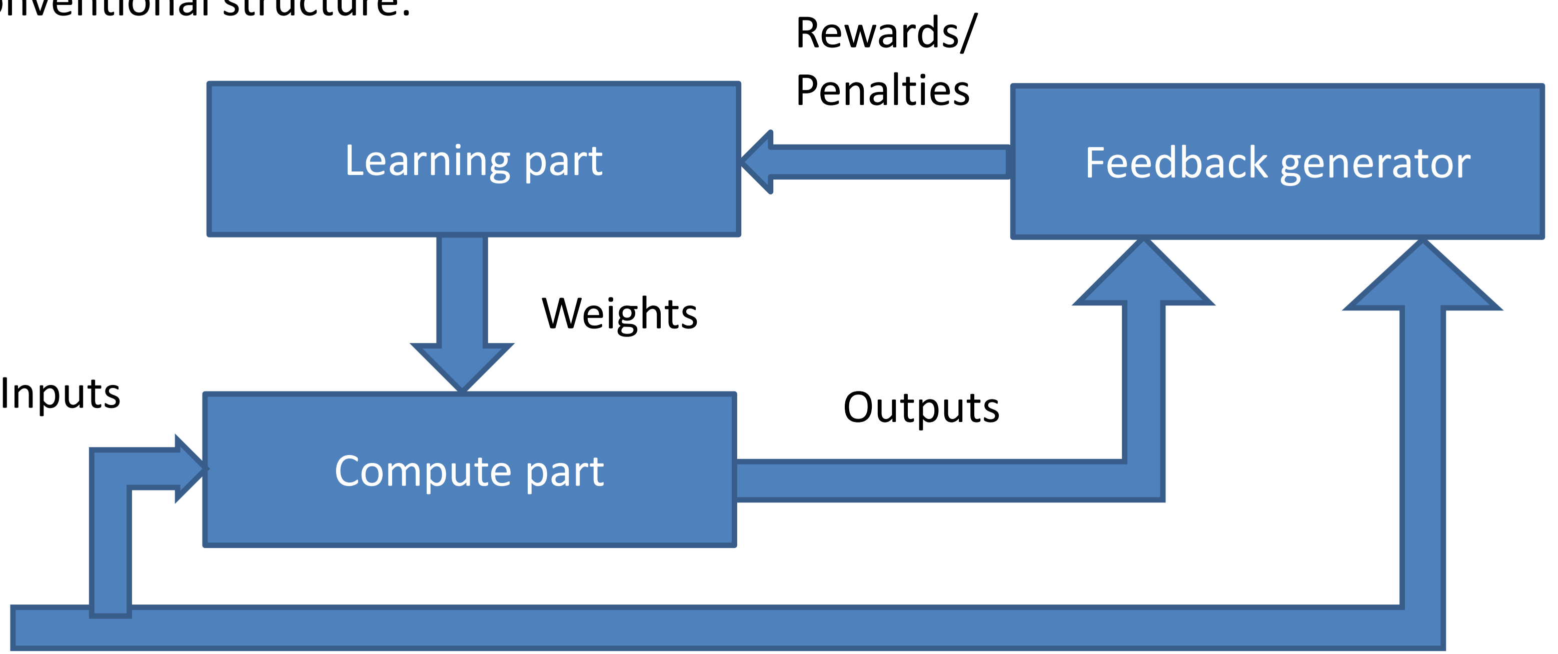
Power-modulated multi-layer system

- Multiple layers of the system design can turn on at different power levels (analogies with living organisms' nervous systems or underwater life, layers of different cost labour in resilient economies)
- As power goes higher new layers turn on, while the lower layers ("back up") remain active
- The more active layers the system has the more power resourceful it is



Learning Hardware

Conventional structure:



Most of the operations here are done by using conventional binary arithmetic, which is not power-adaptive and uses centrally provided Power and Clocks – hence poor power-proportionality and robustness

Proposed approach

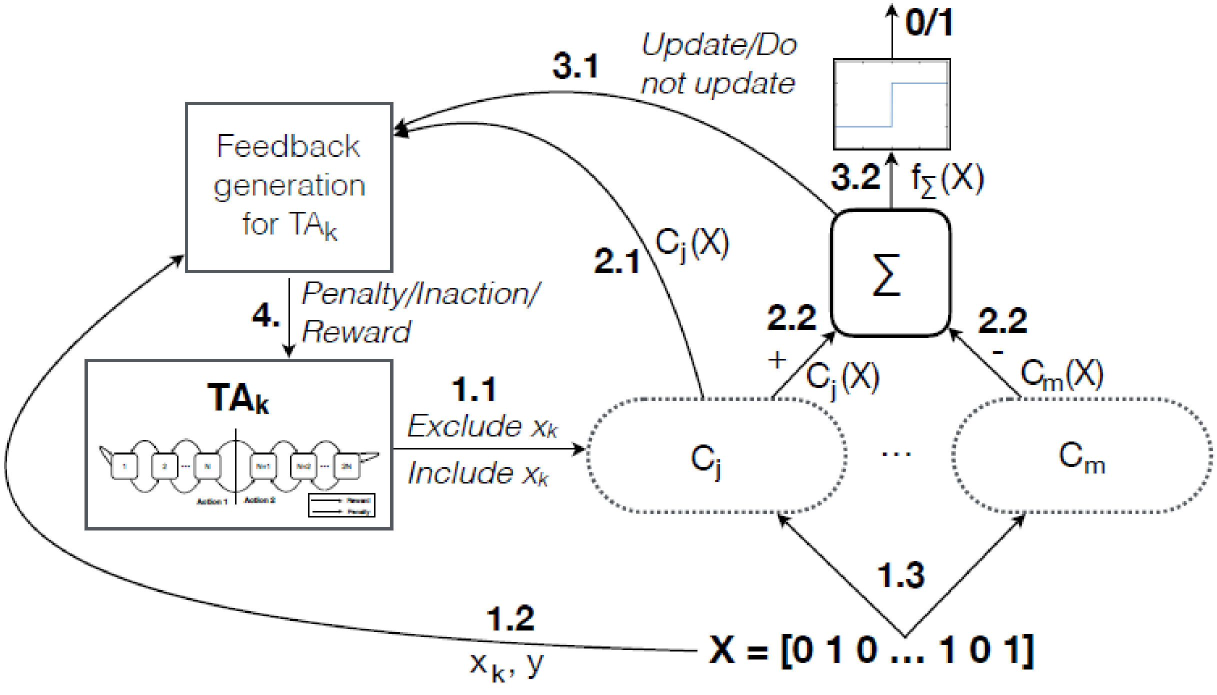
- Event-driven, robust to power and timing fluctuations
- Decentralised Tsetlin Automata (TAs) for learning on demand
- Mixed digital-analog compute where elements are enabled and controlled by individual TAs
- Natural approximation in its nature, both in learning and compute
- Asynchronous logic for h/w implementation

Why Tsetlin Automata?

Hypothesis: TAs provide a minimalist (energy-wise and robustness-wise) way to adaptation

- TAs can act as generators of control signals, naturally enabling:
  - Compute function shaping (include/exclude parts of compute)
  - Distributed Power and Clock gating
- TAs can be easily implemented in hardware:
  - Directly (in digital or mixed signal way)
  - Via microprogrammable structures,using transition-output tables in memory and simple access microcode in h/w
  - Can be prototyped in FPGAs and on microcontrollers
- TAs can be made :
  - With fixed structure (linear tactics)
  - With variable structure or with tunable memory depths
  - For stationary and varying environments

Granmo’s Tsetlin Machine Data flow



1.1 Tsetlin Automaton TA_k decides to include or exclude variable x_k in clause C_j

1.2 Value of x_k and y in training example (X, y) sent for generating feedback to TA_k

1.3 Training example X sent to each clause, including C_j

2.1 Output of clause C_j sent for generating feedback to TA_k

2.2 Output of all clauses (votes) sent for summation

3.1 Decision on whether to update the TA of clause C_j. Decision is based on $f_\Sigma(X)$ and T, and controls whether feedback is generated for TA_k.

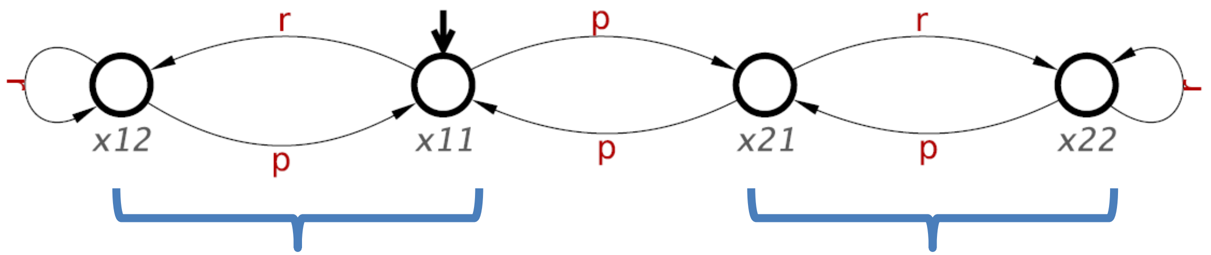
3.2 Output of summation, $f_\Sigma(X)$, sent to threshold function for classification

4. A penalty/reward/inaction generated from Feedback Type I (y = 1) or Type II (y=0) is sent to TA_k

The idea of Tsetlin Machine:
https://www.dropbox.com/s/usk78fj381k2qrw/Tsetlin_Machine_170119.pdf?dl=0

Tsetlin Automaton: async design

4-state TA_k:



Action1 (Exclude x_k)

Action2: Include x_k)

Logic implementation (equations obtained from our tool Workcraft – next slides):

$X11=x12'*x21'*x22'+p*A1$

$X12= r*A1 + x11'*x12$

$A1=p*(A2'*x21+x12+A1) + r*(x12+x11)$

$X21=x11'*x12'*x22'+p*A1$

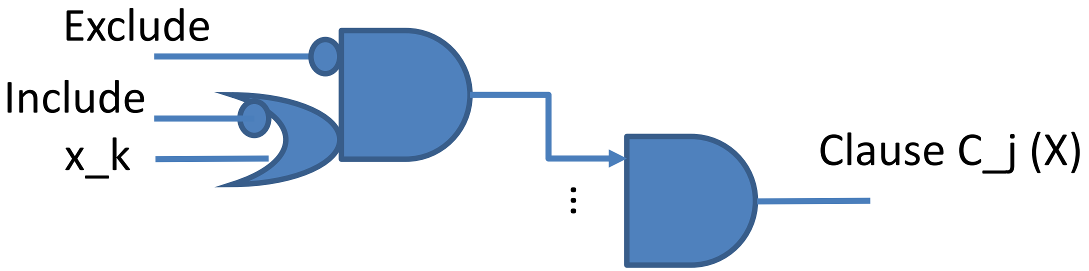
$X22= r*A2 + x21'*x22$

$A2=p*(A1'*x11+x22+A2) + r*(x22+x21)$

Approximate performance:

- response time can be in the order of 100ps
- energy cost in the order of 100fJ per action

Clause and function computation



- Clause can produce either levels or pulses
- We can use various energy-efficient ways of summing +1's and -1 as, or accumulating events (e.g. up and down counters)

Feedback computation (after logic minimisation)

Reward: R_kj=C_j*Inc_k*Type I

Penalty: P_kj=x_k*C_j*Exc_k*Type I + x_k'*C_j*Type II

Inaction: I_kj=C_j'+ x_k'*Type I + x_k*Type II

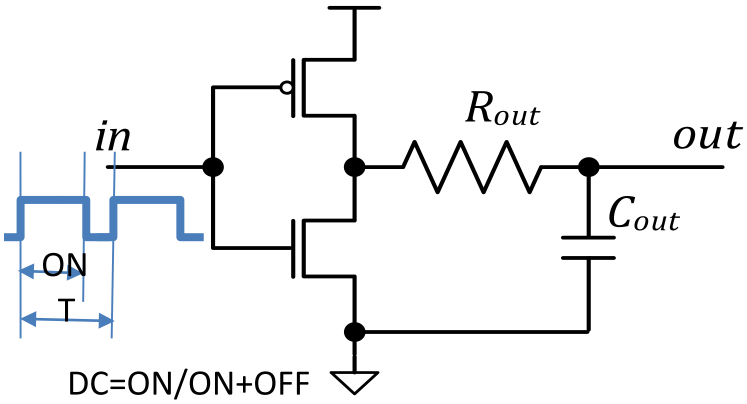
We assume:

- Exclude=Not(Include); Exc=0/Inc=1
- Type I = Not (Type II); Type I=0/Type II=1

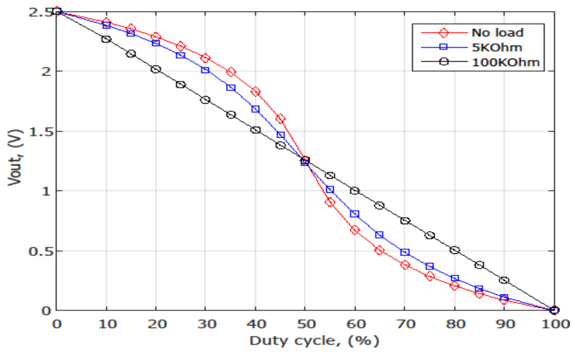
Interpretation:

- We reward TA_k to Include x_k in Cj for Type I
- We penalise TA_k to Exclude x_k when x_k=Cj=1 for Type I or to make x_k=0 and Cj=1 for Type II
- We maintain state for TA_k when Cj=0 or x_k=0 for Type I or x_k=1 for Type II

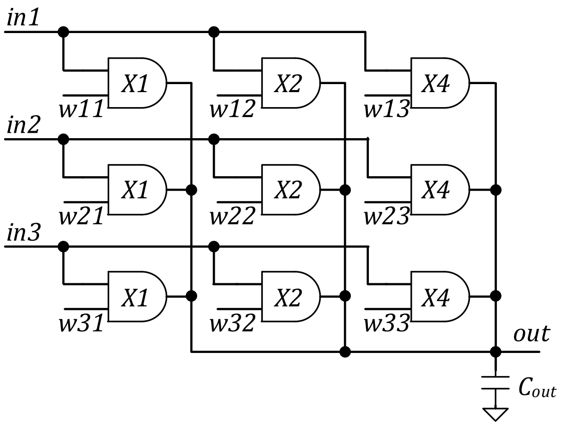
Designing the compute part: mixed analog-digital (DATE'19 paper)



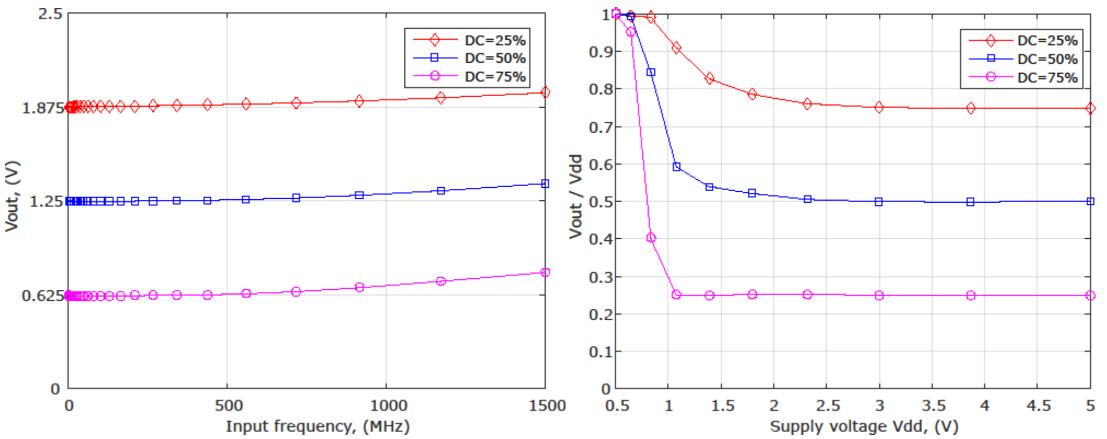
- PMOS and NMOS act as voltage divider.
- Output voltage is inversely proportional to input duty cycle.



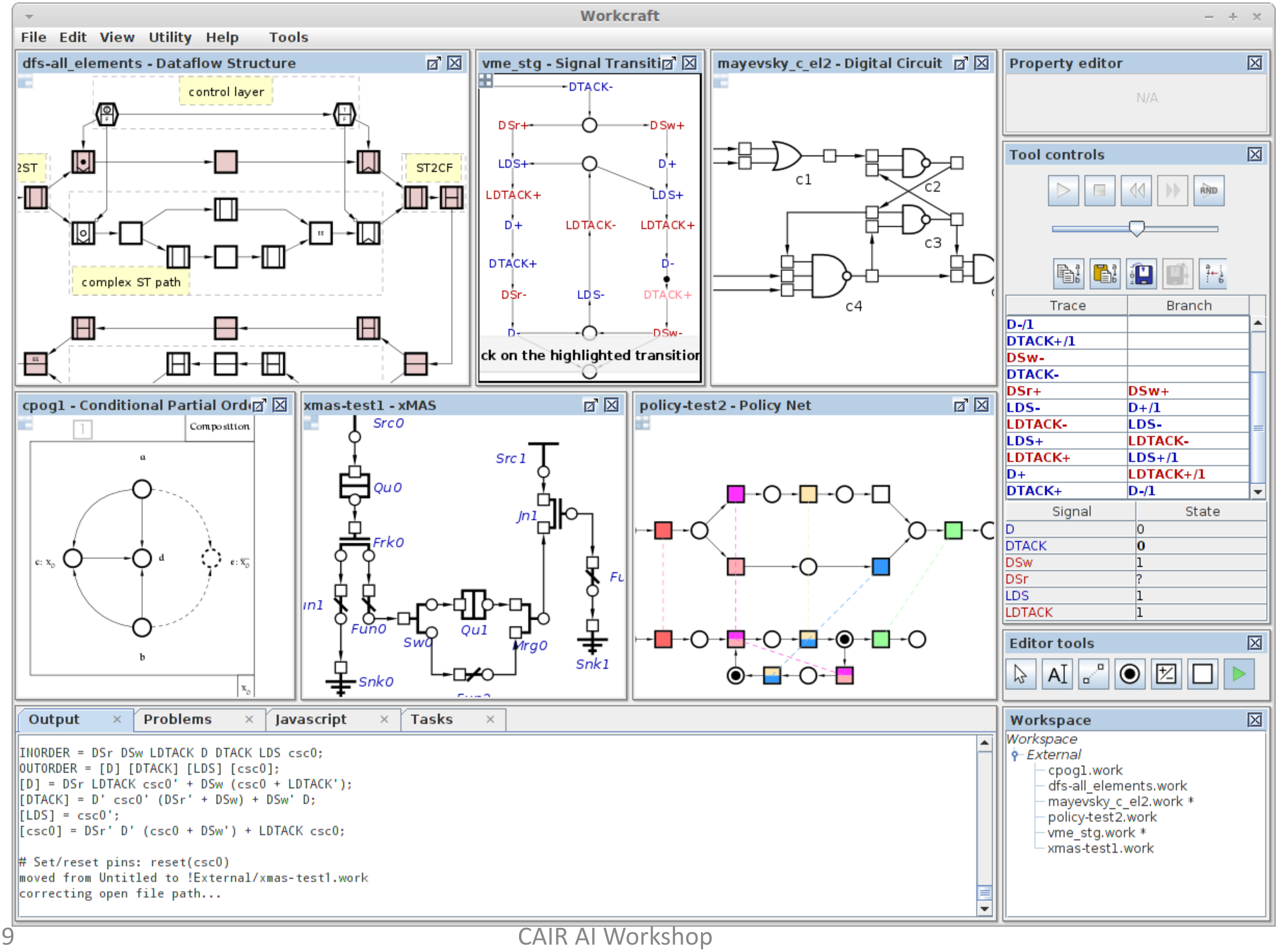
The circuit below adds 3 inputs multiplied by 3-bit weights (can be generated by Tsetlin Automata)



Robustness to Frequency and Voltage variations:



Workcraft: Toolkit for designing asynchronous circuits



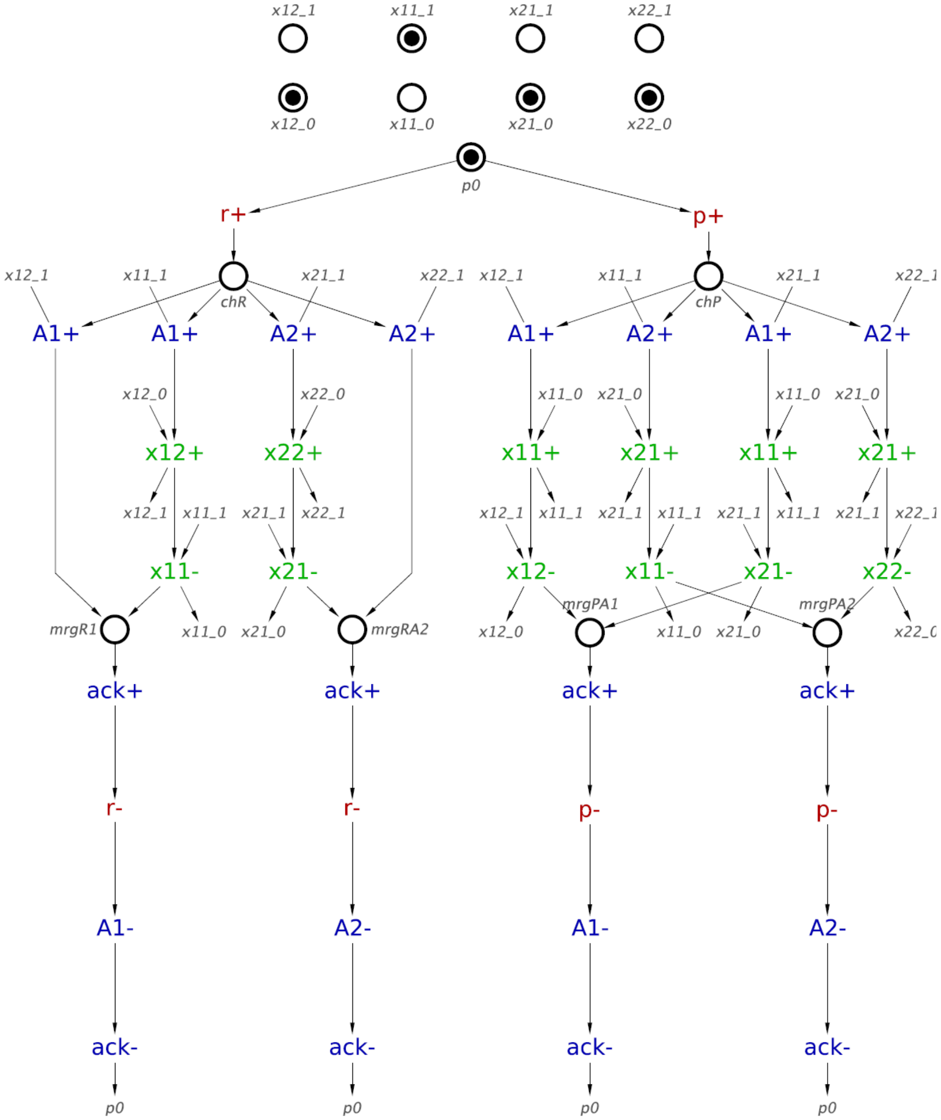
```
INORDER = DSr DSw LDTACK D DTACK LDS csc0;
OUTORDER = [D] [DTACK] [LDS] [csc0];
[D] = DSr LDTACK csc0' + DSw (csc0 + LDTACK');
[DTACK] = D' csc0' (DSr' + DSw) + DSw' D;
[LDS] = csc0';
[csc0] = DSr' D' (csc0 + DSw') + LDTACK csc0;

# Set/reset pins: reset(csc0)
moved from Untitled to !External/xmas-test1.work
correcting open file path...
```

Trace	Branch
D-/1	
DTACK+/1	
DSw-	
DTACK-	
DSr+	DSw+
LDS-	D+/1
LDTACK-	LDS-
LDS+	LDTACK-
LDTACK+	LDS+/1
D+	LDTACK+/1
DTACK+	D-/1

Signal	State
D	0
DTACK	0
DSw	1
DSr	?
LDS	1
LDTACK	1

Workspace
- External
  - cpog1.work
  - dfs-all_elements.work
  - mayevsky_c_el2.work *
  - policy-test2.work
  - vme_stg.work *
  - xmas-test1.work

Example STG for 4-state Tsetlin Automaton



Newcastle people involved in research on real-power computing, asynchronous design, approximate computing and pervasive AI

- Microsystems Group at Engineering:

Dr Alex Bystrov, Dr Andrey Mokhov, Dr Reza Ramezani, Dr Rishad Shafik, Dr Danil Sokolov, Dr Ahmed Soltan, Dr Fei Xia; PhD students: Sergey Mileiko, Thanasin Bunnam, Adrian Wheeldon

- Computing Science:

Dr Victor Khomenko, Prof Maciej Koutny

- Academic collaboration:

Prof Steve Furber's group in The Uni of Manchester

- Industry collaboration:

Temporal Computing and Dialog Semiconductor

Thank you!

More information:

https://www.ncl.ac.uk/engineering/research/eee/microsystems/

http://async.org.uk/

http://workcraft.org

https://blogs.ncl.ac.uk/alexyakovlev/