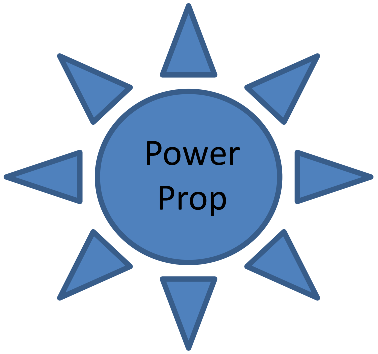
Run smarter
Live longer!

Putting computing on a strict diet with energy-proportionality

Alex Yakovlev, microSystems,
School of EEE, Newcastle University
async.org.uk
www.ncl.ac.uk/eee/research/groups/micro/

XXIX DCIS, Madrid
26th November 2014

Energy drives logic


Power
Prop

The more you get
The more you give!

Outline

- **Resource-driven computing**
  - What is Energy-modulated computing?
- **Understanding Real-Power**
  - What is a computational load for energy source? What is energy effort? What is computational output?
- **Power-proportionality**
  - How to compute from a wide range of power levels? How to optimize computational activity for energy budget? What is power-adaptation?
- **Designing for power-proportionality**
  - How to design for multiple modes in one core? How to achieve graceful degradation for power levels? What models can be used for multimodal and reconfigurable microarchitectures?
- **Looking into the future**

Resource-driven Computing

Interplay

Working conditions:

Energy-constrained systems

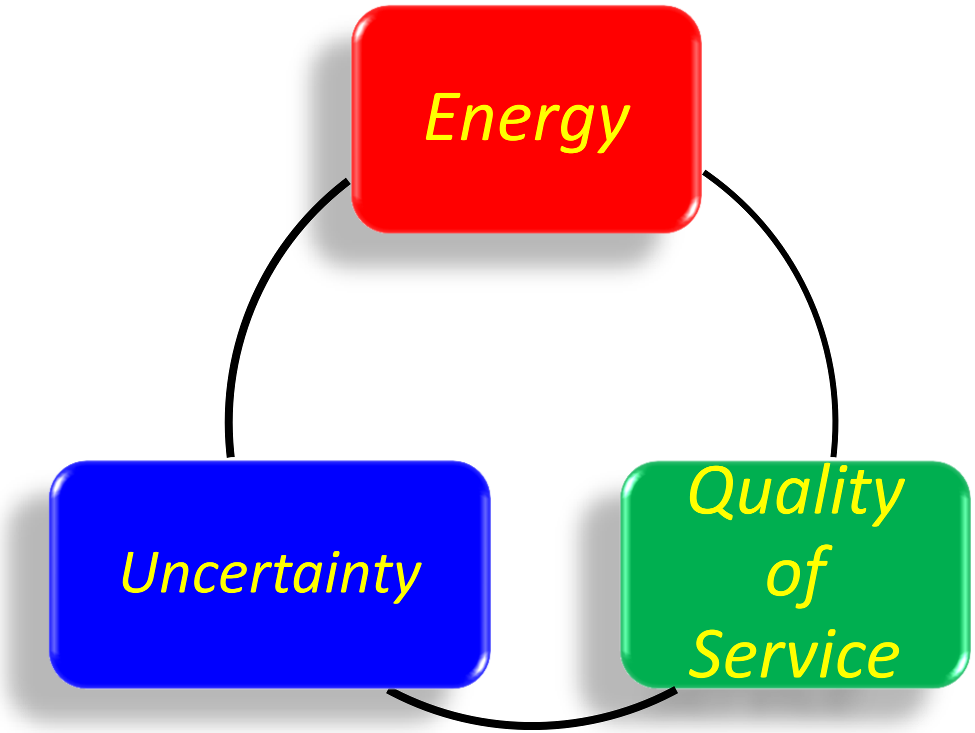
- **Solar energy, e-beam power supply, small batteries, …**

Unreliable power supply

- **Voltage fluctuations, low battery, …**

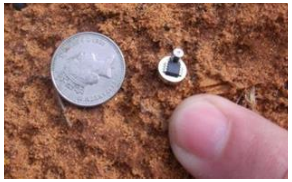
Hostile environments

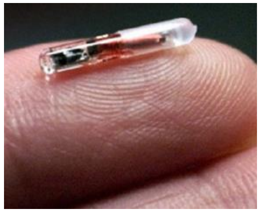
- **High/low temperatures, noise, …**

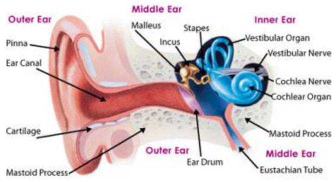

Energy

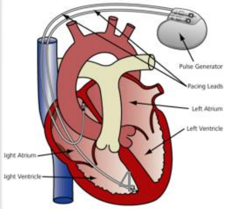
Uncertainty

Quality of Service





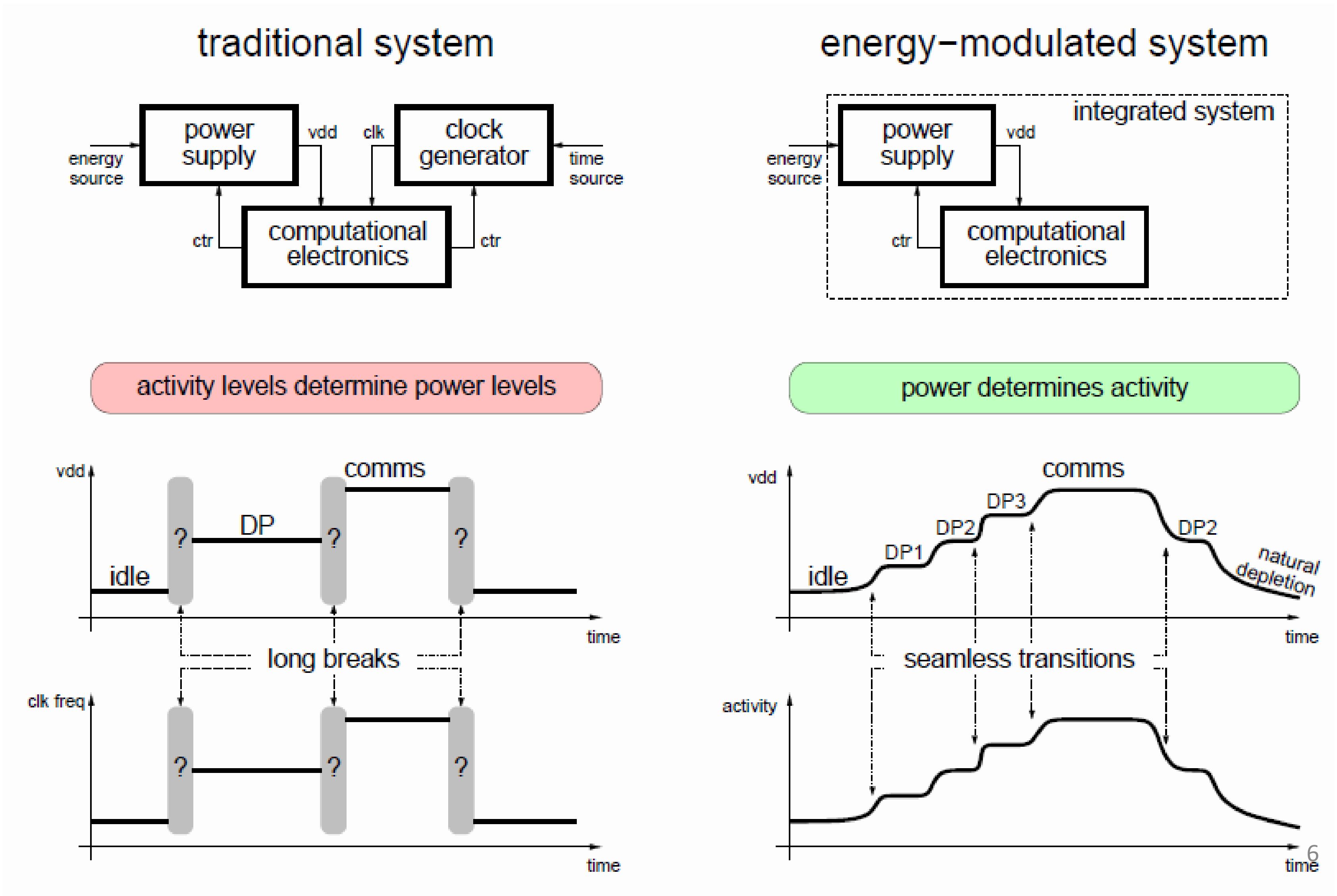




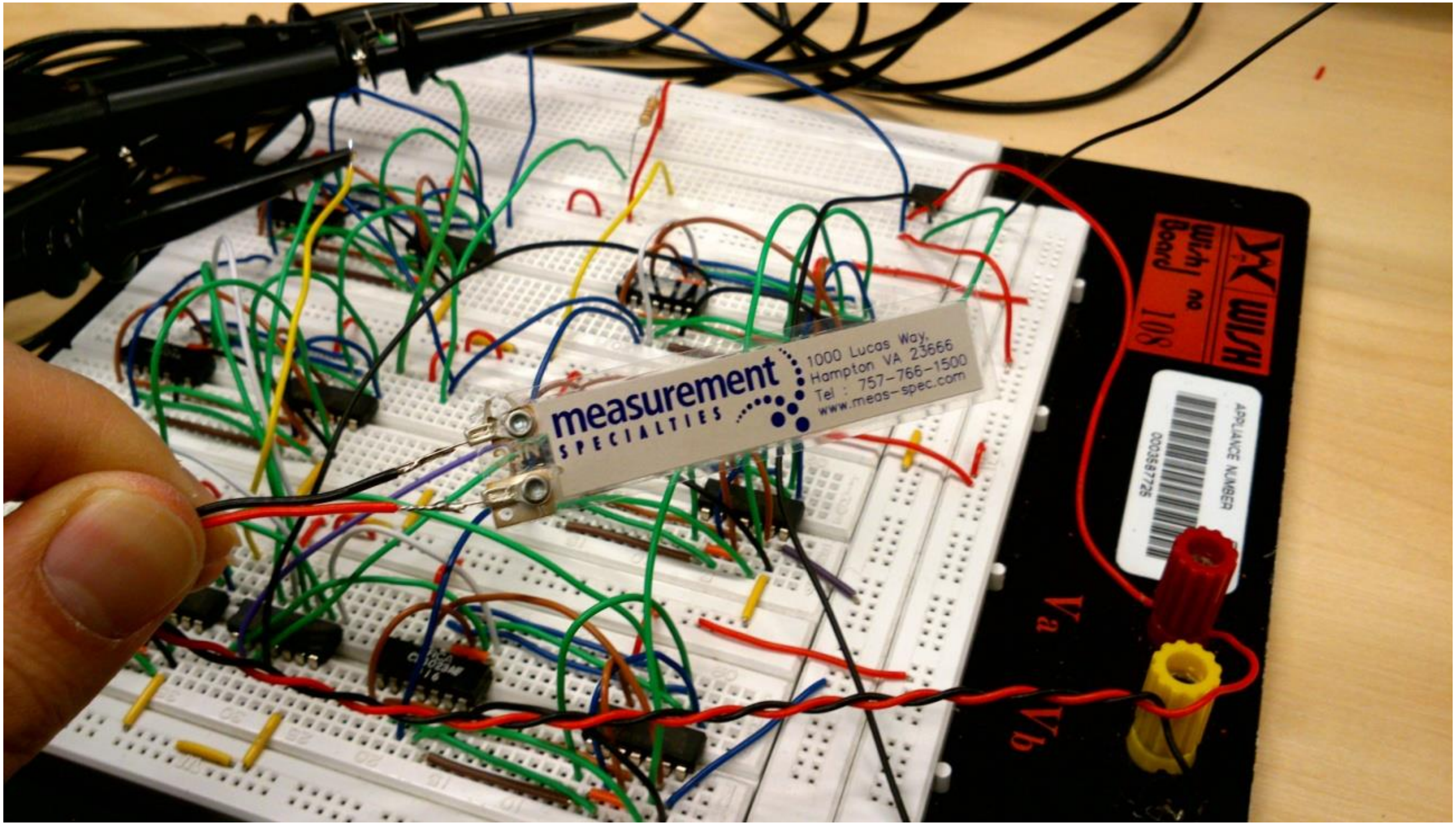
Power/Energy modulation

- The principle of **power/energy-modulated computing** is that the flow of energy entering a computing system determines its computational flow
- It is fundamental for building future **real-power** systems, particularly systems for **survivability**
- Any piece of electronics becomes active and performs to a certain level of its delivered **quality in response to some level of energy and power**
- A **quantum of energy** when applied to a computational device can **be converted** into a corresponding **amount of computation activity**
- Depending on their design and implementation systems can produce **meaningful activity at different power levels**
- As **power levels become uncertain** we cannot always guarantee completely certain computational activity

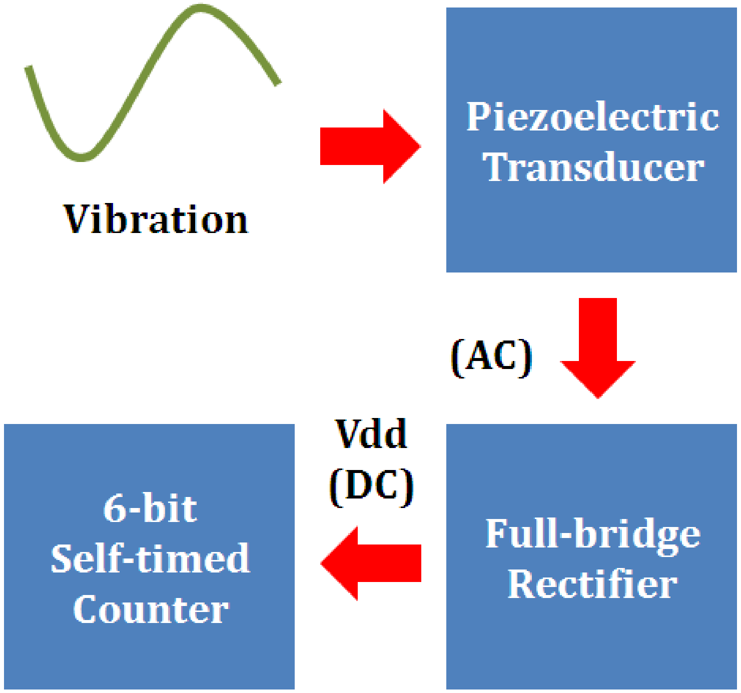
Traditional vs energy-modulated view



Piezo-Film Experiment



Experiment Results



Vibration

Piezoelectric Transducer

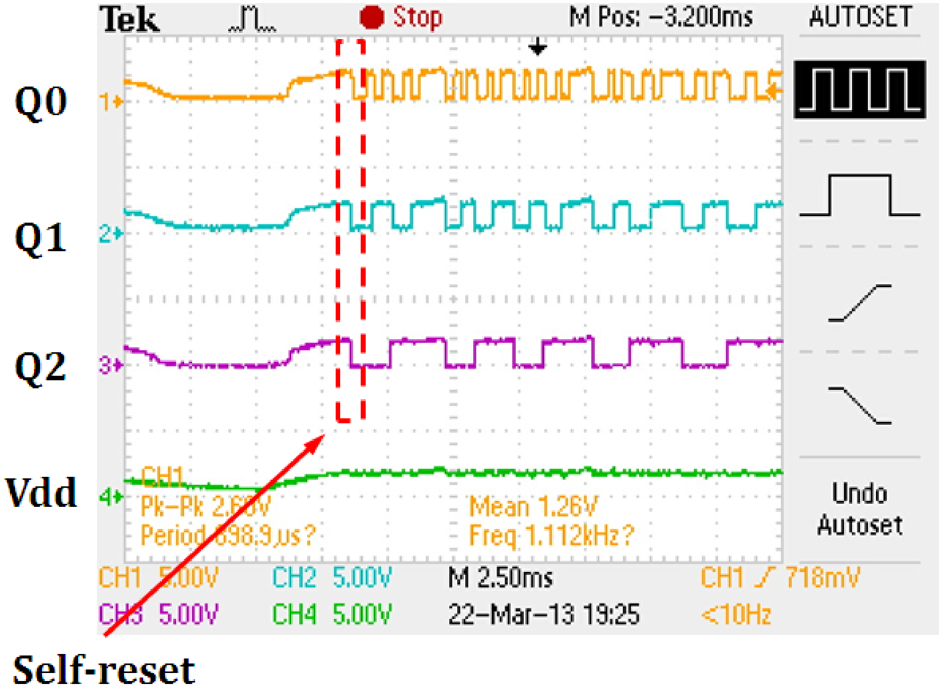
(AC)

Full-bridge Rectifier

Vdd (DC)

6-bit Self-timed Counter

(A)



Tek Stop M Pos: −3.200ms AUTOSET

CH1
Pk−Pk 2.60V
Period 898.9μs ?

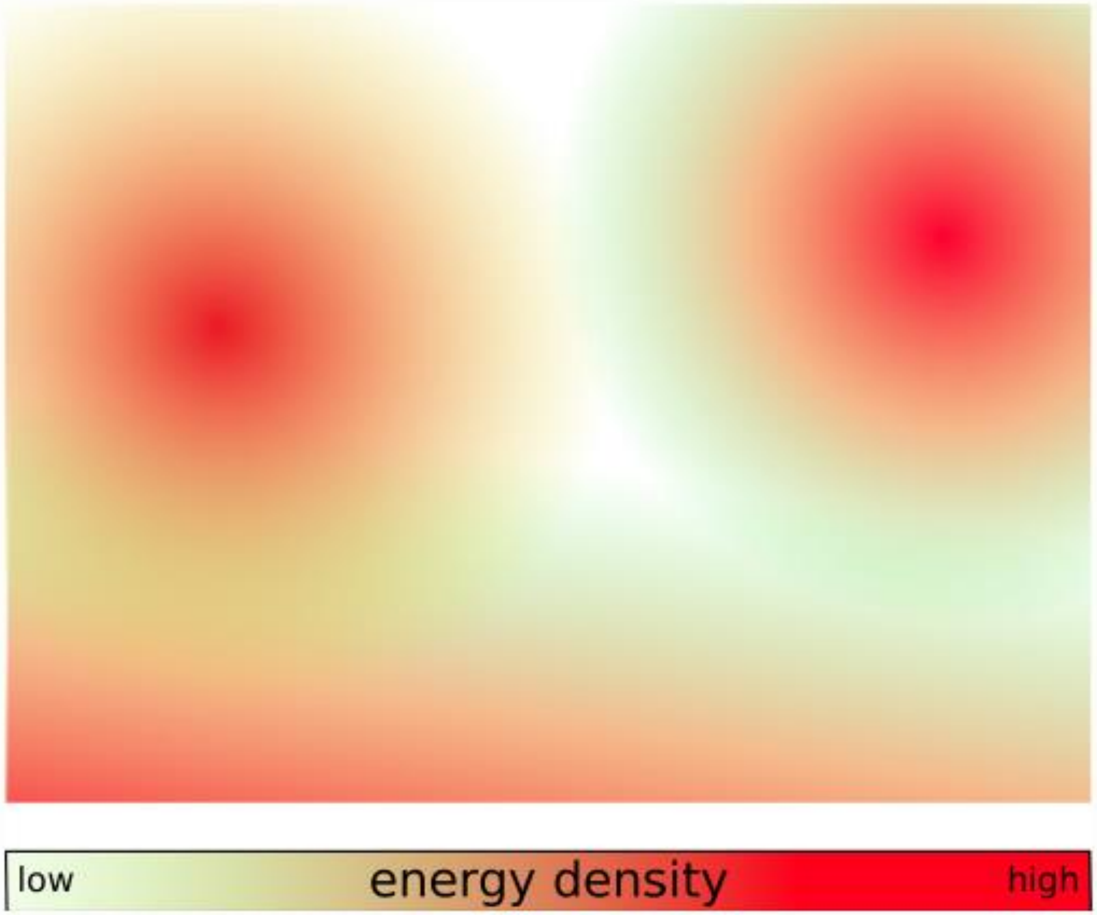
Mean 1.26V
Freq 1.112kHz ?

CH1 5.00V CH2 5.00V M 2.50ms CH1 ⟋ 718mV
CH3 5.00V CH4 5.00V 22−Mar−13 19:25 <10Hz

(B)

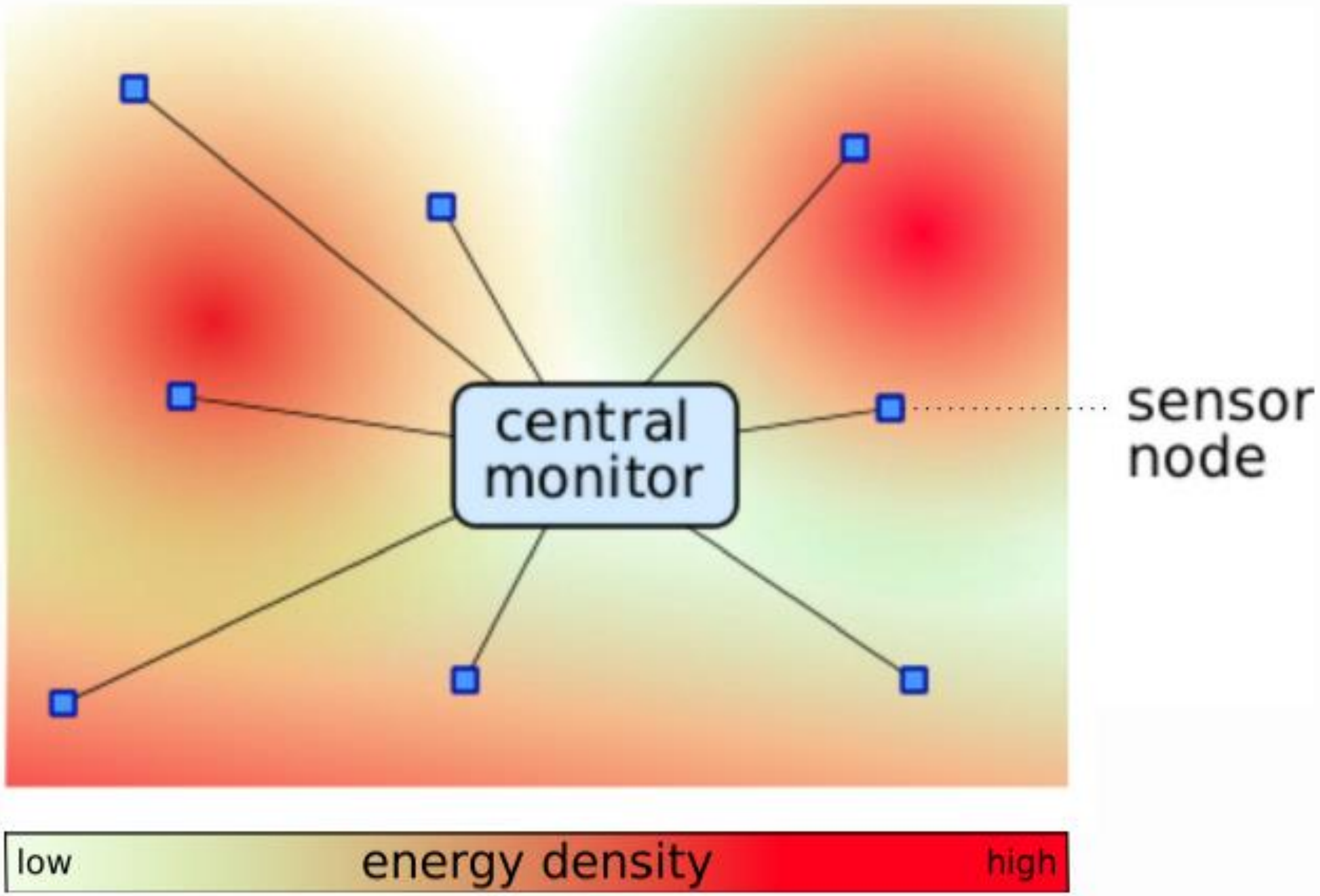
Example: Walls Alive (condition monitoring)

- Energy field (thermal, mechanical vibration, etc)



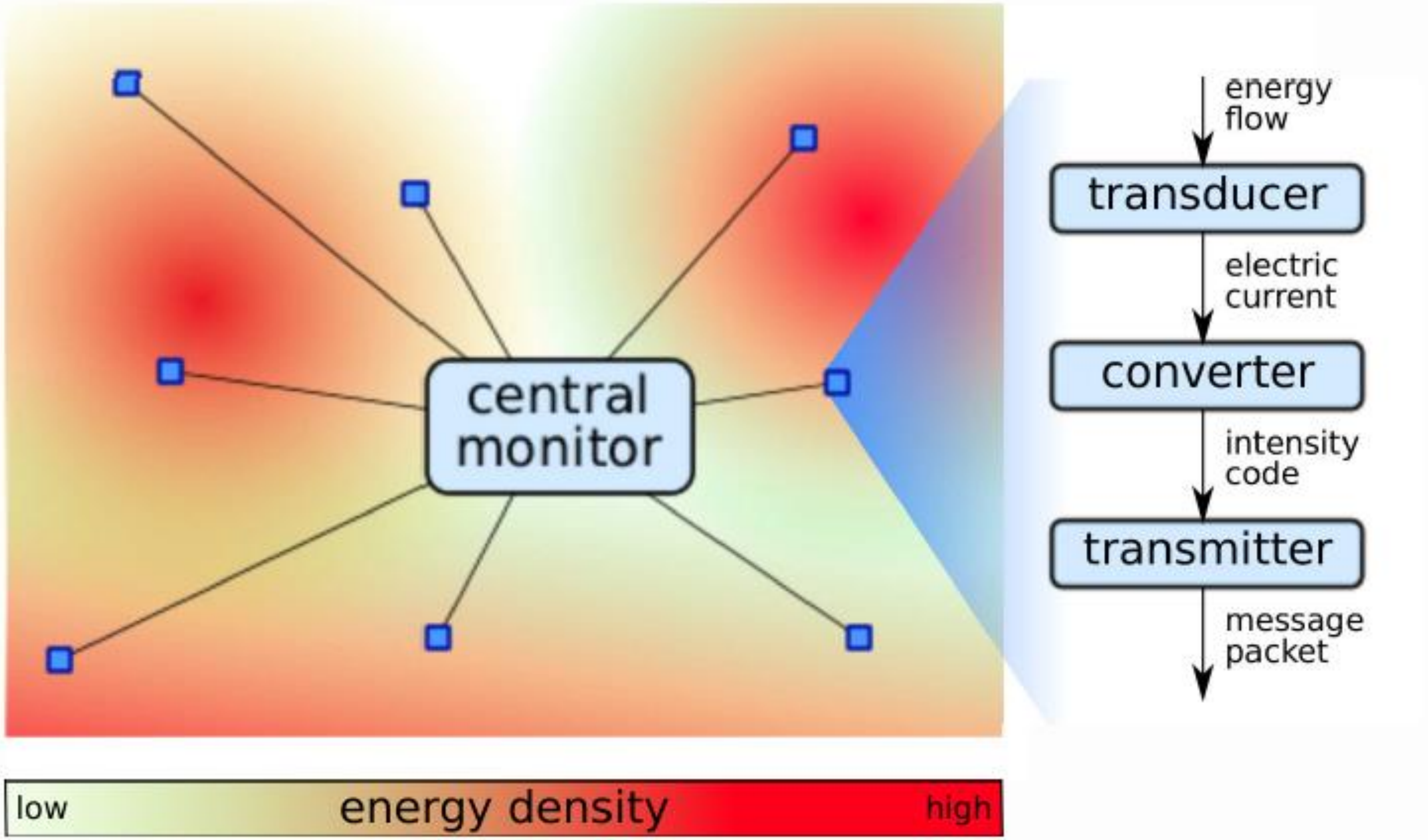
Example: condition monitoring

- Network of sensors for spaced and temporal energy mapping

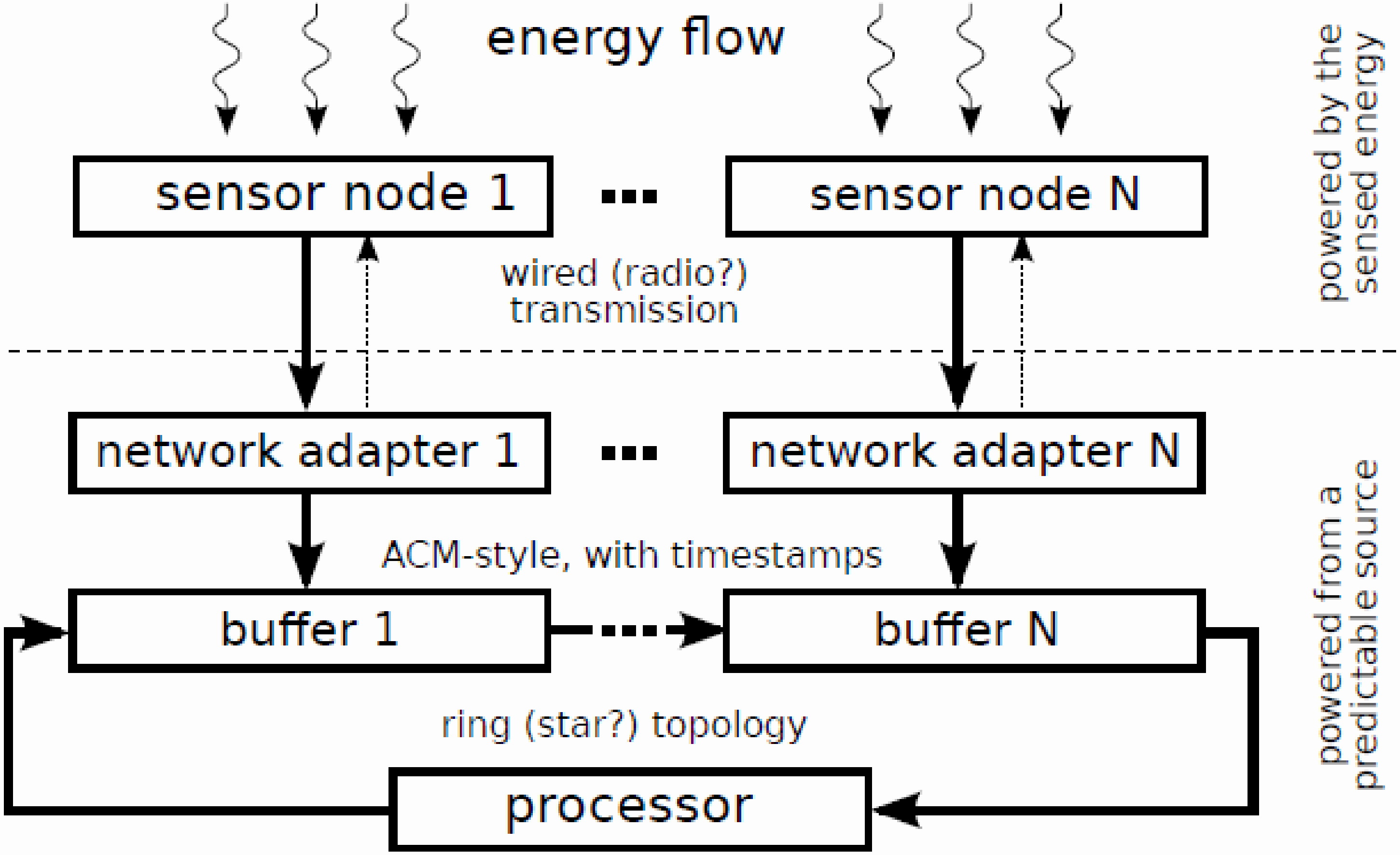


Example: condition monitoring

- Sensor node structure

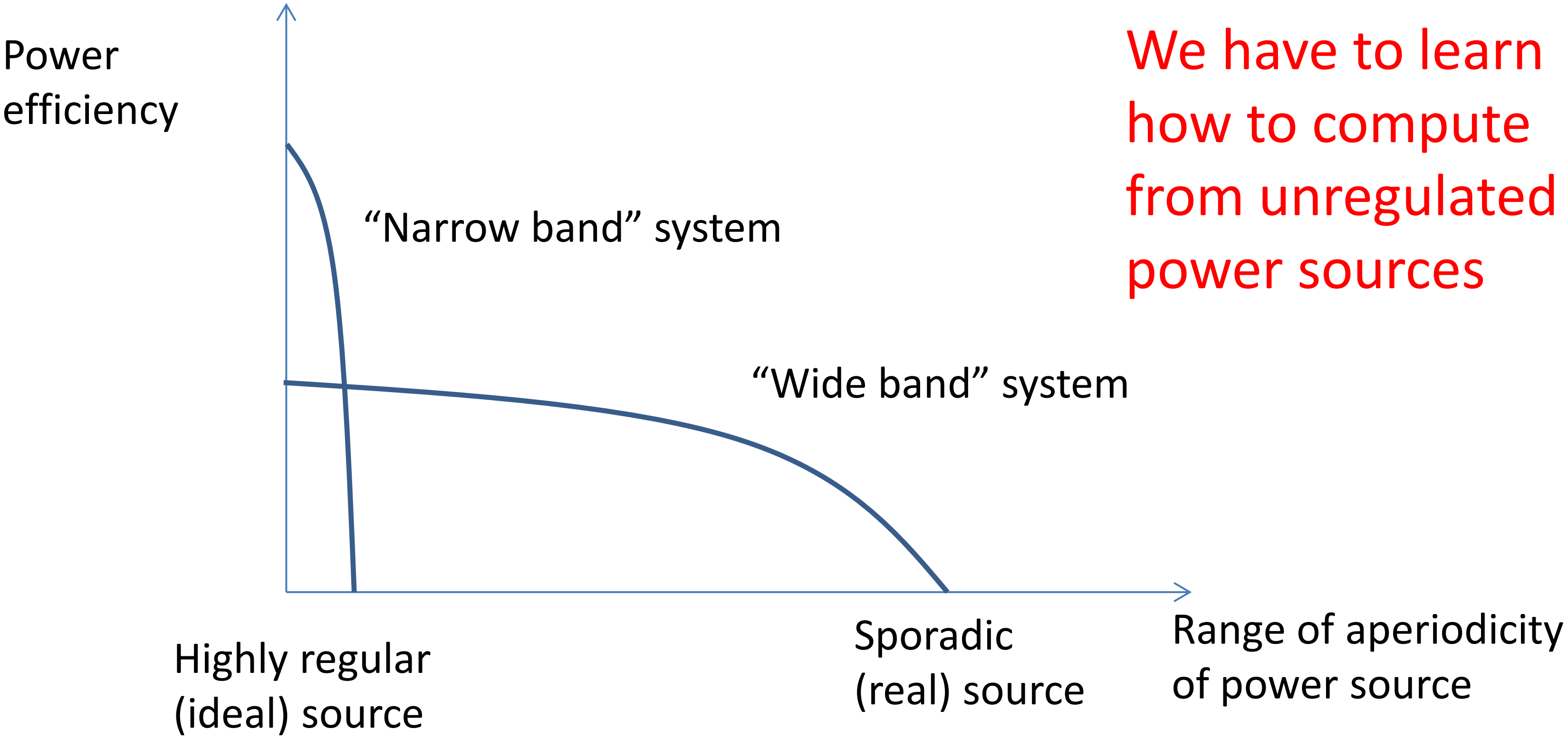


- System architecture



Power efficiency and regularity

- Modern systems rely on highly regular (periodic) power sources – they “invest” some power into power regulation
- Future systems will have to operate in a wide dynamic range, paying the price in efficiency in a particular band

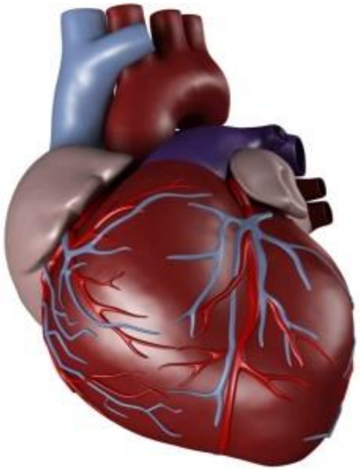


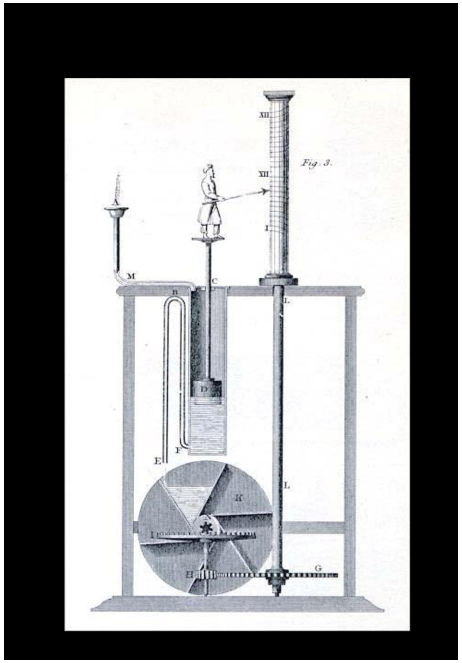
We have to learn how to compute from unregulated power sources

Towards Real-Power systems

- **How to design Real Power systems?**
  - Firstly, understanding real-power
  - Secondly, developing design principles for real-power
- **Understanding Real Power:**
  - What is computational load? Is it a resistor or distributed capacitor? A bit of both? What is the dynamics of the process of discharging a capacitor by a switching circuit?
  - What is energy effort at the logic level? How much computation action does a circuit perform for a given amount of energy?

Understanding Real-Power Systems

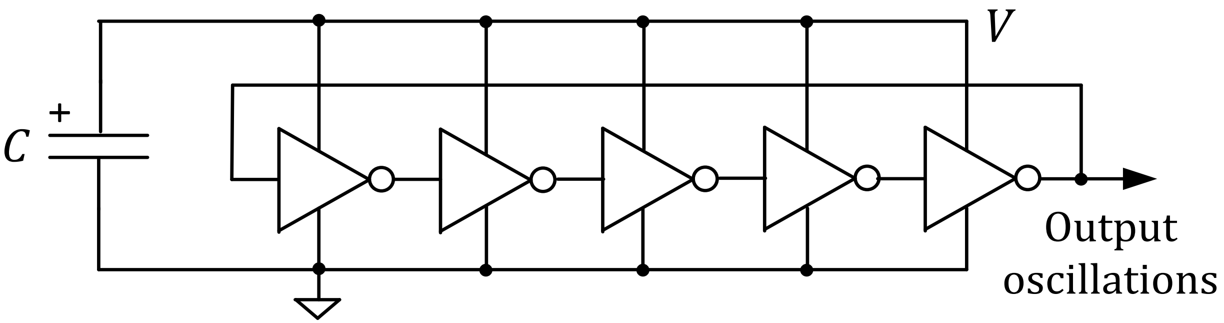




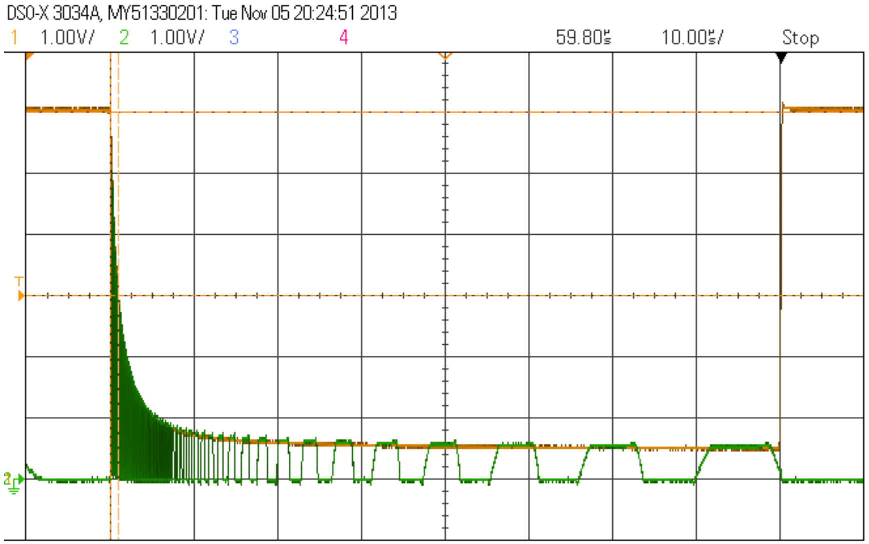
Ctesibius' Clepsydra , 3rd century BC

Faraday's homopolar motor, 1821

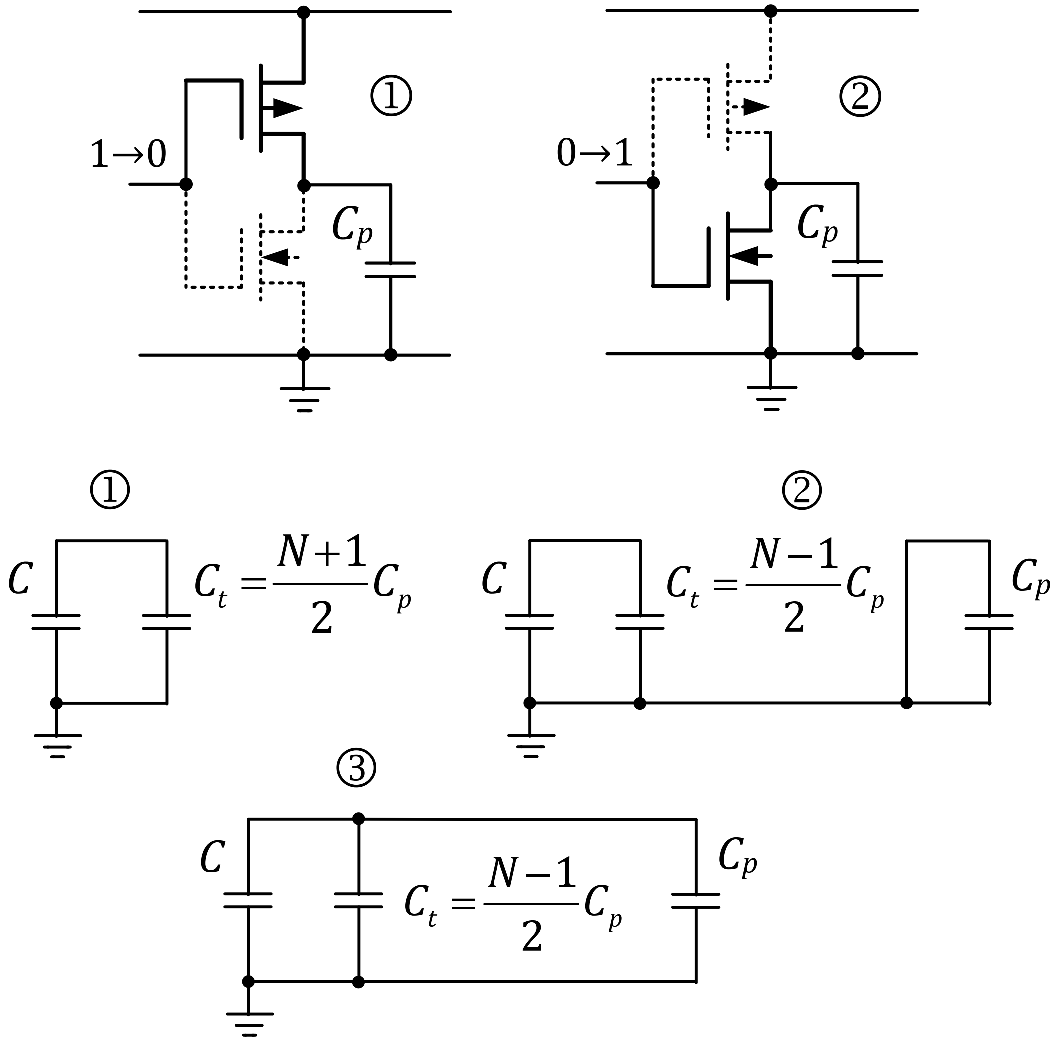
What is Computational Load?



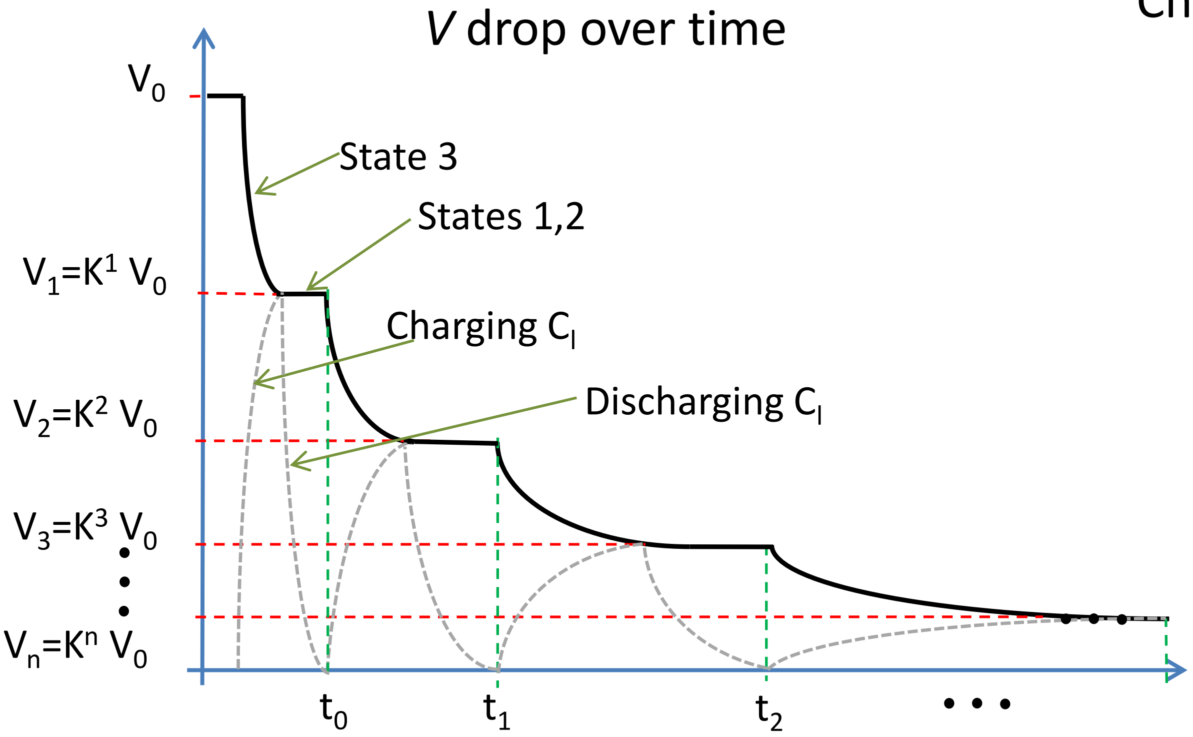
- We employ a simple **ring-oscillator** to serve as a digital circuit load.
- It is due to the fact that ring-oscillator can closely mimic the switching behaviour of many **closed loop self-timed circuits**.



Circuit model


①
1→0
C_p
②
0→1
C_p
①
C
$C_t = \frac{N+1}{2} C_p$
②
C
$C_t = \frac{N-1}{2} C_p$
C_p
③
C
$C_t = \frac{N-1}{2} C_p$
C_p

Circuit Model: switching process



Charge equilibrium at:

$$V_1 = V_0 \frac{C}{C + C_l}$$

$$K = \frac{C}{C + C_l}$$

$$V = K^n V_0$$

$$V_N = K^n$$

Solution for Super-threshold

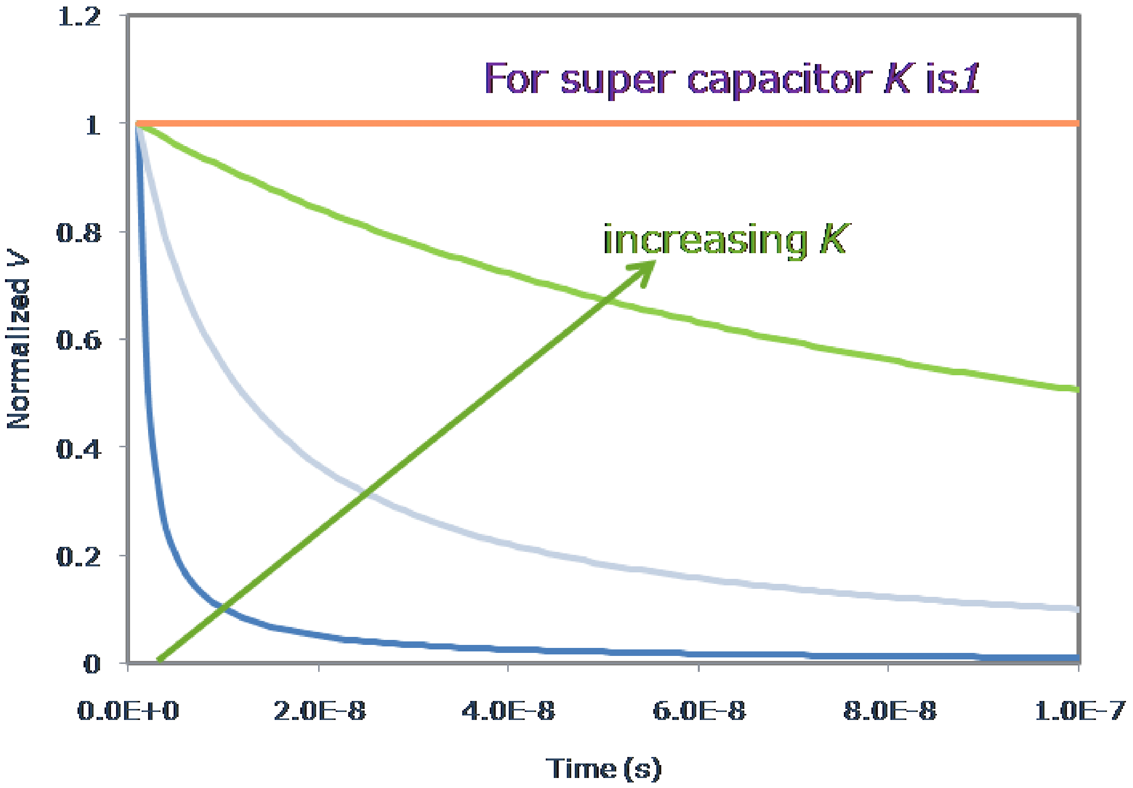
A valid assumption: in super-threshold region we can assume that the propagation delay is inversely proportional to the voltage, so we have:

Switching index	V_N	$t_s = \frac{A}{V}$	Physical time (*t*)
0	K^0	$\frac{A}{K^0}$	$\frac{A}{K^0}$
1	K^1	$\frac{A}{K^1}$	$\frac{A}{K^0}+\frac{A}{K^1}$
2	K^2	$\frac{A}{K^2}$	$\frac{A}{K^0}+\frac{A}{K^1}+\frac{A}{K^2}$
…			
n	K^n	$\frac{A}{K^n}$	$\sum_{i=0}^{n}\frac{A}{K^i}$

Solution for Super-threshold

$$V_N = \frac{AK}{t(1-K)+AK}$$

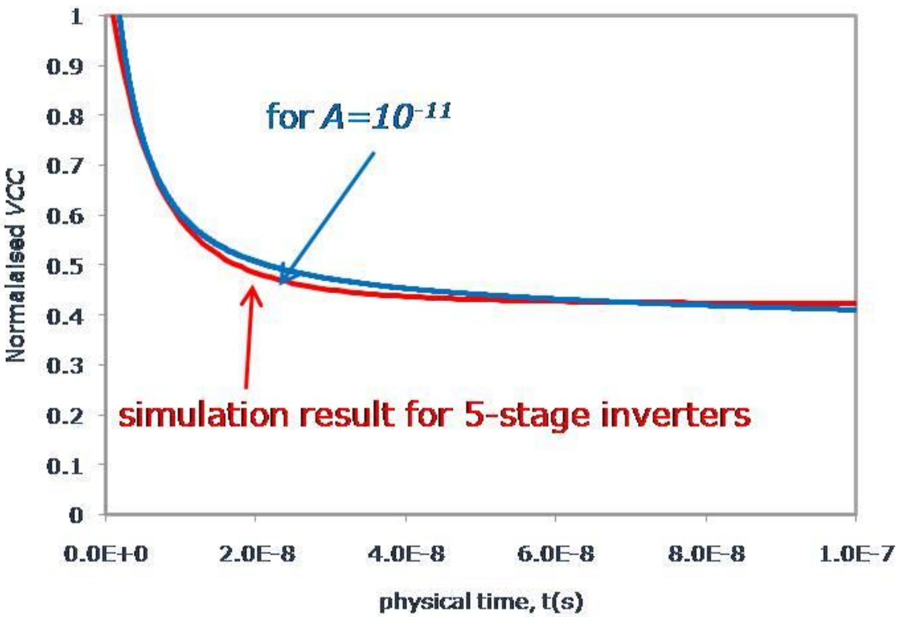
Hyperbolic function of time



More accurate solution for Super-threshold

A general model of gate delay propagation [1] is used:

$$t_p = \begin{cases} t_{p1} = \dfrac{pc_l V}{(V - V_{TH})^{\alpha}} \\ t_{p2} = \dfrac{pc_l V}{I_0 e^{\frac{V - V_{TH}}{N_s}}} \end{cases}$$



Assuming $\alpha = 1.3$

$$\int_0^n \frac{AK^i}{(K^i - V_{THN})^{\alpha}} di = V_{THN} + \left(\frac{-\frac{10}{3}A}{\ln K \cdot (t - B \cdot A)}\right)^{3.33}, B = \frac{\frac{10}{3}}{\ln K \cdot (1 - V_{THN})^{0.3}}$$

[1] “Sub-threshold Design for Ultra Low Power Systems”, A.Wang, B. H. Calhoun, A. P. Chandrakasan

Analysis of hyperbolic decays

- For super-threshold and α=2:

$$V_N = \frac{AK}{(1-K)t + AK} = \frac{1}{at+1}$$

where $A = 2pC_{_p}$ $and\, a = \frac{1-K}{AK}$

- For arbitrary α:

$V_N = \frac{1}{(at+1)^{1/(\alpha-1)}}$ and $a = \frac{1-K^{\alpha-1}}{AK^{\alpha-1}}$

- Differential equation:

$$\frac{dV}{dt} = -aV^{\alpha}$$

The oscillator is a voltage (and time)-varying resistor:

$R(V) = \frac{2p}{V}$ or $R(t) = 2p(at+1)$

Analysis of hyperbolic decay rates

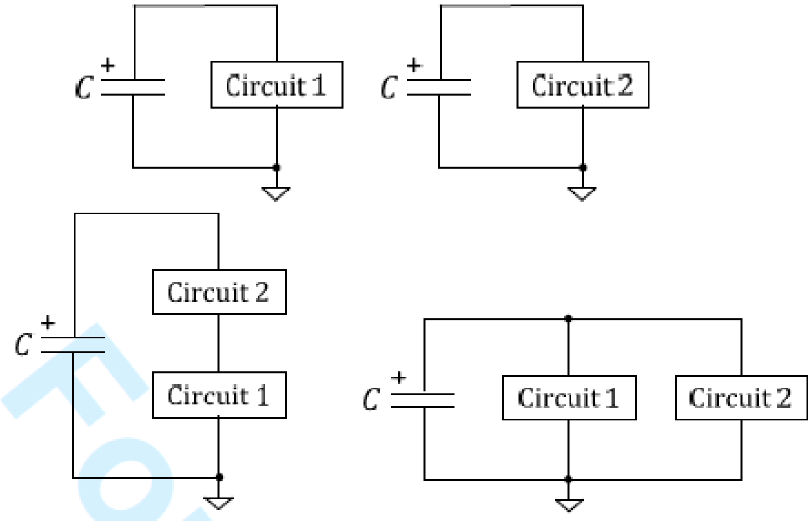
- For stack:

$$\left(\frac{1}{a}\right)^{\frac{1}{\alpha}} = \left(\frac{1}{a_1}\right)^{\frac{1}{\alpha}} + \left(\frac{1}{a_2}\right)^{\frac{1}{\alpha}} \text{ or}$$

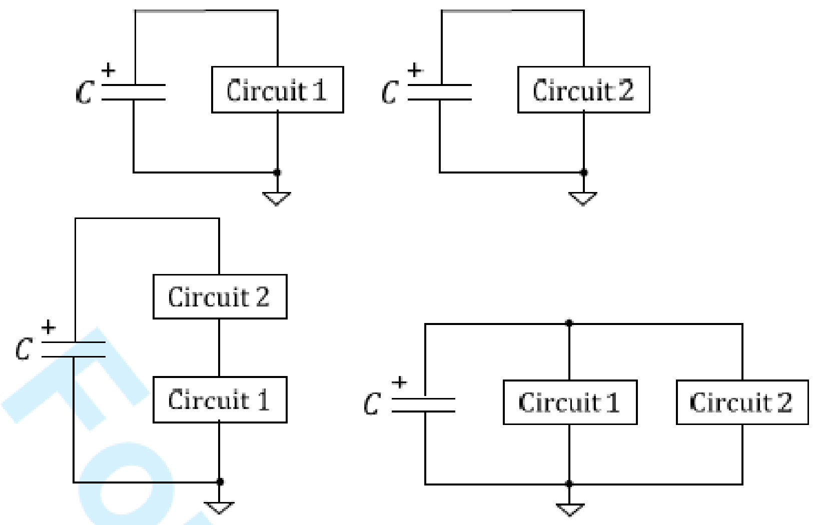
$$a = \frac{a_1 a_2}{\left(\sqrt[\alpha]{a_1} + \sqrt[\alpha]{a_2}\right)^{\alpha}}$$

- For parallel: $a = a_1 + a_2$
- Confirmed by physical experiments with discrete CMOS components:
  - ***The value of alpha is 1.5***
  - ***The discharging process for a stack of two identical circuits is nearly 3 times slower than for a standalone circuit***

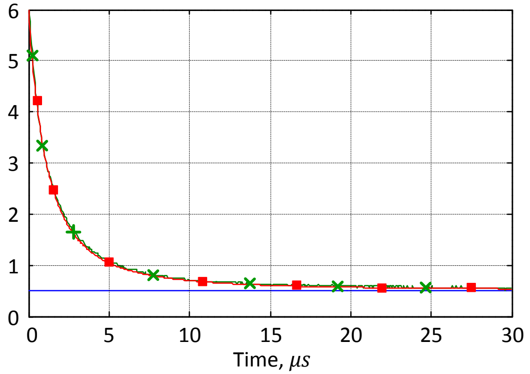
Series (stack) and parallel configurations:



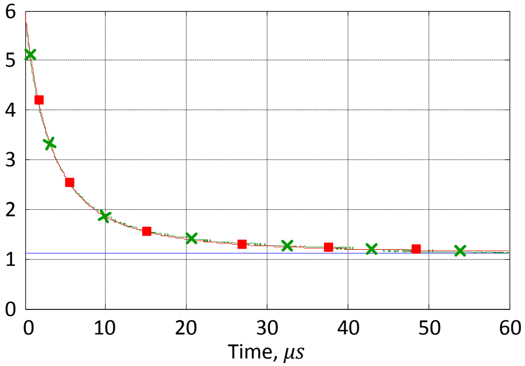
Capacitor discharge experiments



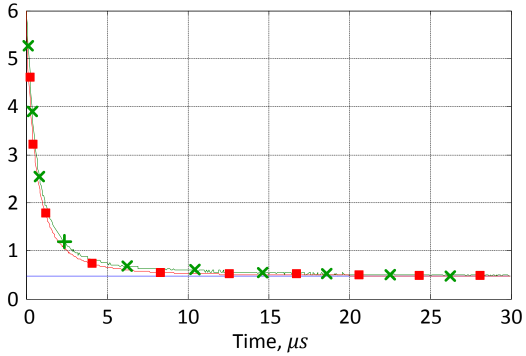
Standalone



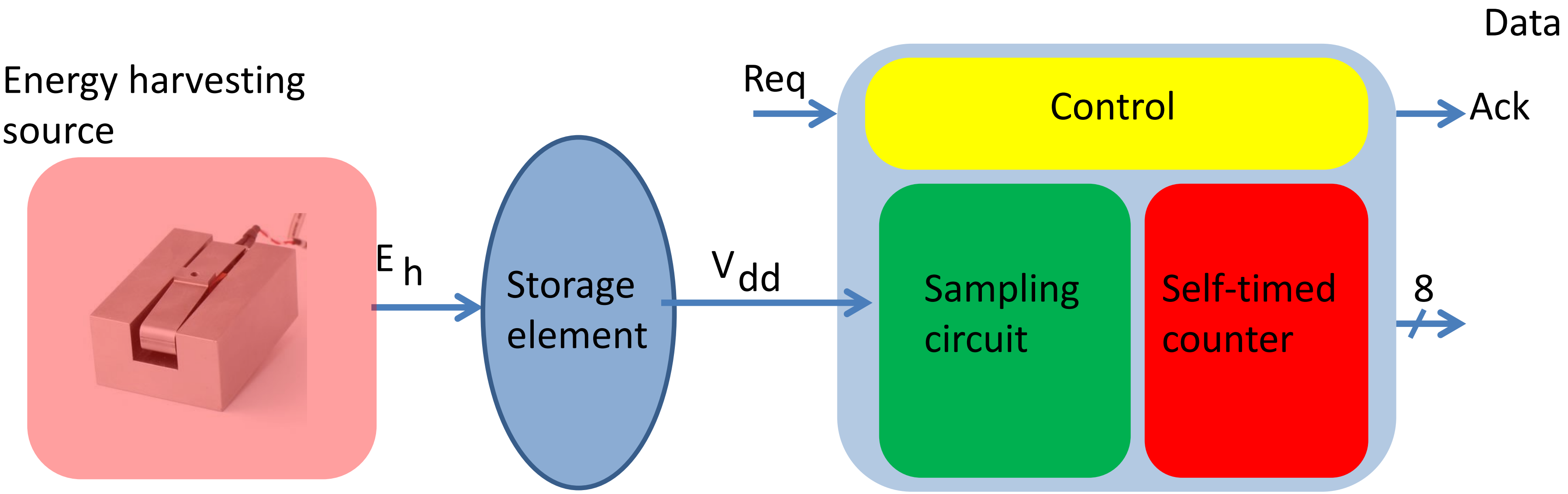
Stack



Parallel

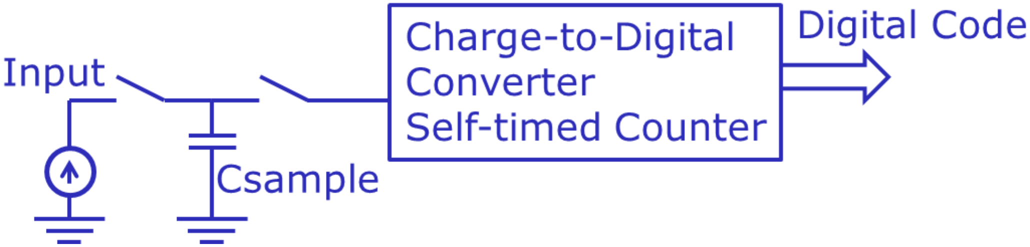


Reference-free voltage sensor



Reference-free voltage sensing

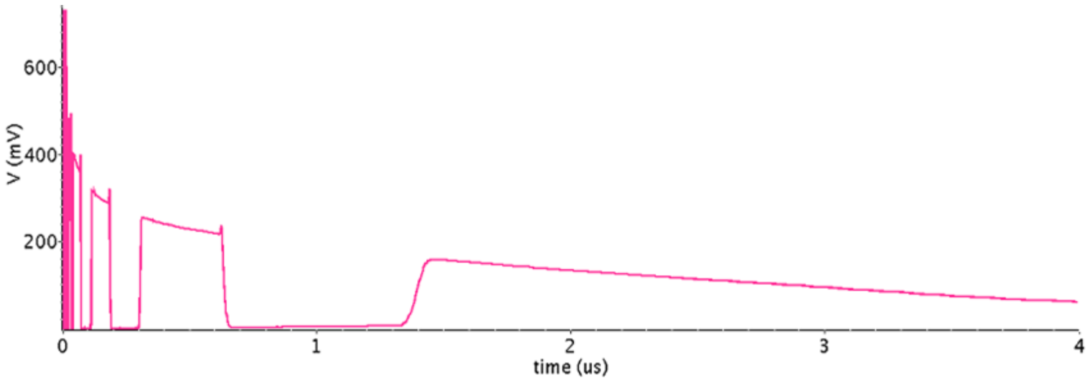
- Voltage sensor requiring only timing reference

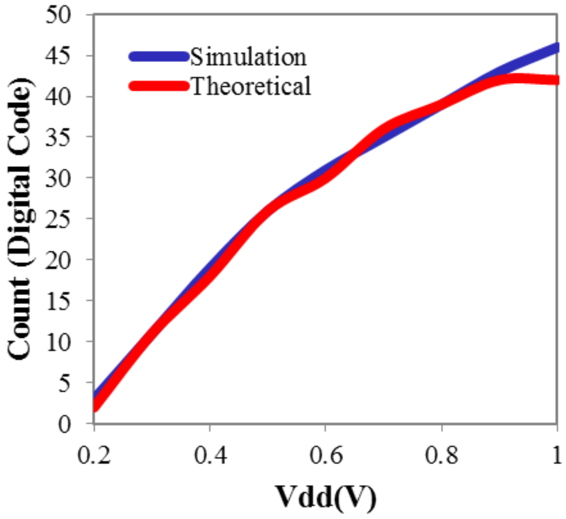

Input

Csample

Charge-to-Digital Converter Self-timed Counter

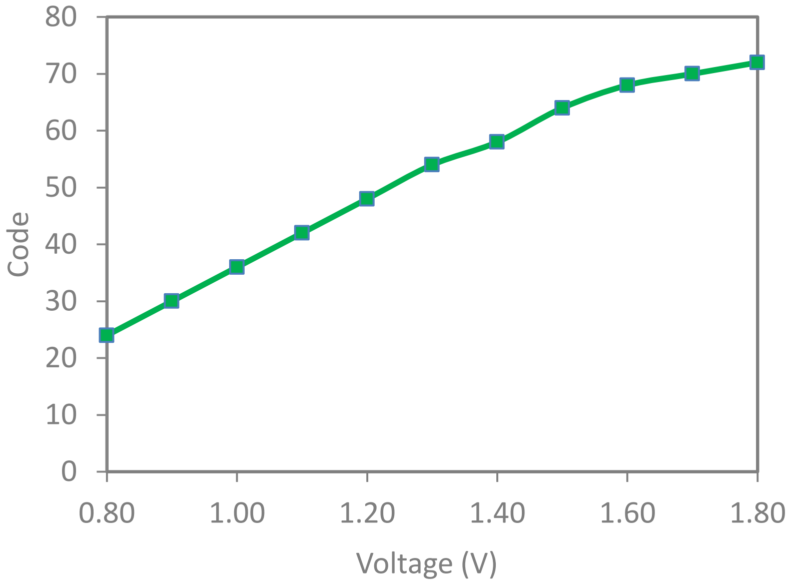
Digital Code

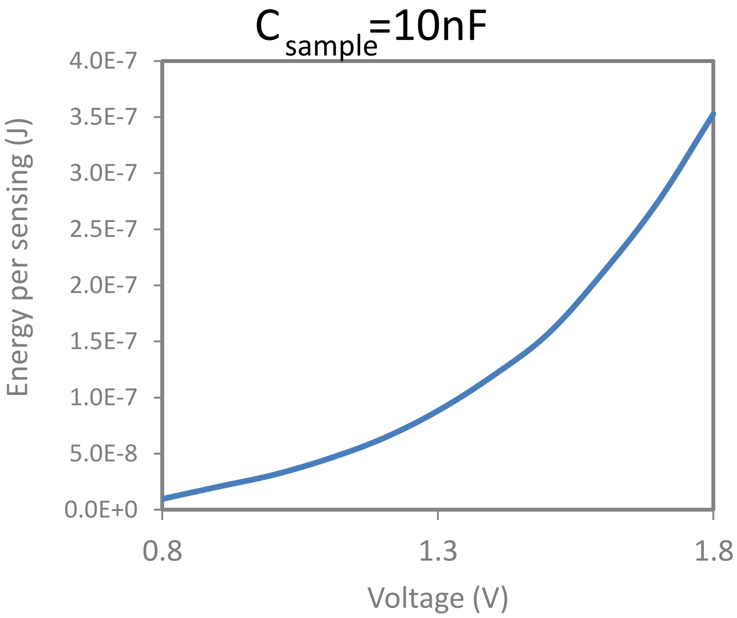




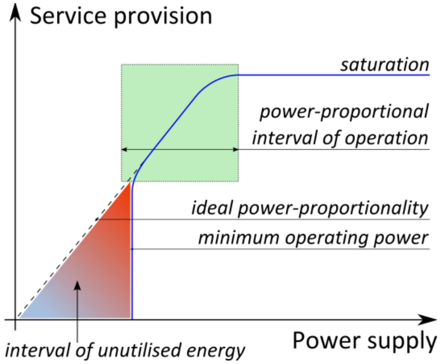
Apparatus and method for voltage sensing, Newcastle University, GB Patent Number 2479156, 30 March 2010.

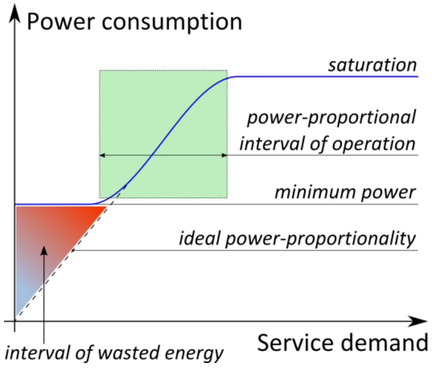
Output count and energy consumption



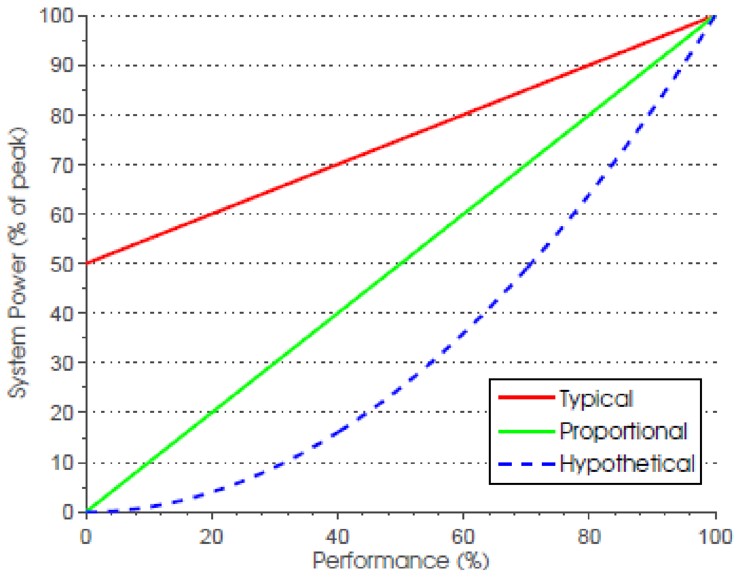


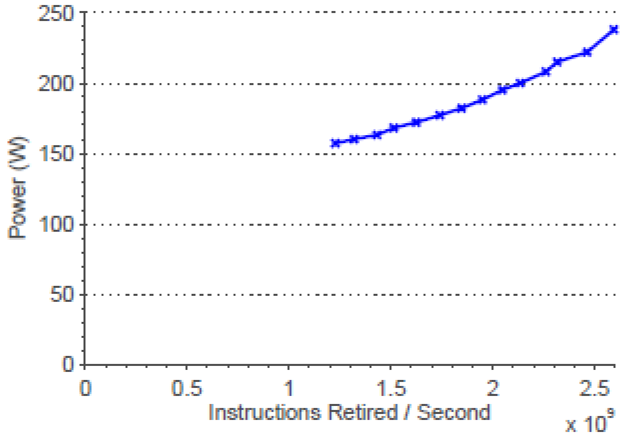
Power Proportionality





Power Proportionality





Core i7 power drawn at different frequencies

Issues reported in literature:

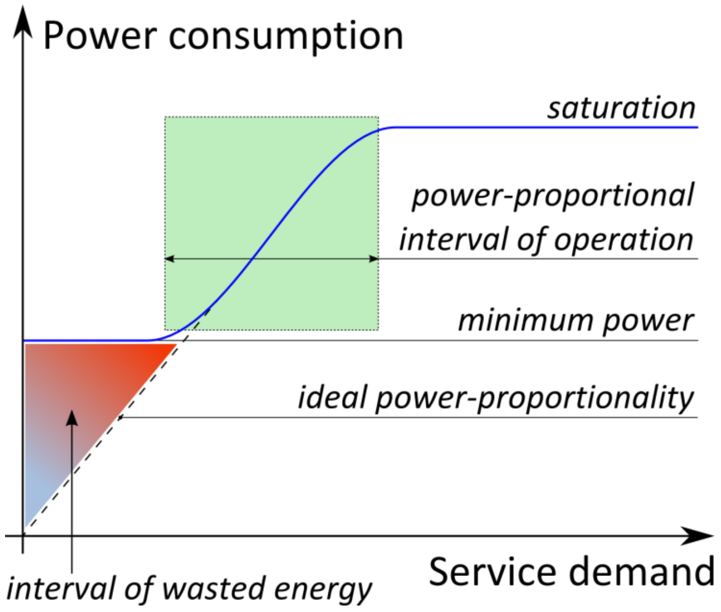
- Performance-power tradeoff for commodity systems is linear; the best strategy is “Race to sleep”; additional “run” power states are of little use; changes in existing commodity operating systems have little influence
- The focus should be on the time to transition to and from sleep!
- For a new type of systems such as WSN there is a non-linear region – the slogan is: learn how to run CMOS slowly and exploit scheduling optimizations

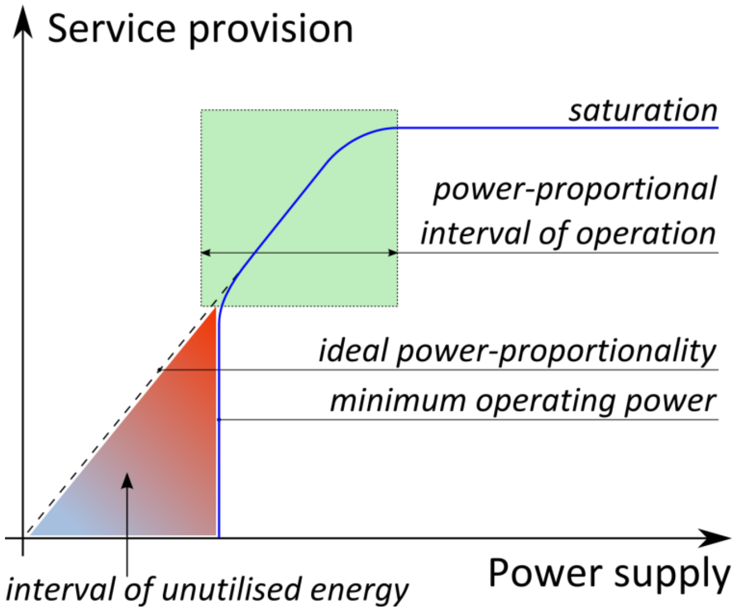
Source: S. Dawson-Haggerty et al. Power Optimization – Reality Check, UC Berkeley, 2009

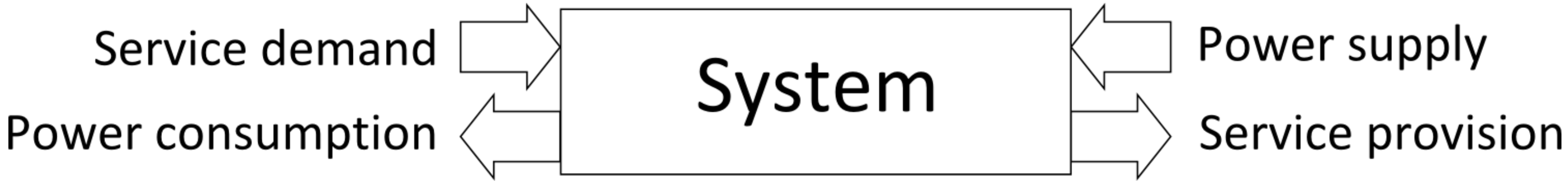
Power proportionality ("knee-stretching")

Service-modulated processing

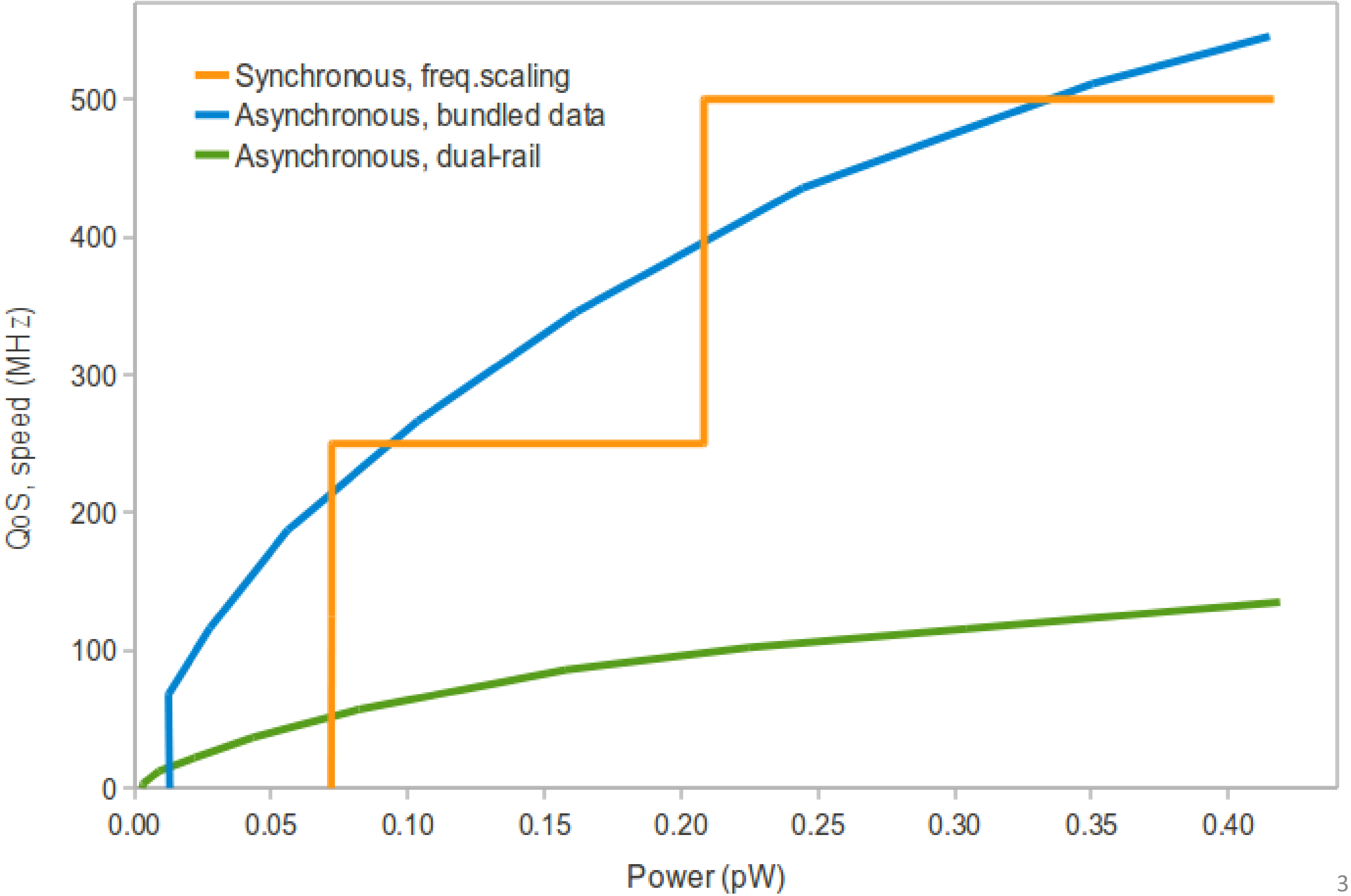
Energy-modulated processing



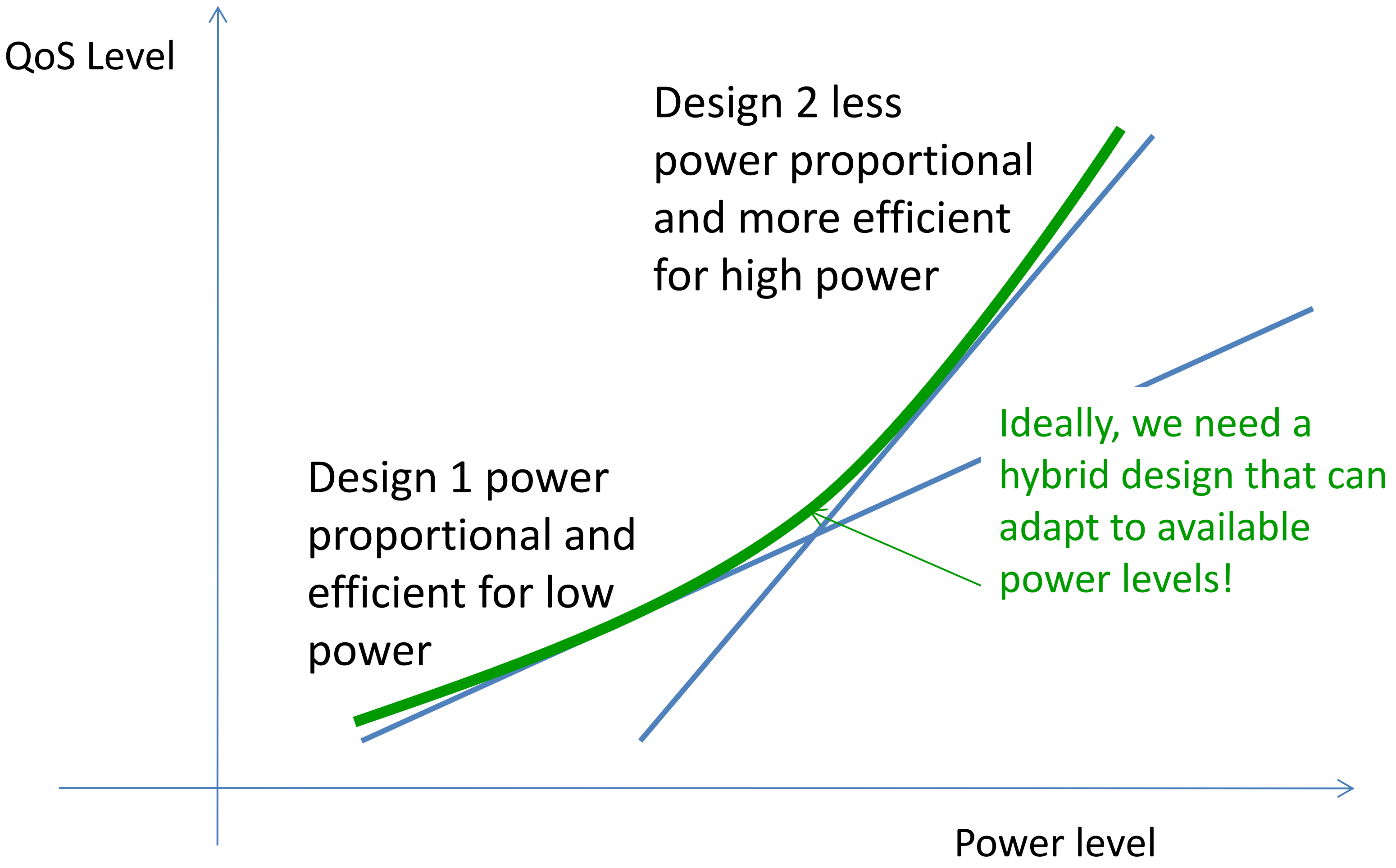




Multiplier: Quality of Service vs Power



From power-proportional to power-adaptive

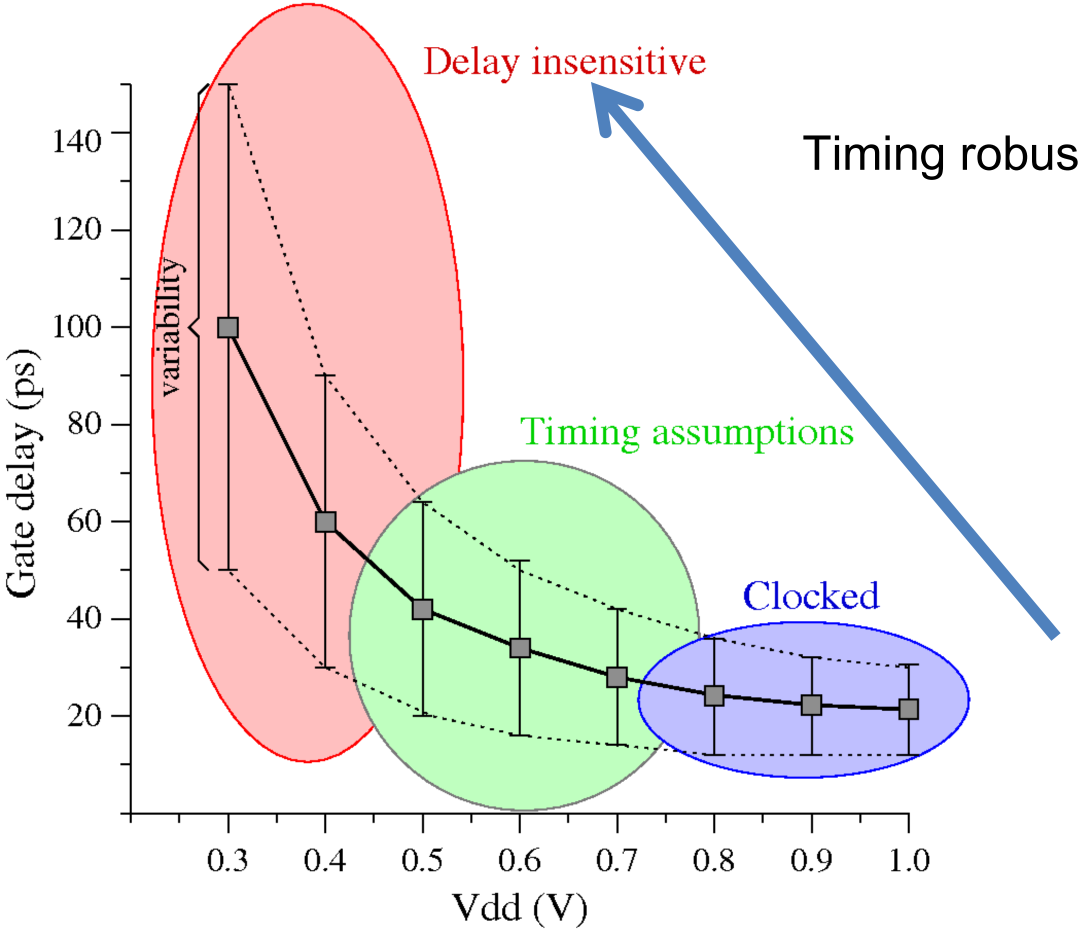


Design 2 less power proportional and more efficient for high power

Design 1 power proportional and efficient for low power

Ideally, we need a hybrid design that can adapt to available power levels!

Relationship with uncertainty (e.g. timing variability)



Timing robustness

Technology node:
90nm

Source of variability analysis:
Yu Cao, Clark, L.T., 2007

Towards designing power-adaptive systems

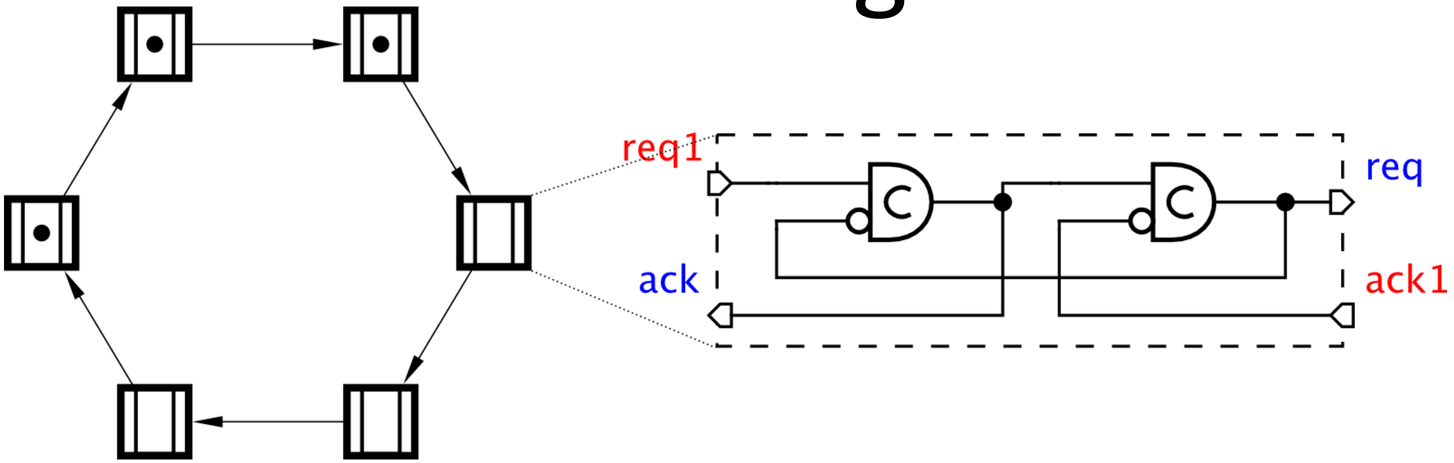
- Truly energy-modulated design must be power-adaptive
- Systems that are power adaptive are more resourceful and more resilient
- Power-adaptive systems can work in a broad range of power levels
- How to design such systems?

Grand-prix race with a fuel limit

© The Cahier Archive

The goal: Given a finite amount of fuel, maximize the total number of laps made all the cars on the circuit.
Unknown parameters: What is the optimum engine power?
What is the optimum number of cars on the circuit?

Ring-pipeline with a finite energy budget



Experiment:

a. A ring micropipeline with 5 stages is used in the experiment.

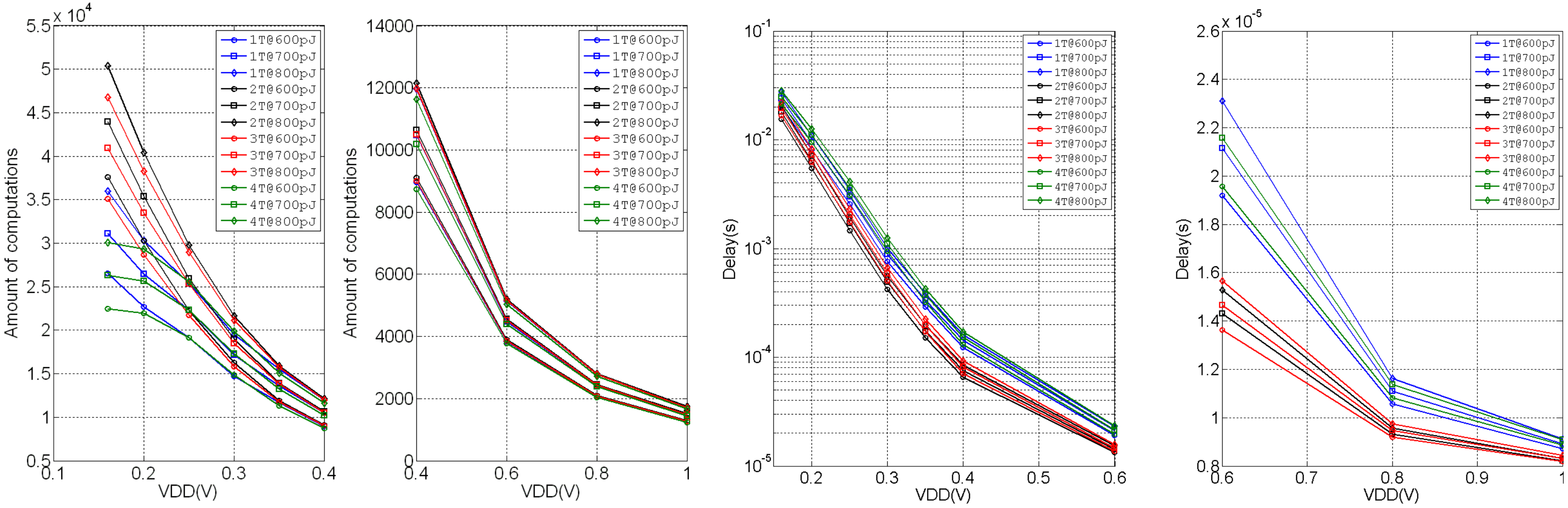
b. Simulation Results are obtained with different parallelism (1, 2, 3, 4 tokens), in different working voltages (1.0V, 0.8V, 0.6V, 0.4V, 0.35V, 0.25V, 0.2V, 0.16V), and under different amount of energy (600pJ, 700pJ, 800pJ).

c. A run stops when the energy is fully consumed.

d. The amount of computation is counted for each run.

e. A unit of computation is defined as one pulse generated in the pipeline.

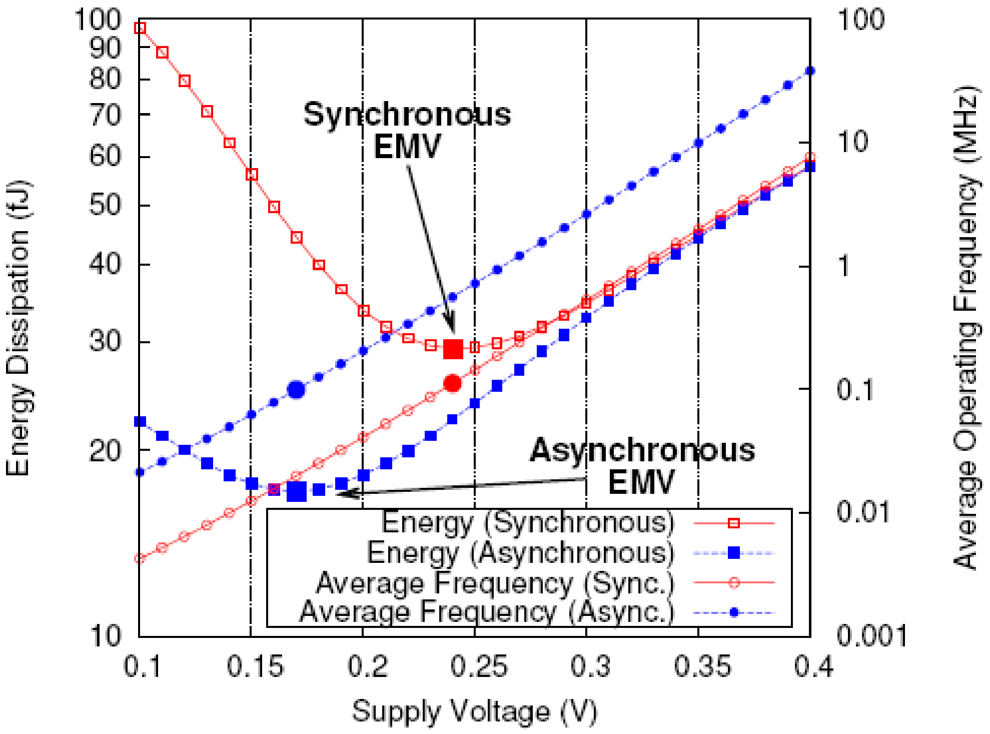
Ring pipeline with a given energy budget



Conclusions:

- The higher the concurrency the greater the amount of computation and the smaller the amount of leakage.
- At sub-threshold voltages, the amount of computation is STRONGLY affected by degree of concurrency, due to the effect of leakage.
- Above threshold, the amount of computation that is practically insensitive to the degree of concurrency.

Synchronous vs Self-Timed Design (in terms of energy efficiency)

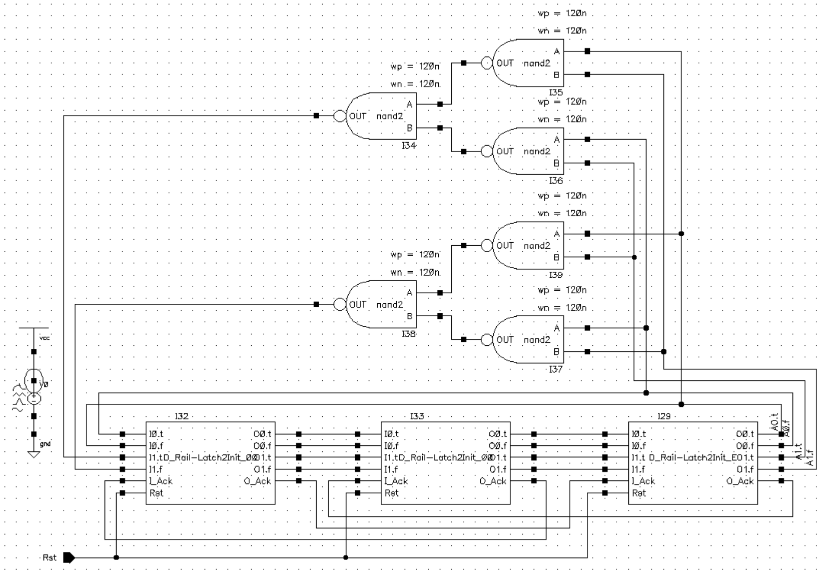


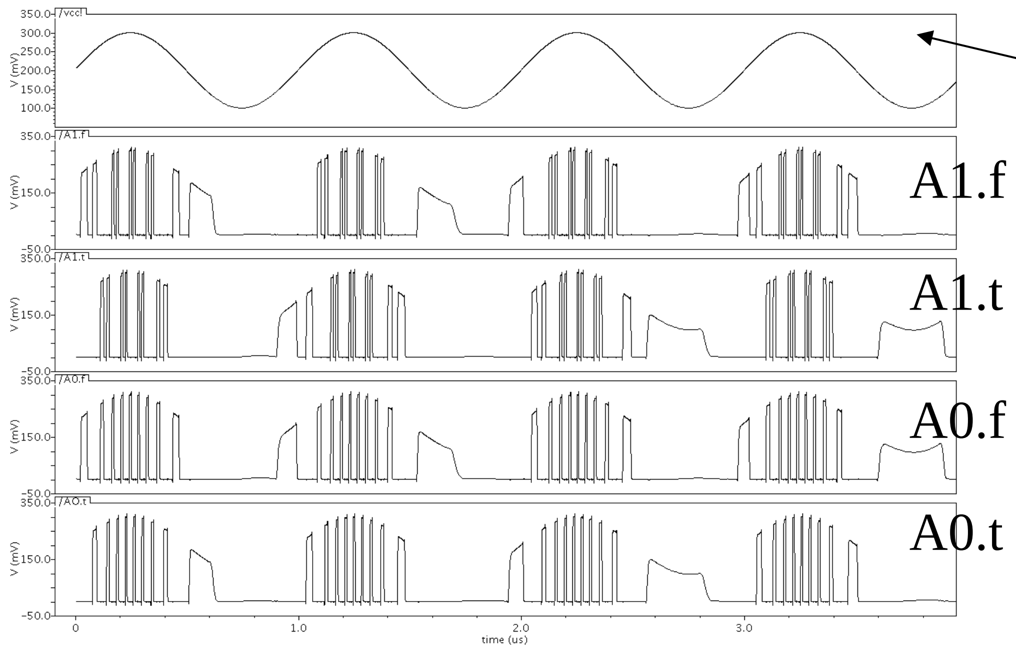
Asynchronous (self-timed) logic can provide completion detection and thus reduce the interval of leakage to minimum, thereby doing nothing well!

Source: Akgun et al, ASYNC’10

Closer look at AC-powered self-timed logic

2-bit Sequential Dual-rail
Asynchronous Counter



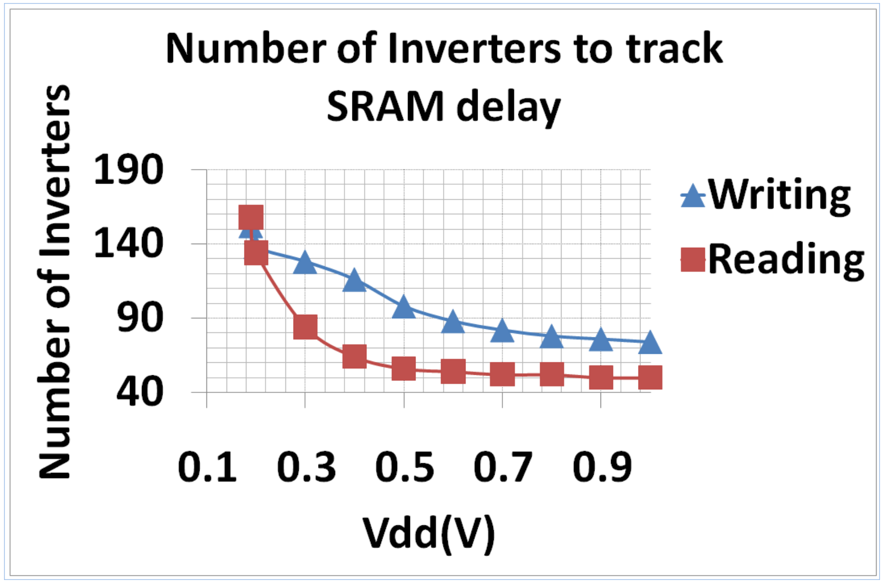


Supply: AC
200mV±100mV
Frequency: 1Mhz

Self-timed logic with completion detection is robust to power supply variations

Circuit-level: speed-independent SRAM

- Mismatch between delay lines and SRAM memories when reducing Vdd

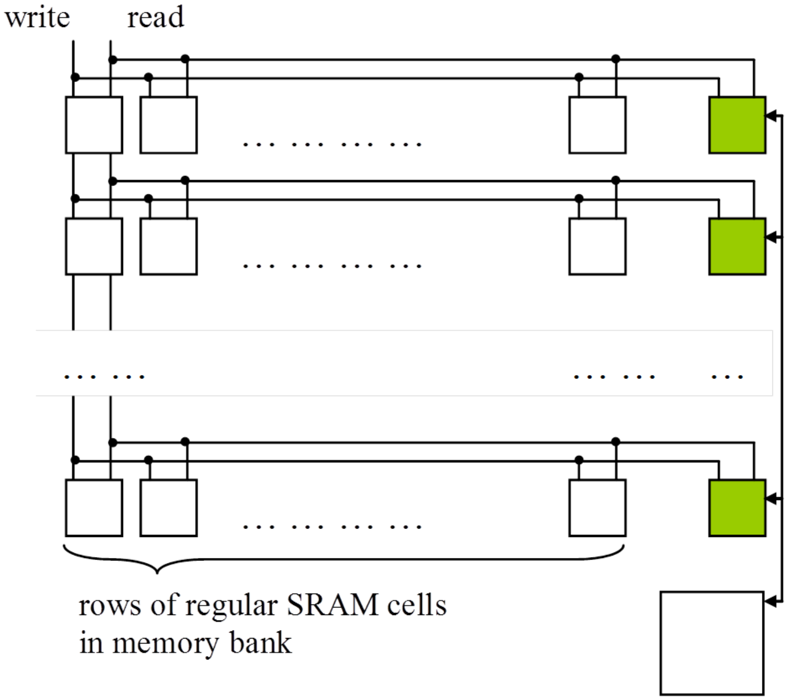


For example, under 1V Vdd, the delay of SRAM reading is equal to 50 inverters and under 190mV, the delay is equal to 158 inverters

- The problem has been well known so far
- Existing solutions:
  - Different delay lines in different range of Vdd
  - Duplicating a column of SRAM to be a delay line to bundle the whole SRAM
- The solutions require:
  - voltage references
  - DC-DC adaptor
- Completion detection needed?!

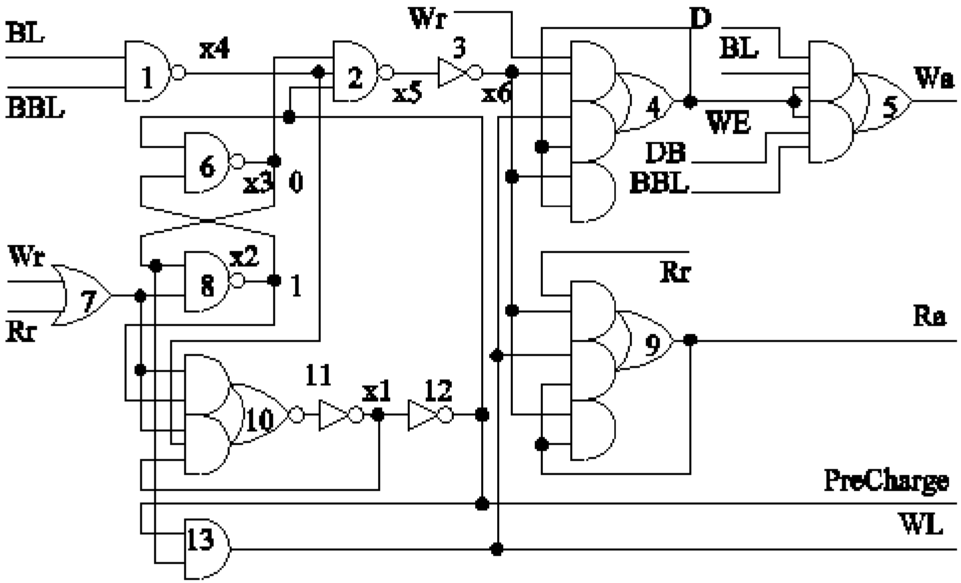
A. Baz et.al. PATMOS 2010, JOLPE 2011

Circuit-level: speed-independent SRAM



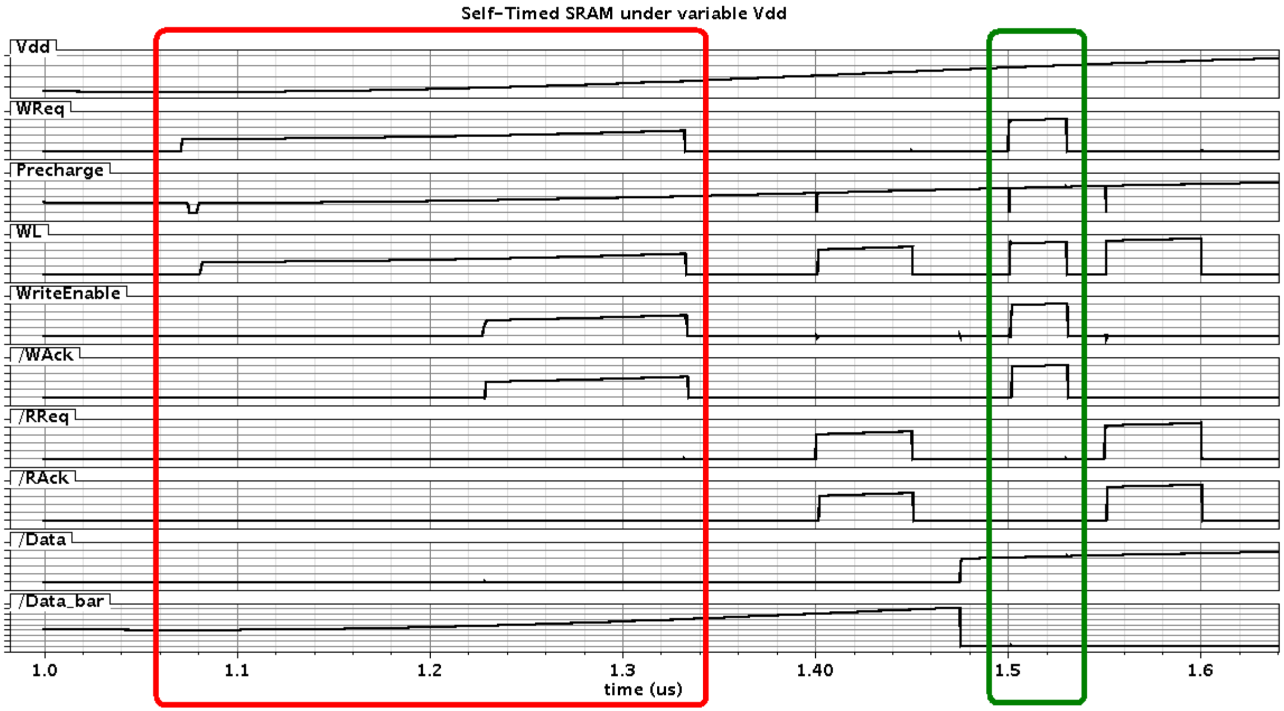
rows of regular SRAM cells in memory bank

timing controller for a column of latency bundling cells



Speed-independent control circuit (synthesized from a Petri net specification)

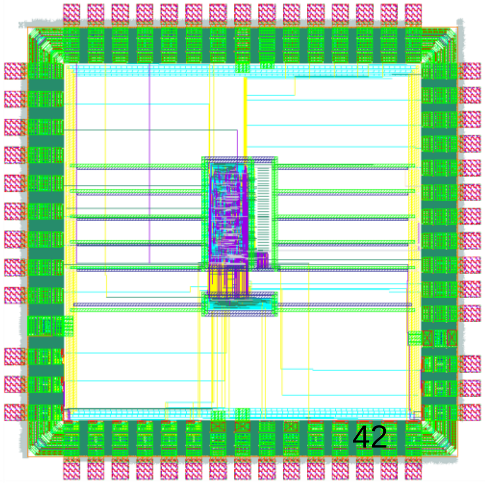
Circuit-level: speed-independent SRAM



Low Vdd – slow response

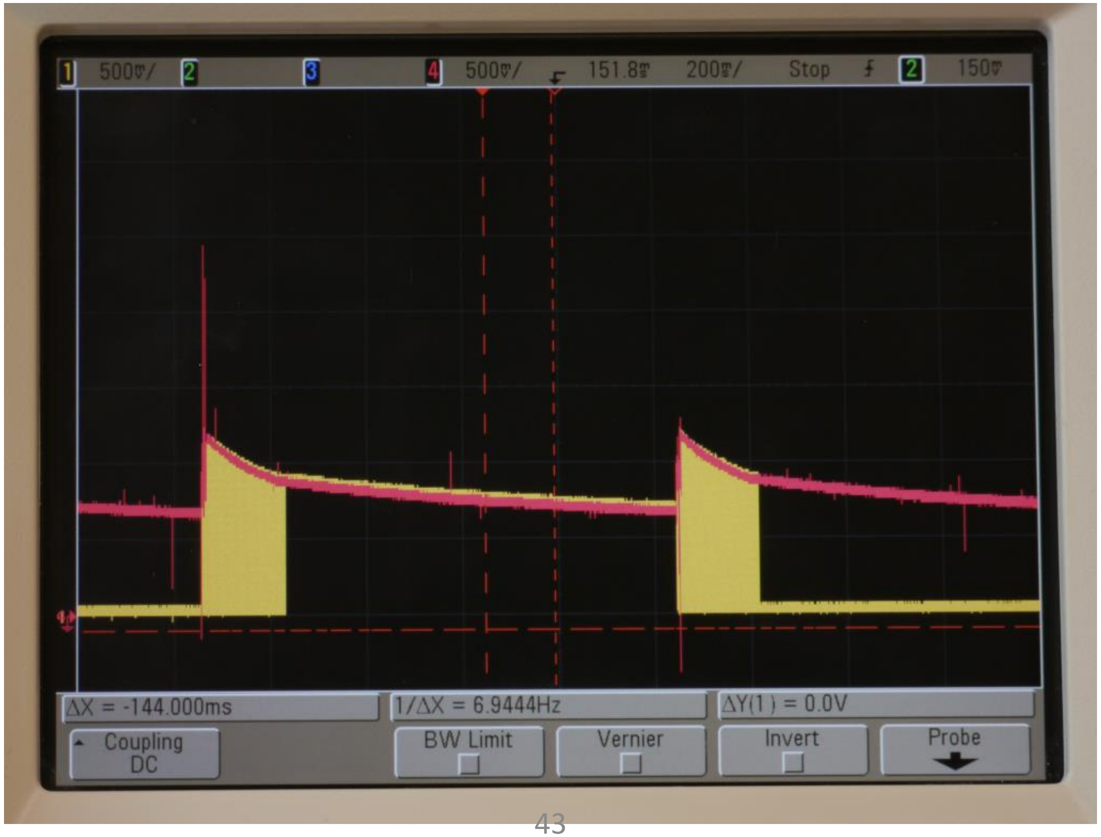
High Vdd – fast response

Self-timed SRAM chip: UMC CMOS 90nm

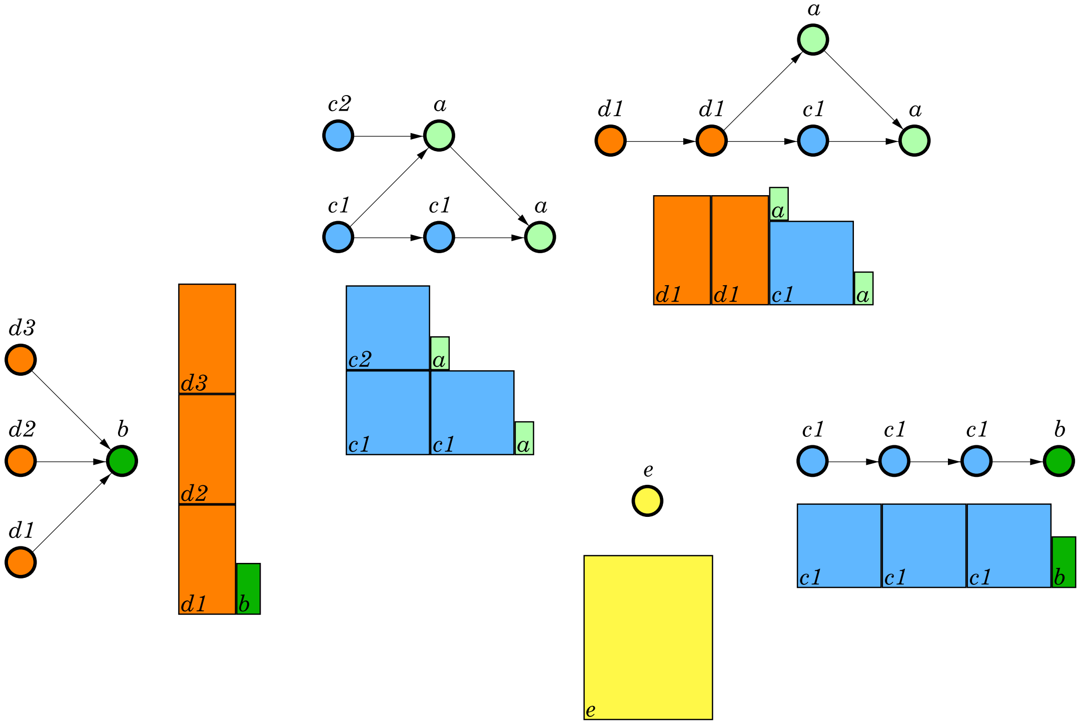


SRAM testing and results

- SRAM operations modulated by Vdd from a Capacitor Bank
- When Vdd goes below 0.75v, the ack signal is not generated by SRAM
- The circuit automatically wakes up when Vdd goes up



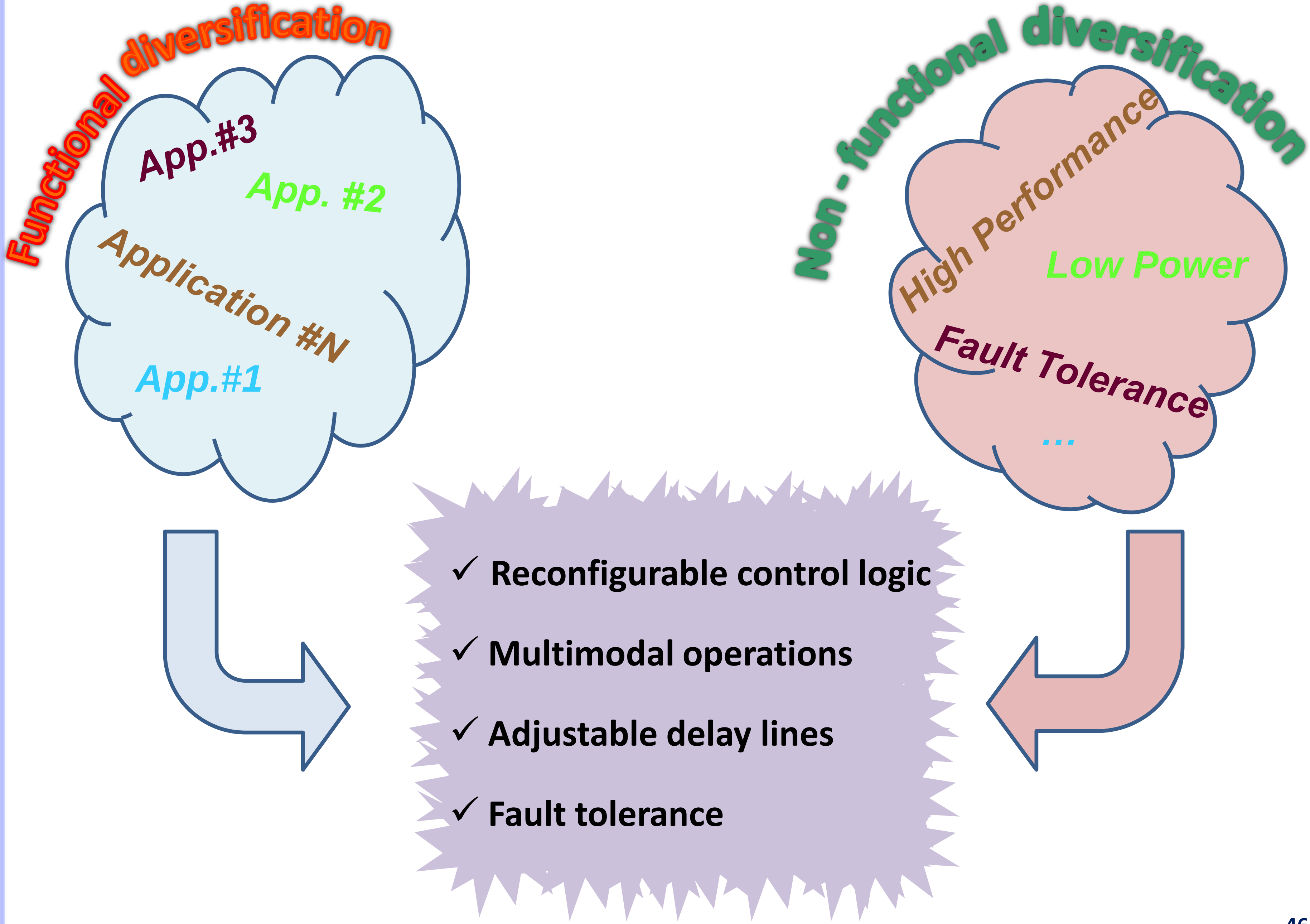
Power Proportional design using Formal Models



Achieving Power Proportionality

- Support for wide range of voltages
  - Asynchronous design
  - Unstable voltage supply (energy harvesting)
- Components optimised for different modes
  - Survival mode (power)
  - Mission mode (energy efficiency)
  - Emergency mode (performance)
- Reconfigurable instructions
  - Altering instruction behaviour in runtime

Diversification …



Functional diversification

App.#3

App. #2

Application #N

App.#1

Non - functional diversification

High Performance

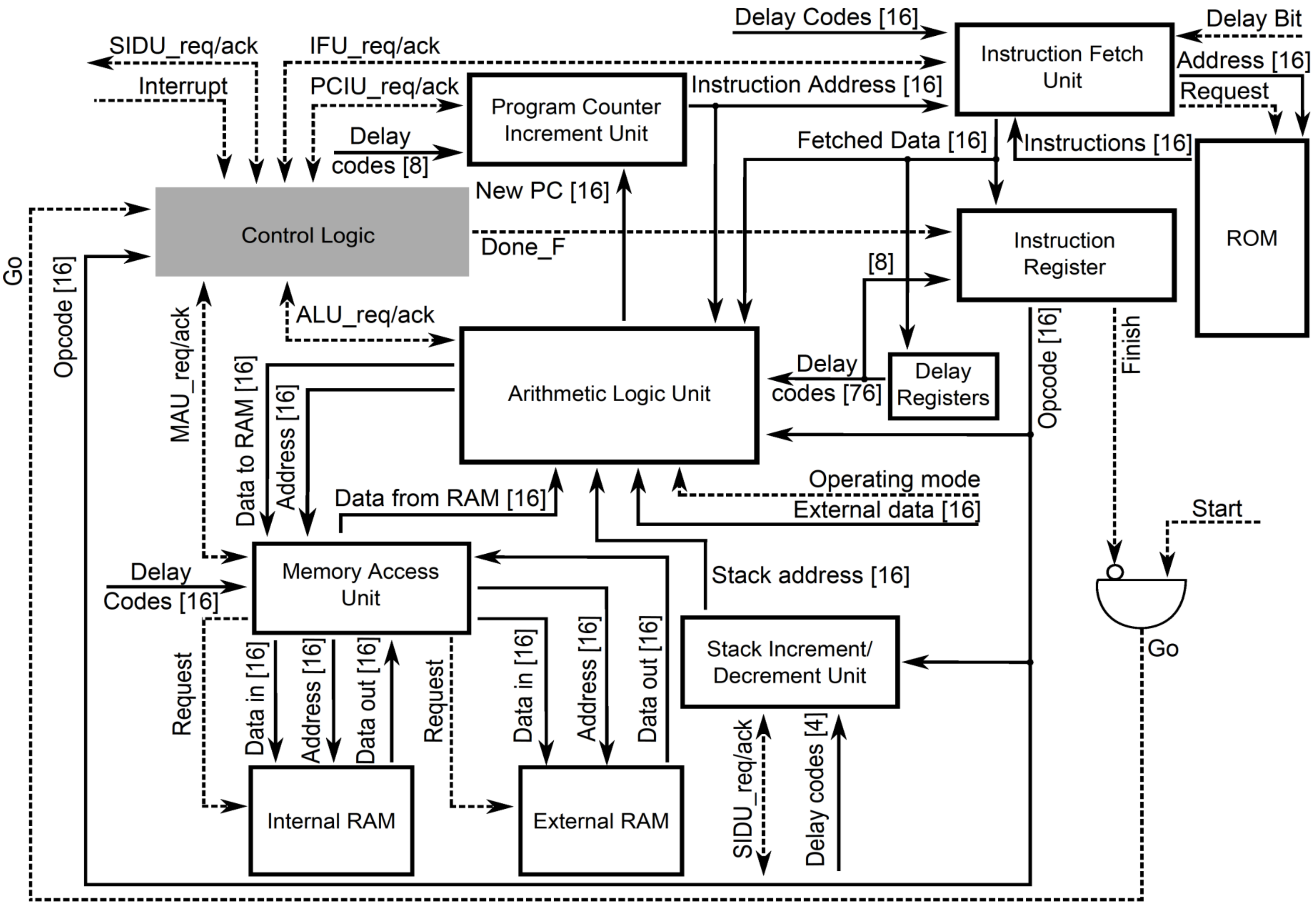
Low Power

Fault Tolerance

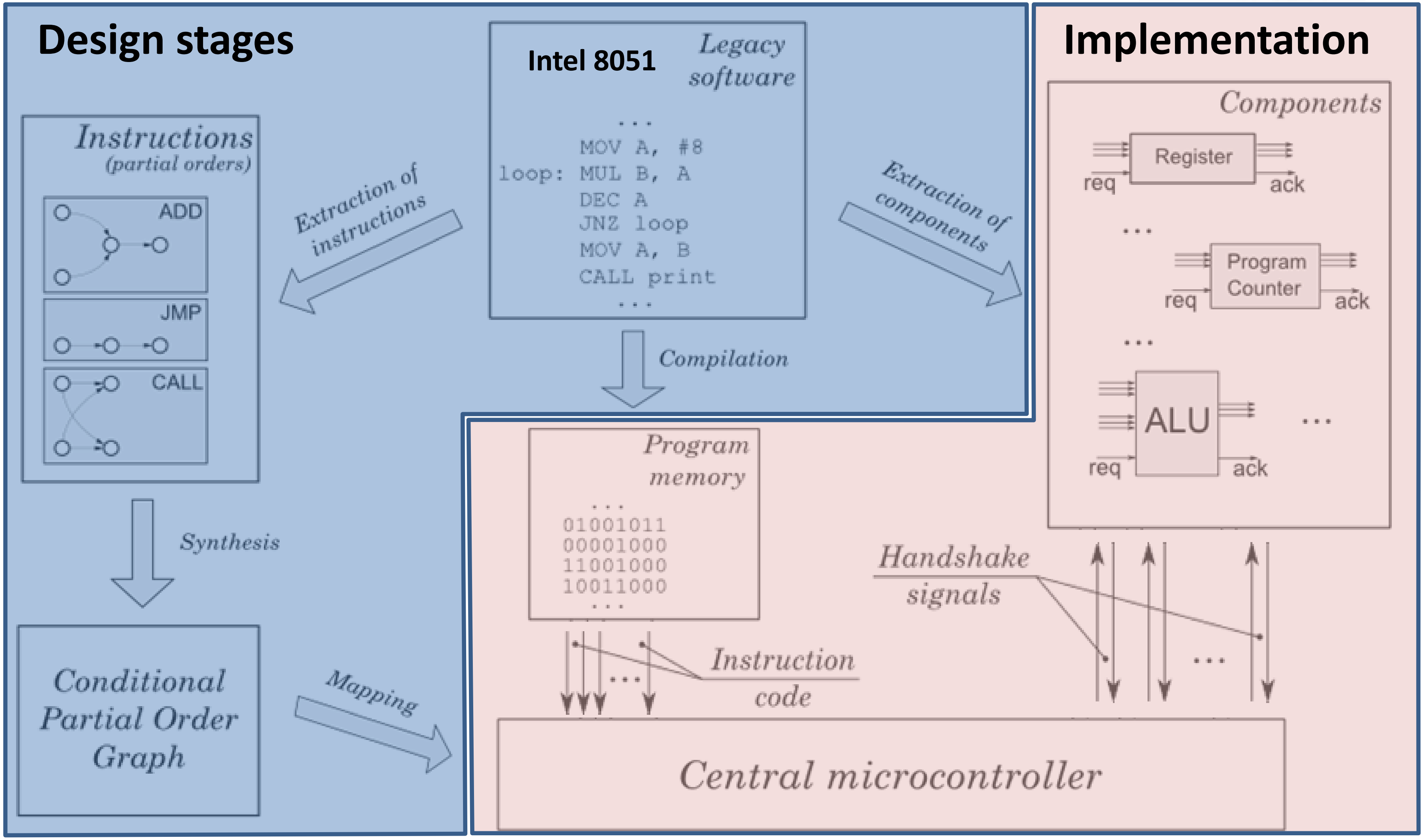
…

- ✓ Reconfigurable control logic
- ✓ Multimodal operations
- ✓ Adjustable delay lines
- ✓ Fault tolerance

Microprocessor architecture



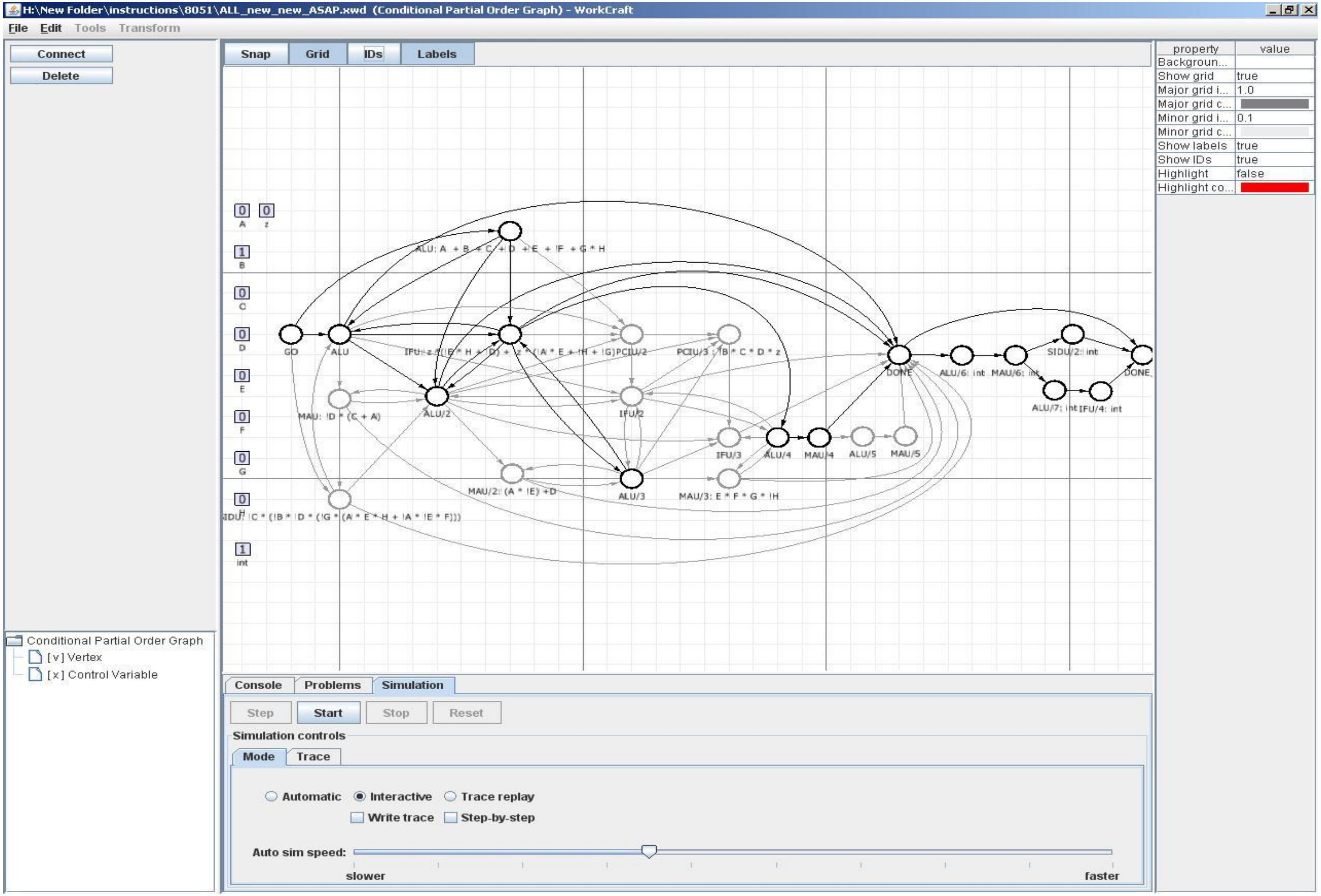
Conceptual view of the design process



```
...
      MOV A, #8
loop: MUL B, A
      DEC A
      JNZ loop
      MOV A, B
      CALL print
...
```

```
...
01001011
00001000
11001000
10011000
...
```

WORKCRAFT FRAMEWORK



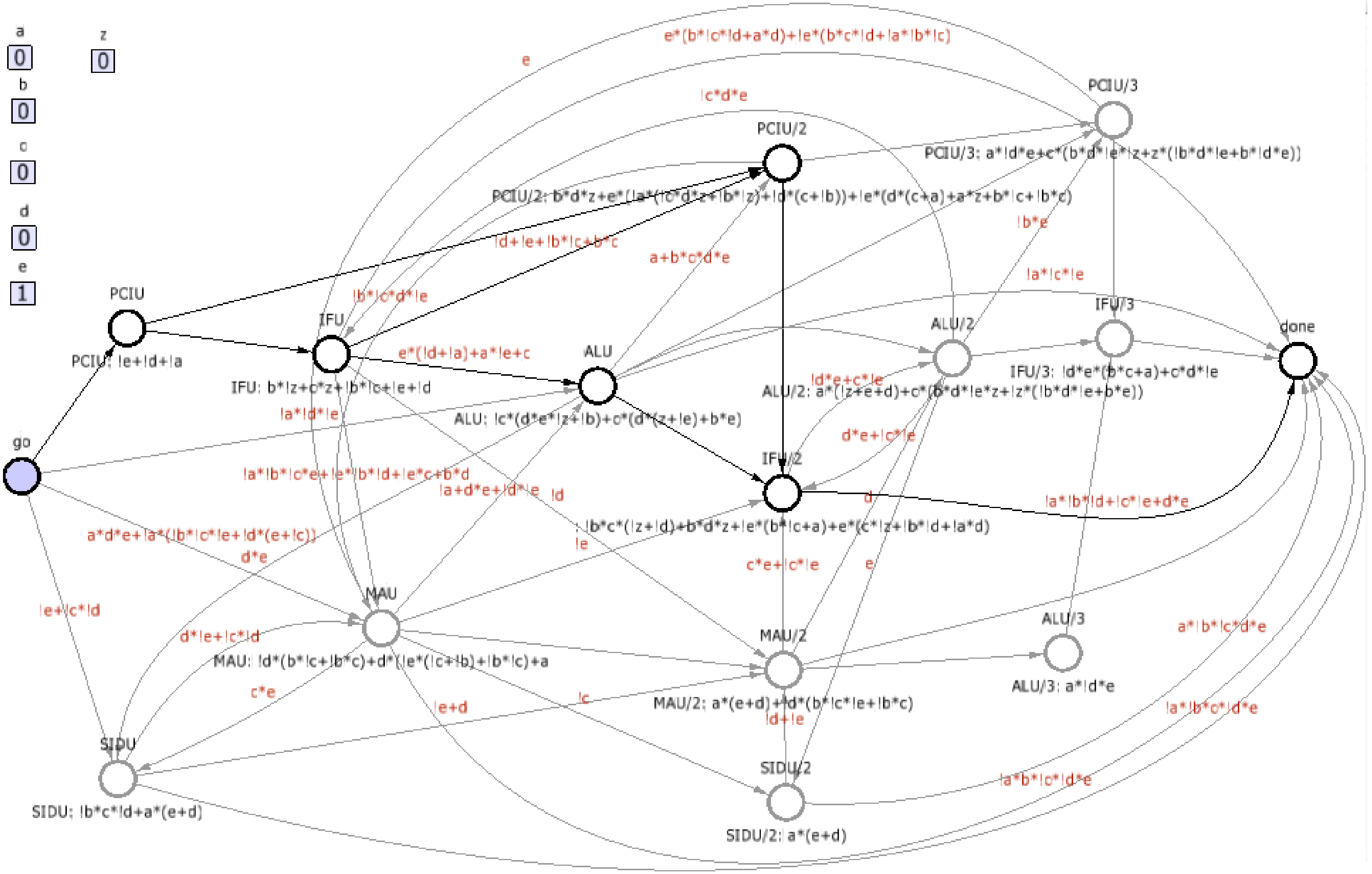
property	value
Backgroun...	
Show grid	true
Major grid i...	1.0
Major grid c...	▨
Minor grid i...	0.1
Minor grid c...	░
Show labels	true
Show IDs	true
Highlight	false
Highlight co...	🟥

0 A 0 z 1 B 0 C 0 D 0 E 0 F 0 G 0 H 1 int

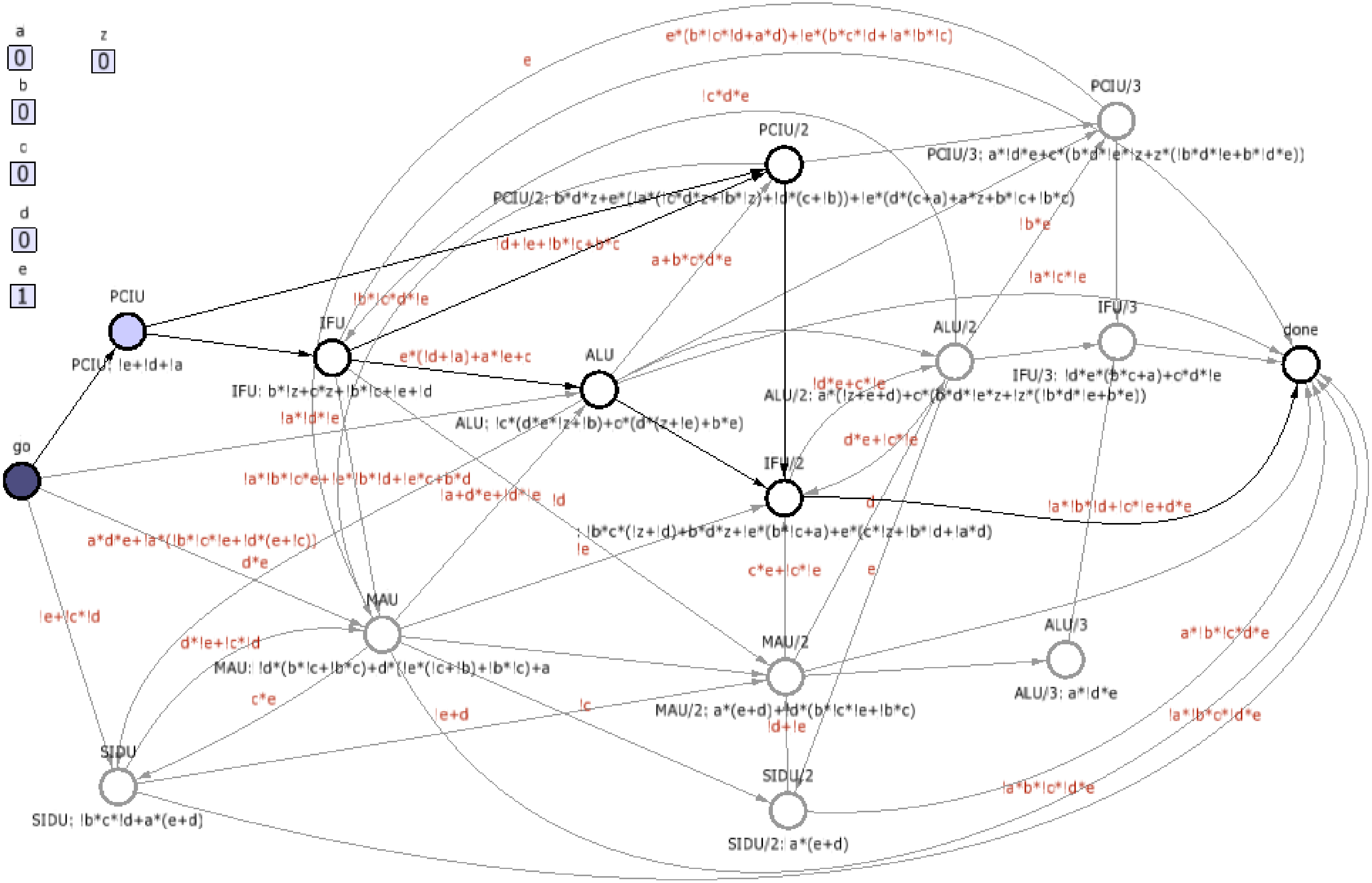
○ Automatic ● Interactive ○ Trace replay

□ Write trace □ Step-by-step

Intel 8051 ISA…

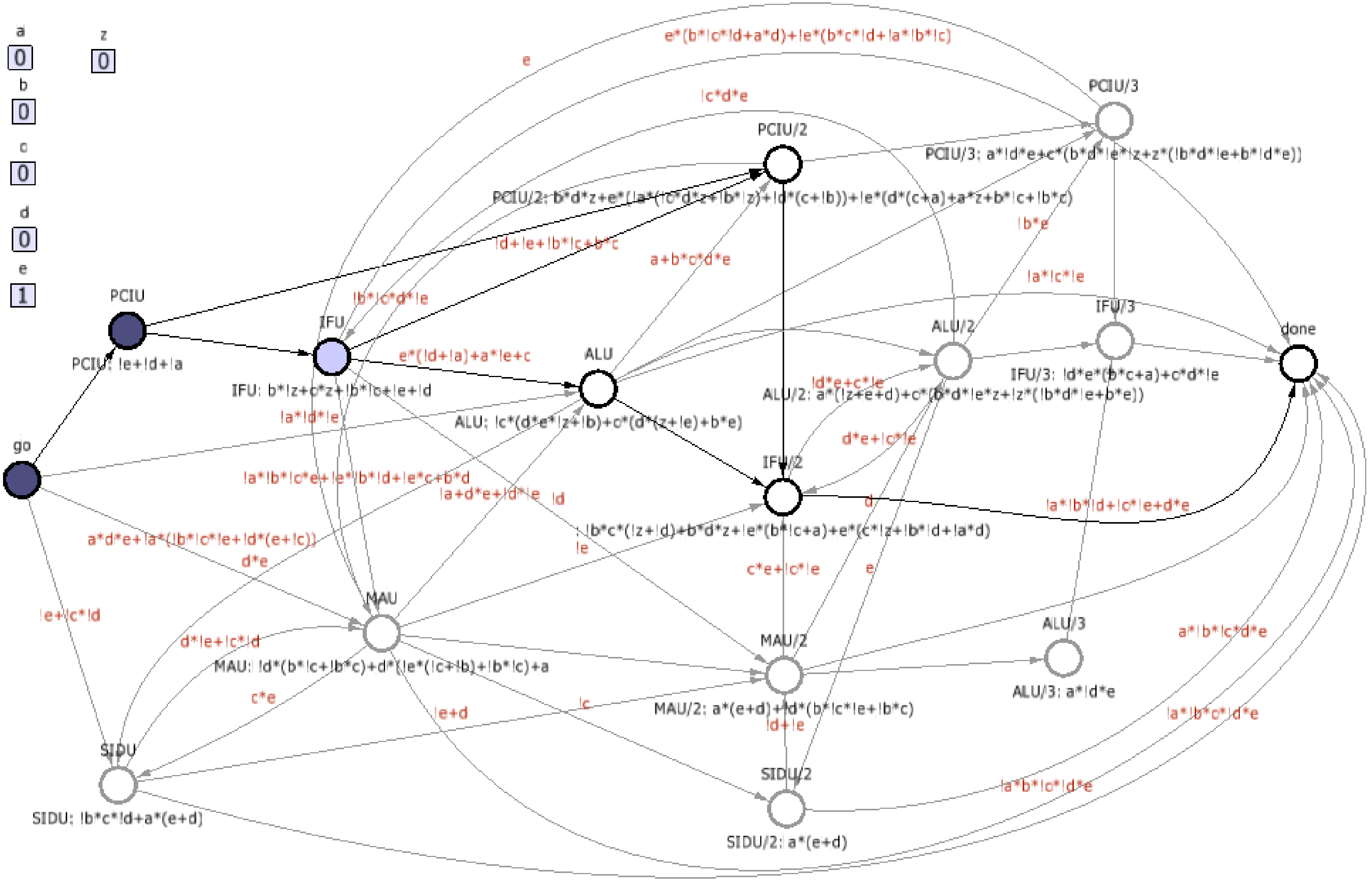

a
0
z
0
b
0
c
0
d
0
e
1
e*(b*!c*!d+a*d)+!e*(b*c*!d+!a*!b*!c)
e
!c*d*e
PCIU/3
PCIU/2
PCIU/3: a*!d*e+c*(b*d*!e*!z+z*(!b*d*!e+b*!d*e))
PCIU/2: b*d*z+e*(!a*(!c*d*z+!b*!z)+!d*(c+!b))+!e*(d*(c+a)+a*z+b*!c+!b*c)
!b*e
!d+!e+!b*!c+b*c
a+b*c*d*e
!a*!c*!e
PCIU
!b*!c*d*!e
IFU
IFU/3
ALU/2
done
e*(!d+!a)+a*!e+c
ALU
PCIU: !e+!d+!a
!d*e+c*!e
IFU/3: !d*e*(b*c+a)+c*d*!e
IFU: b*!z+c*z+!b*!c+!e+!d
ALU/2: a*(!z+e+d)+c*(b*d*!e*z+!z*(!b*d*!e+b*e))
!a*!d*!e
ALU: !c*(d*e*!z+!b)+c*(d*(z+!e)+b*e)
d*e+!c*!e
go
IFU/2
!a*!b*!c*e+!e*!b*!d+!e*c+b*d
!a+d*e+!d*!e
!d
d
!a*!b*!d+!c*!e+d*e
: !b*c*(!z+!d)+b*d*z+!e*(b*!c+a)+e*(c*!z+!b*!d+!a*d)
a*d*e+!a*(!b*!c*!e+!d*(e+!c))
!e
d*e
c*e+!c*!e
e
MAU
!e+!c*!d
ALU/3
a*!b*!c*d*e
d*!e+!c*!d
MAU/2
MAU: !d*(b*!c+!b*c)+d*(!e*(!c+!b)+!b*!c)+a
ALU/3: a*!d*e
c*e
!c
!e+d
MAU/2: a*(e+d)+!d*(b*!c*!e+!b*c)
!d+!e
!a*!b*c*!d*e
SIDU
SIDU/2
!a*b*!c*!d*e
SIDU: !b*c*!d+a*(e+d)
SIDU/2: a*(e+d)

Intel 8051 ISA…

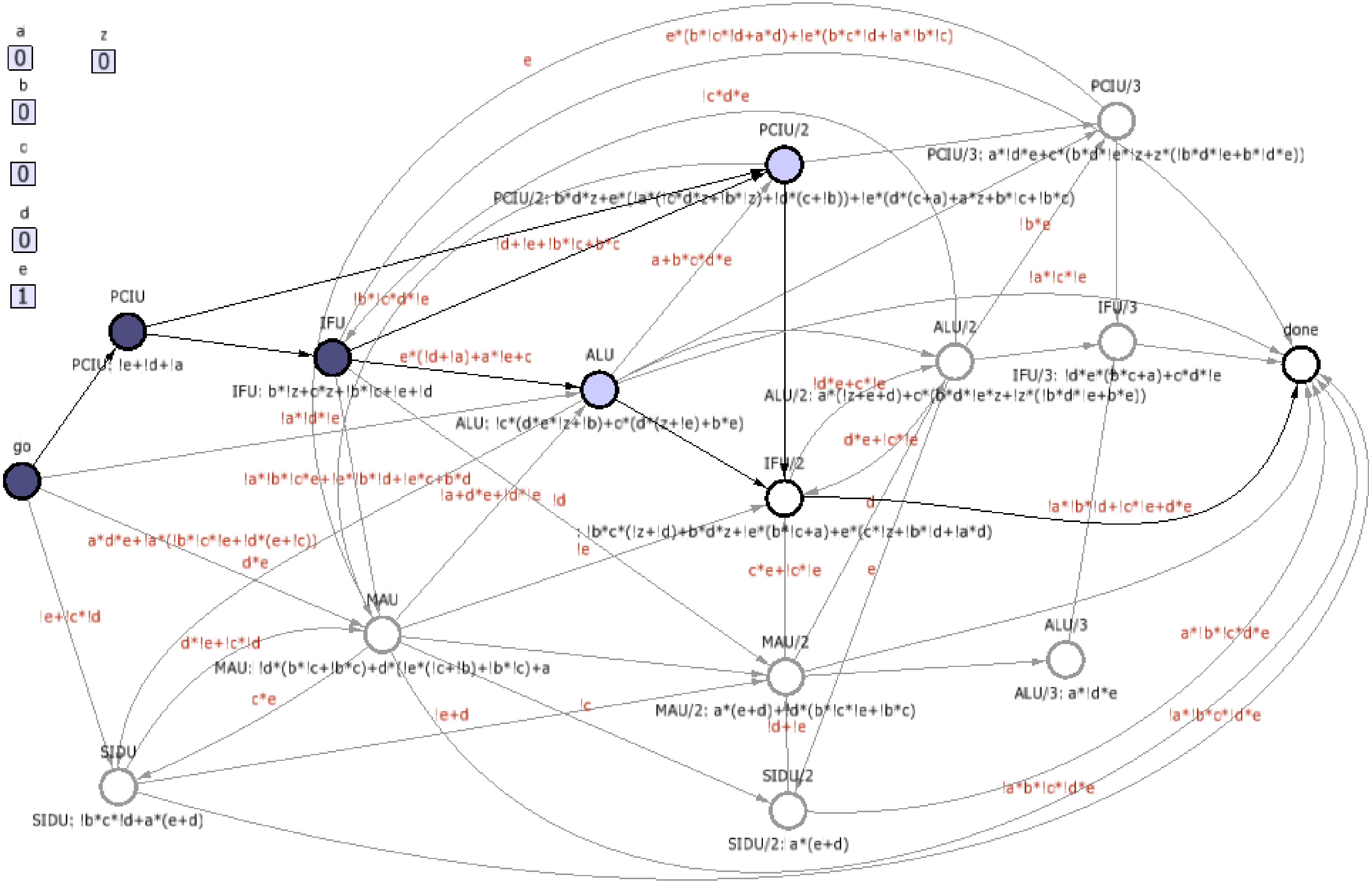


a
0
z
0
b
0
c
0
d
0
e
1
e
e*(b*!c*!d+a*d)+!e*(b*c*!d+!a*!b*!c)
!c*d*e
PCIU/3
PCIU/2
PCIU/3: a*!d*e+c*(b*d*!e*!z+z*(!b*d*!e+b*!d*e))
PCIU/2: b*d*z+e*(!a*(!c*d*z+!b*!z)+!d*(c+!b))+!e*(d*(c+a)+a*z+b*!c+!b*c)
!b*e
!d+!e+!b*!c+b*c
a+b*c*d*e
!a*!c*!e
!b*!c*d*!e
PCIU
IFU
IFU/3
ALU/2
done
e*(!d+!a)+a*!e+c
ALU
PCIU: !e+!d+!a
IFU: b*!z+c*z+!b*!c+!e+!d
IFU/3: !d*e*(b*c+a)+c*d*!e
!d*e+c*!e
ALU/2: a*(!z+e+d)+c*(b*d*!e*z+!z*(!b*d*!e+b*e))
!a*!d*!e
ALU: !c*(d*e*!z+!b)+c*(d*(z+!e)+b*e)
d*e+!c*!e
go
IFU/2
!a*!b*!c*e+!e*!b*!d+!e*c+b*d
!a+d*e+!d*!e
!d
d
!a*!b*!d+!c*!e+d*e
: !b*c*(!z+!d)+b*d*z+!e*(b*!c+a)+e*(c*!z+!b*!d+!a*d)
a*d*e+!a*(!b*!c*!e+!d*(e+!c))
!e
d*e
c*e+!c*!e
e
!e+!c*!d
MAU
ALU/3
a*!b*!c*d*e
MAU/2
d*!e+!c*!d
MAU: !d*(b*!c+!b*c)+d*(!e*(!c+!b)+!b*!c)+a
c*e
ALU/3: a*!d*e
!e+d
!c
MAU/2: a*(e+d)+!d*(b*!c*!e+!b*c)
!a*!b*c*!d*e
!d+!e
SIDU
SIDU/2
!a*b*!c*!d*e
SIDU: !b*c*!d+a*(e+d)
SIDU/2: a*(e+d)

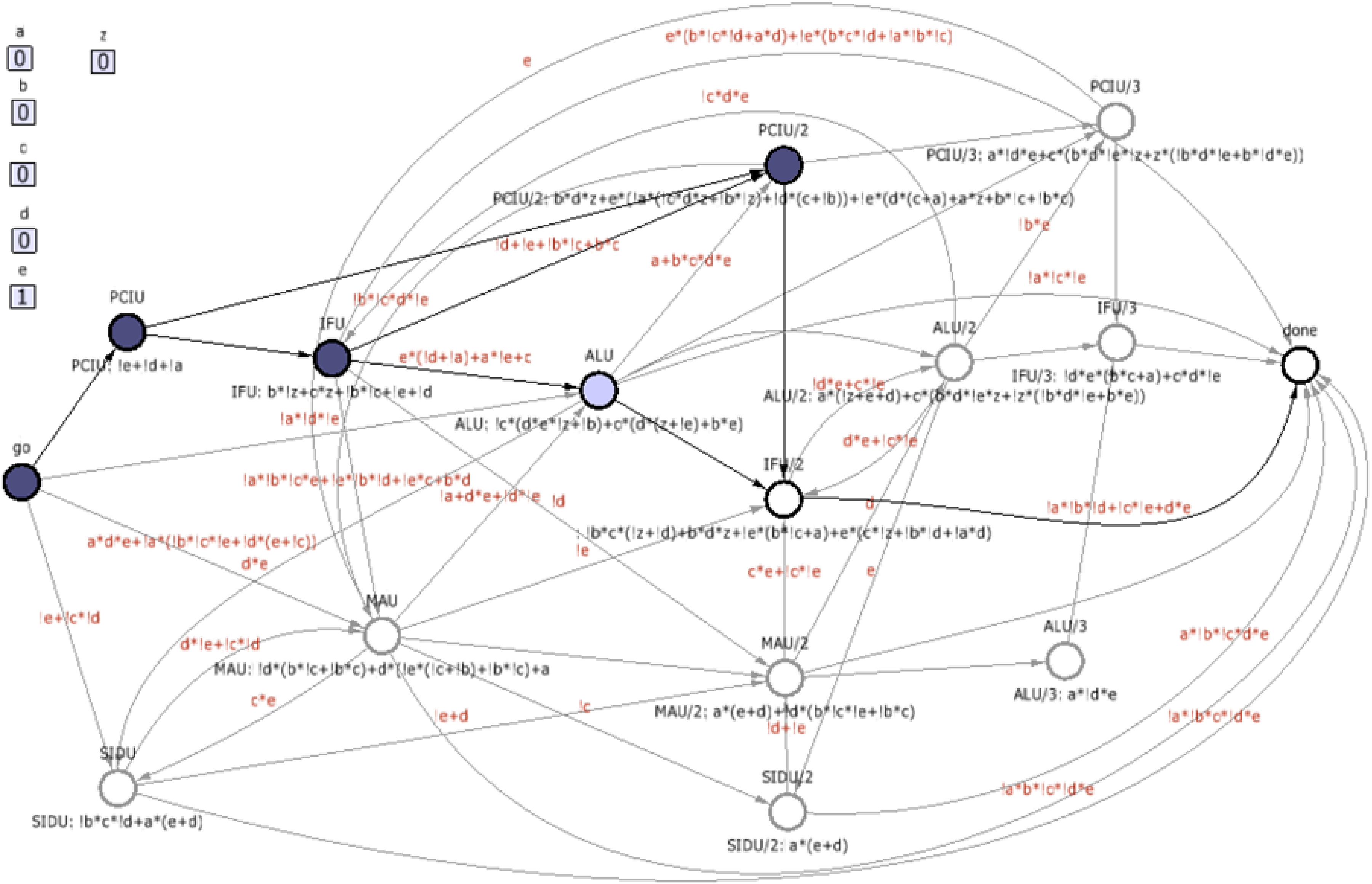
Intel 8051 ISA…


a
0
z
0
b
0
c
0
d
0
e
1
e
e*(b*!c*!d+a*d)+!e*(b*c*!d+!a*!b*!c)
PCIU/3
!c*d*e
PCIU/2
PCIU/3: a*!d*e+c*(b*d*!e*!z+z*(!b*d*!e+b*!d*e))
PCIU/2: b*d*z+e*(!a*(!c*d*z+!b*!z)+!d*(c+!b))+!e*(d*(c+a)+a*z+b*!c+!b*c)
!b*e
!d+!e+!b*!c+b*c
a+b*c*d*e
!a*!c*!e
!b*!c*d*!e
IFU/3
PCIU
ALU/2
done
IFU
e*(!d+!a)+a*!e+c
ALU
PCIU: !e+!d+!a
IFU/3: !d*e*(b*c+a)+c*d*!e
IFU: b*!z+c*z+!b*!c+!e+!d
!d*e+c*!e
ALU/2: a*(!z+e+d)+c*(b*d*!e*z+!z*(!b*d*!e+b*e))
!a*!d*!e
ALU: !c*(d*e*!z+!b)+c*(d*(z+!e)+b*e)
d*e+!c*!e
go
IFU/2
!a*!b*!c*e+!e*!b*!d+!e*c+b*d
!a+d*e+!d*!e
!d
d
!a*!b*!d+!c*!e+d*e
: !b*c*(!z+!d)+b*d*z+!e*(b*!c+a)+e*(c*!z+!b*!d+!a*d)
a*d*e+!a*(!b*!c*!e+!d*(e+!c))
!e
d*e
c*e+!c*!e
e
MAU
!e+!c*!d
ALU/3
a*!b*!c*d*e
MAU/2
d*!e+!c*!d
MAU: !d*(b*!c+!b*c)+d*(!e*(!c+!b)+!b*!c)+a
ALU/3: a*!d*e
c*e
!c
!e+d
MAU/2: a*(e+d)+!d*(b*!c*!e+!b*c)
!d+!e
!a*!b*c*!d*e
SIDU
SIDU/2
!a*b*!c*!d*e
SIDU: !b*c*!d+a*(e+d)
SIDU/2: a*(e+d)

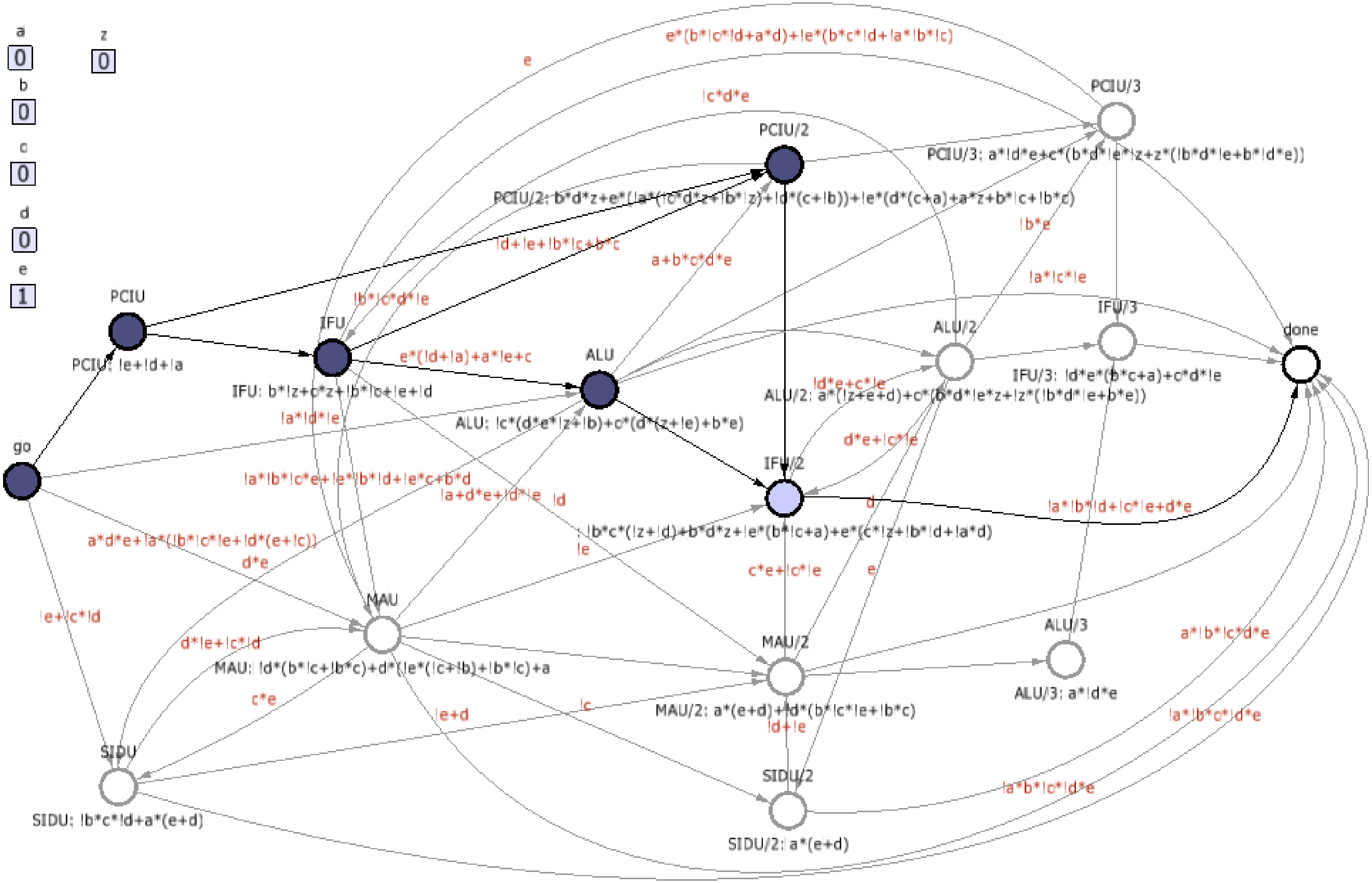
Intel 8051 ISA…


a
0
z
0
b
0
c
0
d
0
e
1
PCIU
PCIU: !e+!d+!a
go
IFU
IFU: b*!z+c*z+!b*!c+!e+!d
ALU
ALU: !c*(d*e*!z+!b)+c*(d*(z+!e)+b*e)
PCIU/2
PCIU/2: b*d*z+e*(!a*(!c*d*z+!b*!z)+!d*(c+!b))+!e*(d*(c+a)+a*z+b*!c+!b*c)
PCIU/3
PCIU/3: a*!d*e+c*(b*d*!e*!z+z*(!b*d*!e+b*!d*e))
ALU/2
ALU/2: a*(!z+e+d)+c*(b*d*!e*z+!z*(!b*d*!e+b*e))
IFU/3
IFU/3: !d*e*(b*c+a)+c*d*!e
done
IFU/2
: !b*c*(!z+!d)+b*d*z+!e*(b*!c+a)+e*(c*!z+!b*!d+!a*d)
MAU
MAU: !d*(b*!c+!b*c)+d*(!e*(!c+!b)+!b*!c)+a
MAU/2
MAU/2: a*(e+d)+!d*(b*!c*!e+!b*c)
ALU/3
ALU/3: a*!d*e
SIDU
SIDU: !b*c*!d+a*(e+d)
SIDU/2
SIDU/2: a*(e+d)
e*(b*!c*!d+a*d)+!e*(b*c*!d+!a*!b*!c)
e
!c*d*e
!d+!e+!b*!c+b*c
a+b*c*d*e
!b*e
!a*!c*!e
!b*!c*d*!e
e*(!d+!a)+a*!e+c
!d*e+c*!e
!a*!d*!e
d*e+!c*!e
!a*!b*!c*e+!e*!b*!d+!e*c+b*d
!a+d*e+!d*!e
!d
d
!a*!b*!d+!c*!e+d*e
a*d*e+!a*(!b*!c*!e+!d*(e+!c))
d*e
!e
c*e+!c*!e
e
!e+!c*!d
d*!e+!c*!d
c*e
!e+d
!c
!d+!e
a*!b*!c*d*e
!a*!b*c*!d*e
!a*b*!c*!d*e

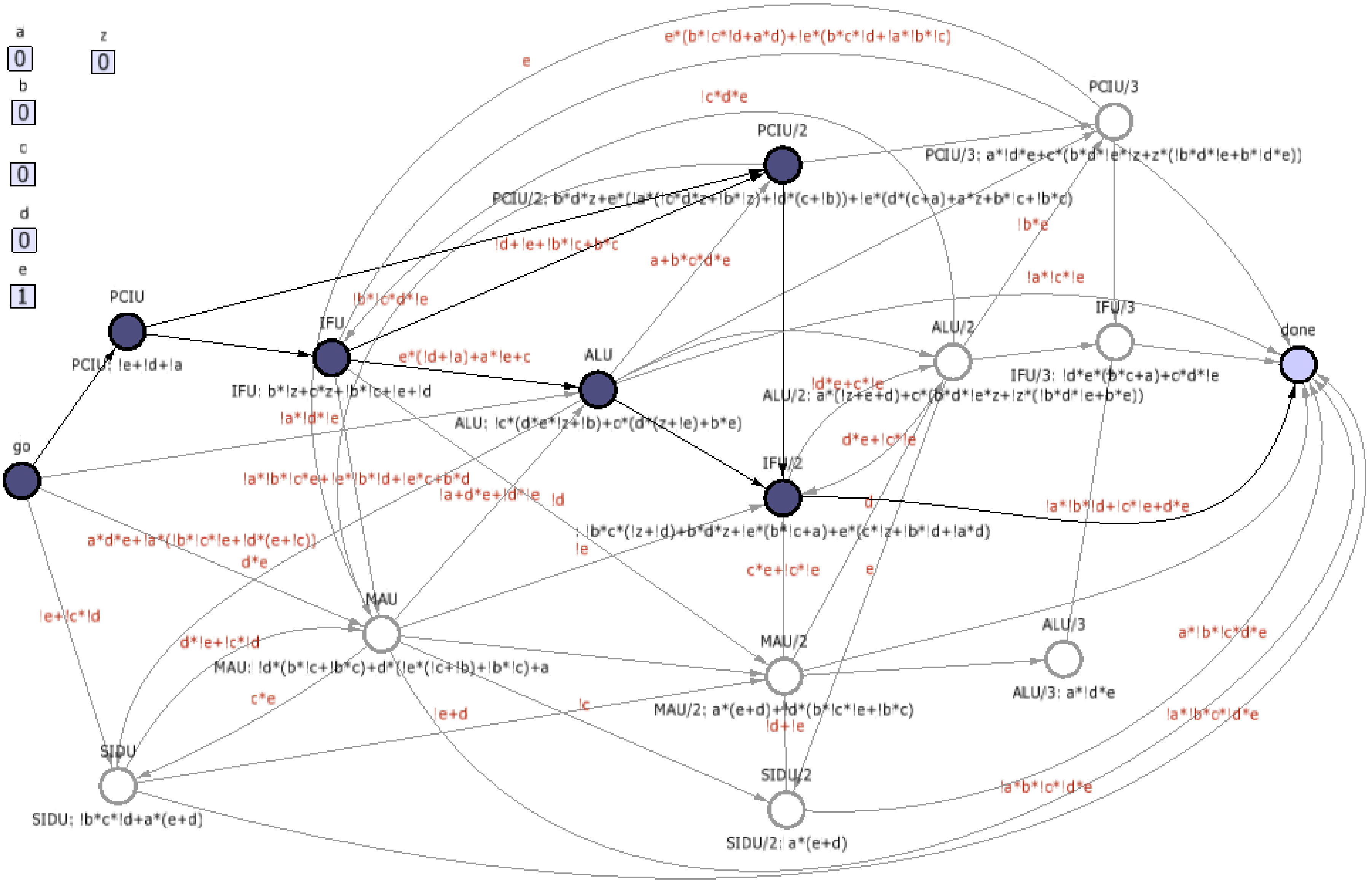
Intel 8051 ISA…


a
0
z
0
b
0
c
0
d
0
e
1
e
e*(b*!c*!d+a*d)+!e*(b*c*!d+!a*!b*!c)
!c*d*e
PCIU/3
PCIU/2
PCIU/3: a*!d*e+c*(b*d*!e*!z+z*(!b*d*!e+b*!d*e))
PCIU/2: b*d*z+e*(!a*(!c*d*z+!b*!z)+!d*(c+!b))+!e*(d*(c+a)+a*z+b*!c+!b*c)
!b*e
!d+!e+!b*!c+b*c
a+b*c*d*e
!a*!c*!e
!b*!c*d*!e
PCIU
IFU
IFU/3
ALU/2
done
ALU
e*(!d+!a)+a*!e+c
PCIU: !e+!d+!a
IFU: b*!z+c*z+!b*!c+!e+!d
!d*e+c*!e
IFU/3: !d*e*(b*c+a)+c*d*!e
ALU/2: a*(!z+e+d)+c*(b*d*!e*z+!z*(!b*d*!e+b*e))
!a*!d*!e
ALU: !c*(d*e*!z+!b)+c*(d*(z+!e)+b*e)
d*e+!c*!e
go
IFU/2
!a*!b*!c*e+!e*!b*!d+!e*c+b*d
!a+d*e+!d*!e
!d
d
!a*!b*!d+!c*!e+d*e
: !b*c*(!z+!d)+b*d*z+!e*(b*!c+a)+e*(c*!z+!b*!d+!a*d)
!e
a*d*e+!a*(!b*!c*!e+!d*(e+!c))
d*e
c*e+!c*!e
e
MAU
!e+!c*!d
ALU/3
a*!b*!c*d*e
MAU/2
d*!e+!c*!d
MAU: !d*(b*!c+!b*c)+d*(!e*(!c+!b)+!b*!c)+a
ALU/3: a*!d*e
c*e
!e+d
!c
MAU/2: a*(e+d)+!d*(b*!c*!e+!b*c)
!d+!e
!a*!b*c*!d*e
SIDU
SIDU/2
!a*b*!c*!d*e
SIDU: !b*c*!d+a*(e+d)
SIDU/2: a*(e+d)

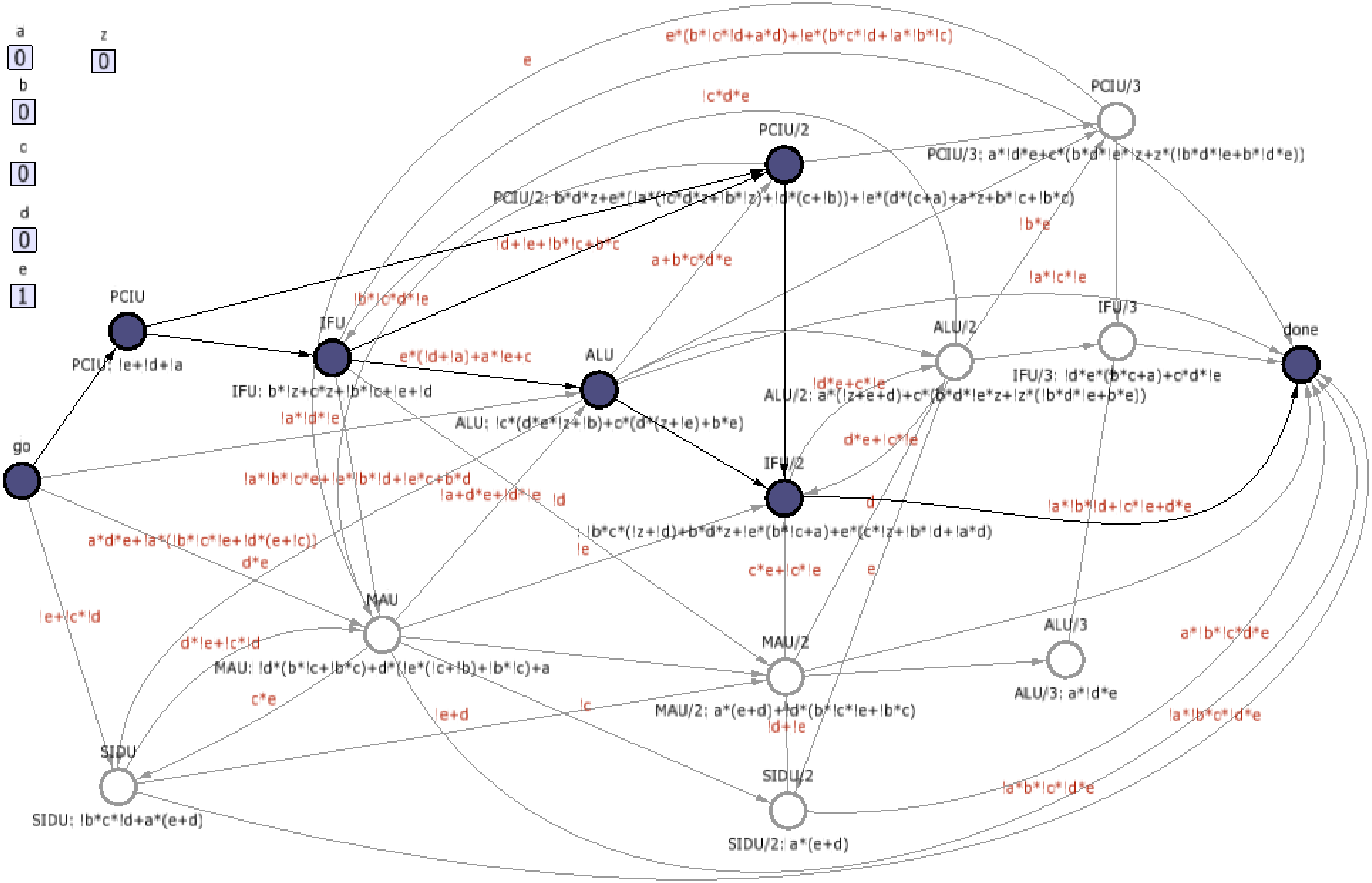
Intel 8051 ISA…


a
0
z
0
b
0
c
0
d
0
e
1
e
e*(b*!c*!d+a*d)+!e*(b*c*!d+!a*!b*!c)
!c*d*e
PCIU/3
PCIU/2
PCIU/3: a*!d*e+c*(b*d*!e*!z+z*(!b*d*!e+b*!d*e))
PCIU/2: b*d*z+e*(!a*(!c*d*z+!b*!z)+!d*(c+!b))+!e*(d*(c+a)+a*z+b*!c+!b*c)
!b*e
!d+!e+!b*!c+b*c
a+b*c*d*e
!a*!c*!e
PCIU
!b*!c*d*!e
IFU/3
IFU
ALU/2
done
e*(!d+!a)+a*!e+c
ALU
PCIU: !e+!d+!a
IFU: b*!z+c*z+!b*!c+!e+!d
!d*e+c*!e
IFU/3: !d*e*(b*c+a)+c*d*!e
ALU/2: a*(!z+e+d)+c*(b*d*!e*z+!z*(!b*d*!e+b*e))
!a*!d*!e
ALU: !c*(d*e*!z+!b)+c*(d*(z+!e)+b*e)
d*e+!c*!e
go
IFU/2
!a*!b*!c*e+!e*!b*!d+!e*c+b*d
!a+d*e+!d*!e
!d
d
!a*!b*!d+!c*!e+d*e
: !b*c*(!z+!d)+b*d*z+!e*(b*!c+a)+e*(c*!z+!b*!d+!a*d)
a*d*e+!a*(!b*!c*!e+!d*(e+!c))
!e
d*e
c*e+!c*!e
e
MAU
!e+!c*!d
ALU/3
a*!b*!c*d*e
MAU/2
d*!e+!c*!d
MAU: !d*(b*!c+!b*c)+d*(!e*(!c+!b)+!b*!c)+a
ALU/3: a*!d*e
c*e
!c
!e+d
MAU/2: a*(e+d)+!d*(b*!c*!e+!b*c)
!a*!b*c*!d*e
!d+!e
SIDU
SIDU/2
!a*b*!c*!d*e
SIDU: !b*c*!d+a*(e+d)
SIDU/2: a*(e+d)

Intel 8051 ISA…

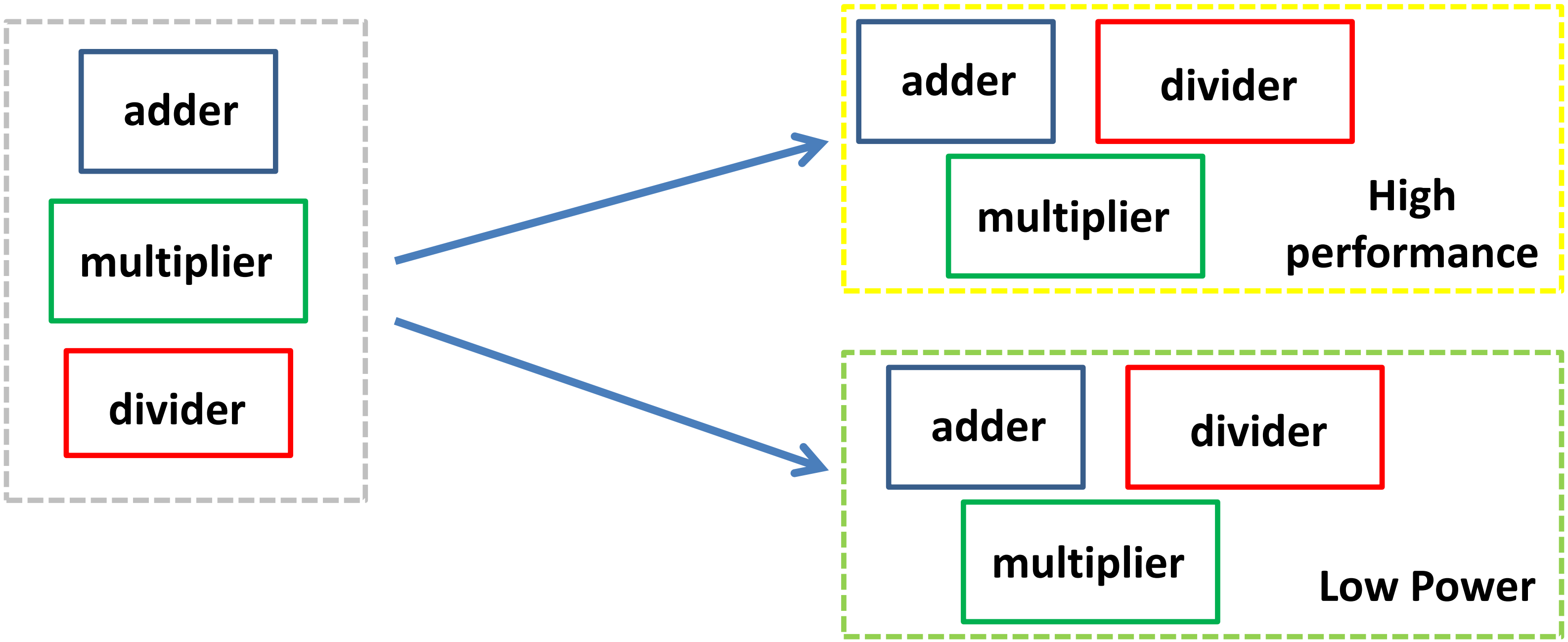

a
0
z
0
b
0
c
0
d
0
e
1
e
e*(b*!c*!d+a*d)+!e*(b*c*!d+!a*!b*!c)
!c*d*e
PCIU/3
PCIU/2
PCIU/3: a*!d*e+c*(b*d*!e*!z+z*(!b*d*!e+b*!d*e))
PCIU/2: b*d*z+e*(!a*(!c*d*z+!b*!z)+!d*(c+!b))+!e*(d*(c+a)+a*z+b*!c+!b*c)
!b*e
!d+!e+!b*!c+b*c
a+b*c*d*e
!a*!c*!e
PCIU
!b*!c*d*!e
IFU
IFU/3
ALU/2
done
e*(!d+!a)+a*!e+c
ALU
PCIU: !e+!d+!a
IFU: b*!z+c*z+!b*!c+!e+!d
!d*e+c*!e
IFU/3: !d*e*(b*c+a)+c*d*!e
ALU/2: a*(!z+e+d)+c*(b*d*!e*z+!z*(!b*d*!e+b*e))
!a*!d*!e
ALU: !c*(d*e*!z+!b)+c*(d*(z+!e)+b*e)
d*e+!c*!e
go
IFU/2
!a*!b*!c*e+!e*!b*!d+!e*c+b*d
!a+d*e+!d*!e
!d
d
!a*!b*!d+!c*!e+d*e
: !b*c*(!z+!d)+b*d*z+!e*(b*!c+a)+e*(c*!z+!b*!d+!a*d)
a*d*e+!a*(!b*!c*!e+!d*(e+!c))
!e
d*e
c*e+!c*!e
e
!e+!c*!d
MAU
ALU/3
a*!b*!c*d*e
MAU/2
d*!e+!c*!d
MAU: !d*(b*!c+!b*c)+d*(!e*(!c+!b)+!b*!c)+a
ALU/3: a*!d*e
c*e
!c
!e+d
MAU/2: a*(e+d)+!d*(b*!c*!e+!b*c)
!a*!b*c*!d*e
!d+!e
SIDU
SIDU/2
!a*b*!c*!d*e
SIDU: !b*c*!d+a*(e+d)
SIDU/2: a*(e+d)

Intel 8051 ISA…



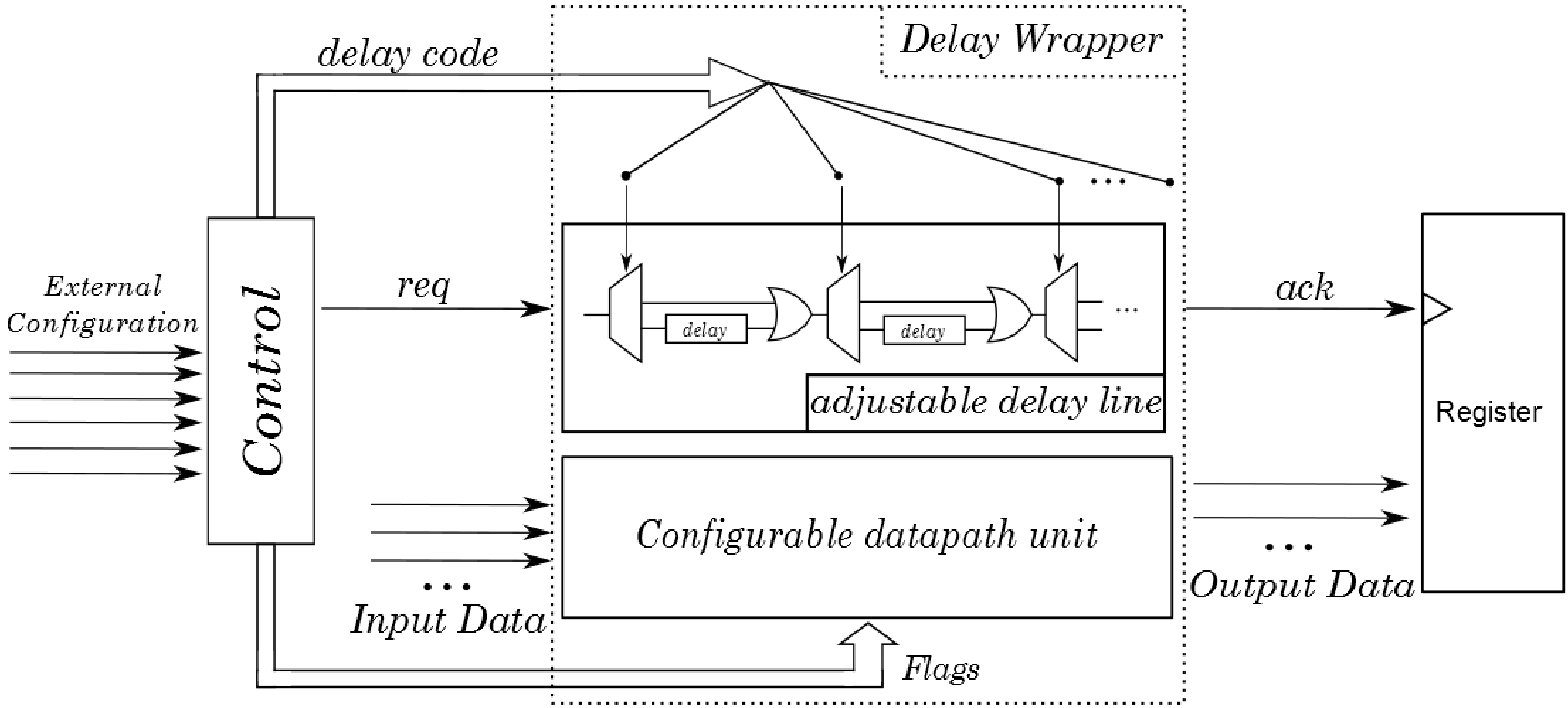
a
0
z
0
b
0
c
0
d
0
e
1
e*(b*!c*!d+a*d)+!e*(b*c*!d+!a*!b*!c)
e
!c*d*e
PCIU/3
PCIU/2
PCIU/3: a*!d*e+c*(b*d*!e*!z+z*(!b*d*!e+b*!d*e))
PCIU/2: b*d*z+e*(!a*(!c*d*z+!b*!z)+!d*(c+!b))+!e*(d*(c+a)+a*z+b*!c+!b*c)
!b*e
!d+!e+!b*!c+b*c
a+b*c*d*e
!a*!c*!e
!b*!c*d*!e
PCIU
IFU
IFU/3
ALU/2
done
e*(!d+!a)+a*!e+c
ALU
PCIU: !e+!d+!a
IFU: b*!z+c*z+!b*!c+!e+!d
!d*e+c*!e
IFU/3: !d*e*(b*c+a)+c*d*!e
ALU/2: a*(!z+e+d)+c*(b*d*!e*z+!z*(!b*d*!e+b*e))
!a*!d*!e
ALU: !c*(d*e*!z+!b)+c*(d*(z+!e)+b*e)
d*e+!c*!e
go
!a*!b*!c*e+!e*!b*!d+!e*c+b*d
IFU/2
!a+d*e+!d*!e
!d
d
!a*!b*!d+!c*!e+d*e
: !b*c*(!z+!d)+b*d*z+!e*(b*!c+a)+e*(c*!z+!b*!d+!a*d)
a*d*e+!a*(!b*!c*!e+!d*(e+!c))
!e
d*e
c*e+!c*!e
e
!e+!c*!d
MAU
ALU/3
a*!b*!c*d*e
d*!e+!c*!d
MAU/2
MAU: !d*(b*!c+!b*c)+d*(!e*(!c+!b)+!b*!c)+a
ALU/3: a*!d*e
c*e
!e+d
!c
MAU/2: a*(e+d)+!d*(b*!c*!e+!b*c)
!d+!e
!a*!b*c*!d*e
SIDU
SIDU/2
!a*b*!c*!d*e
SIDU: !b*c*!d+a*(e+d)
SIDU/2: a*(e+d)

Intel 8051 Datapath…



- ✓ Fully asynchronous implementation (bundled data protocol)
  - – adjustable delay lines
- ✓Fault tolerance operation
- ✓DFT integration

Adaptable datapath unit

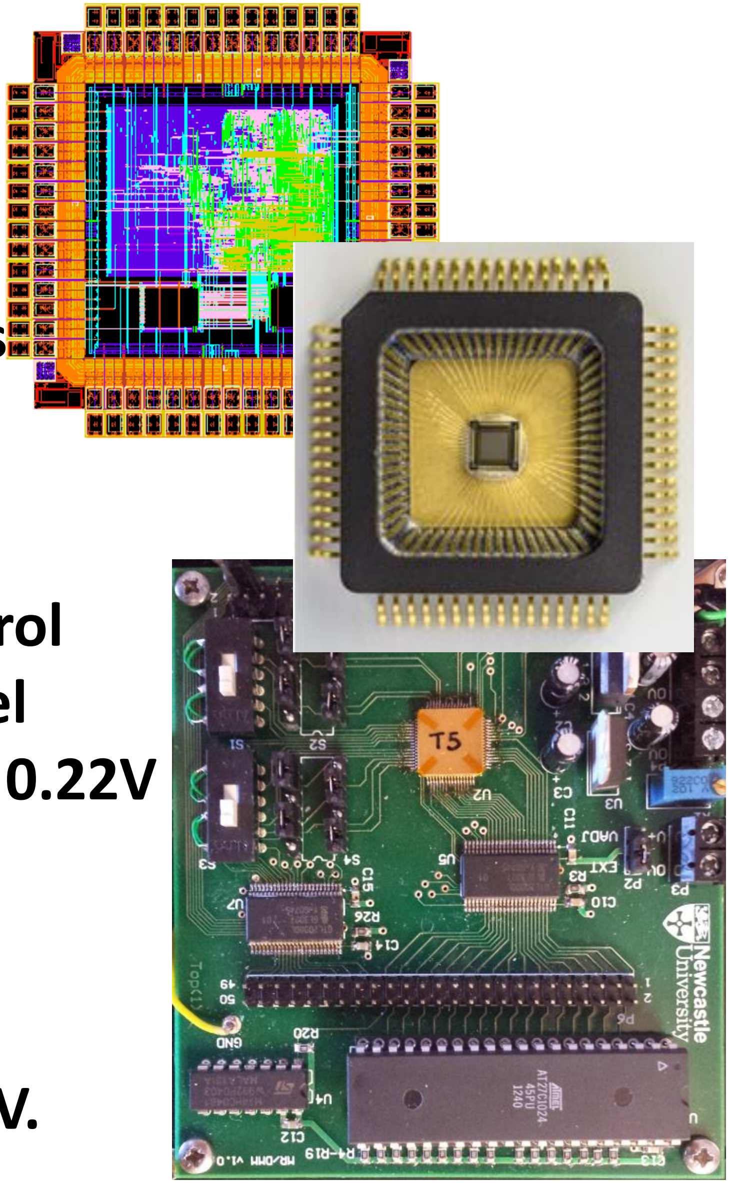


Some measurements…

- 0.89V to 1.5V: full capability mode.

- 0.74V to 0.89V: at 0.89V the RAM starts to fail, so the chip operates using

- 0.22V to 0.74V: at 0.74V the program counter starts to fail, however the control logic synthesised using the CPOG model continues to operate correctly down to 0.22V



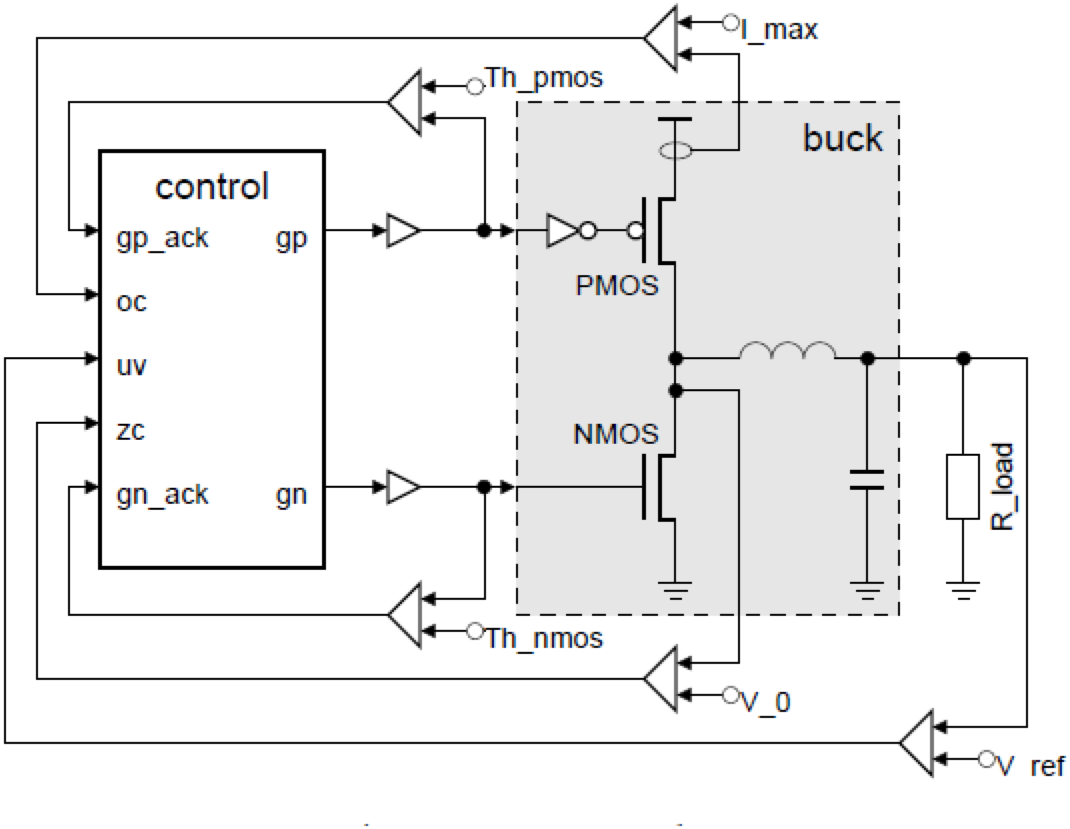
- 67 MIPS at 1.2 V.

- ~2700 instructions per second at 0.25V.

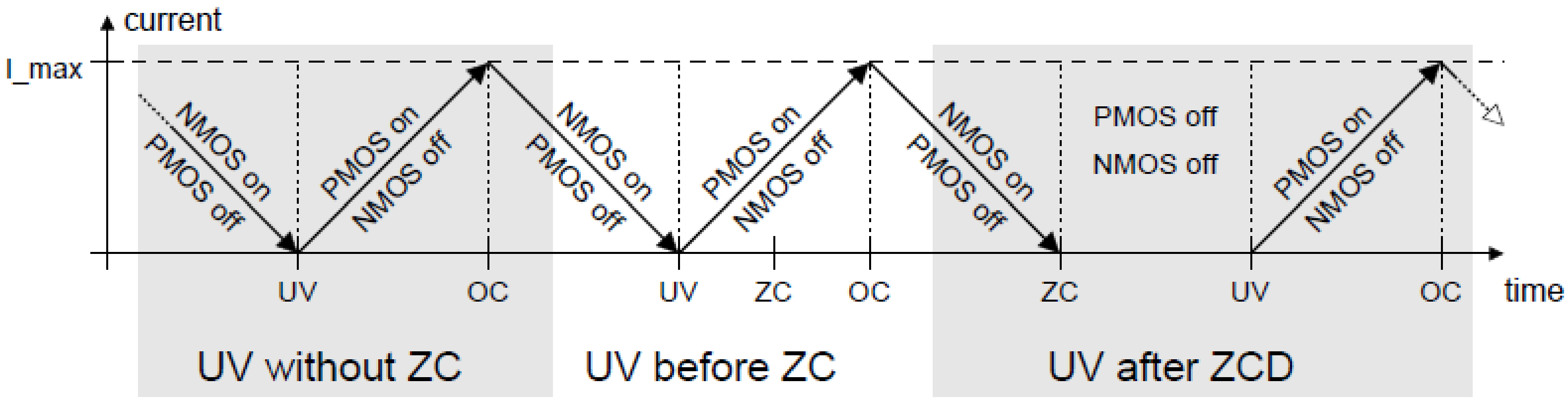
Future of Real-Power Design

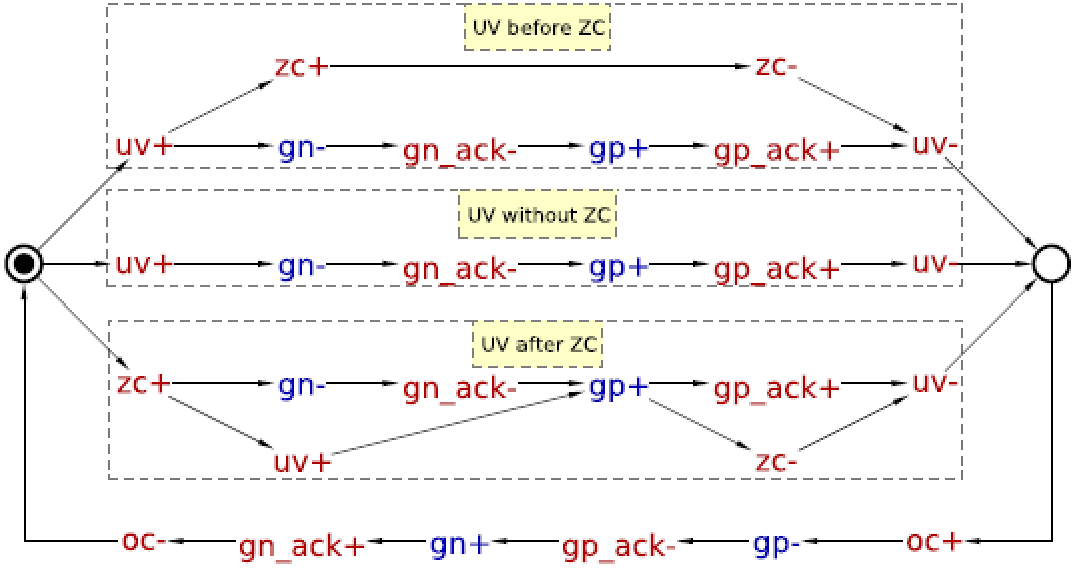
TODAY

Synthesis of asynchronous control DC-DC converters (A4A project)



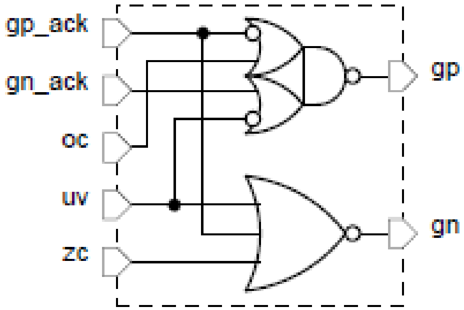
Asynchronous control for Bucks



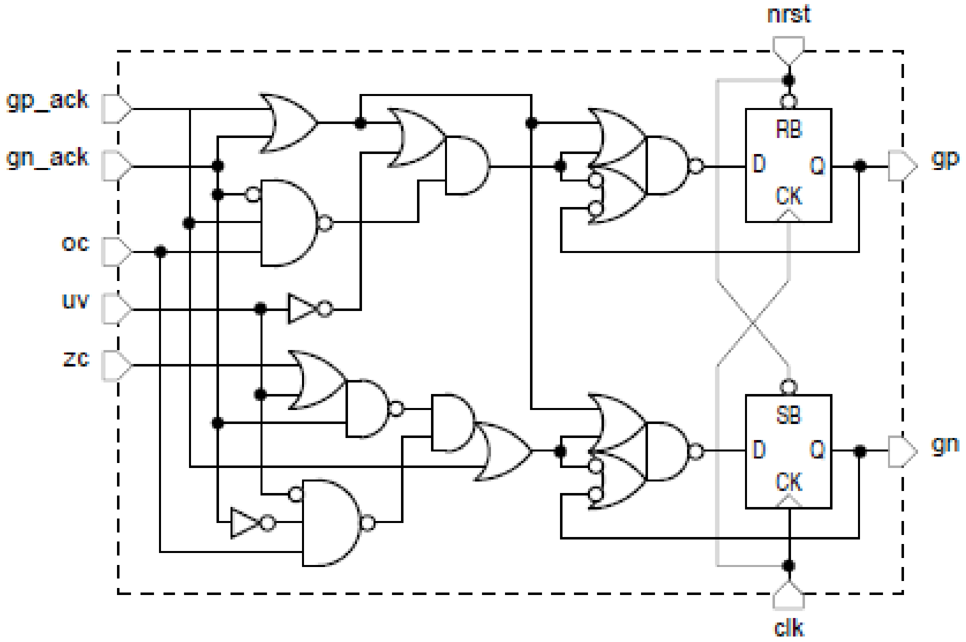


(b) Complete specification

Asynchronous control for Bucks



(a) Complex gate asynchronous implementation

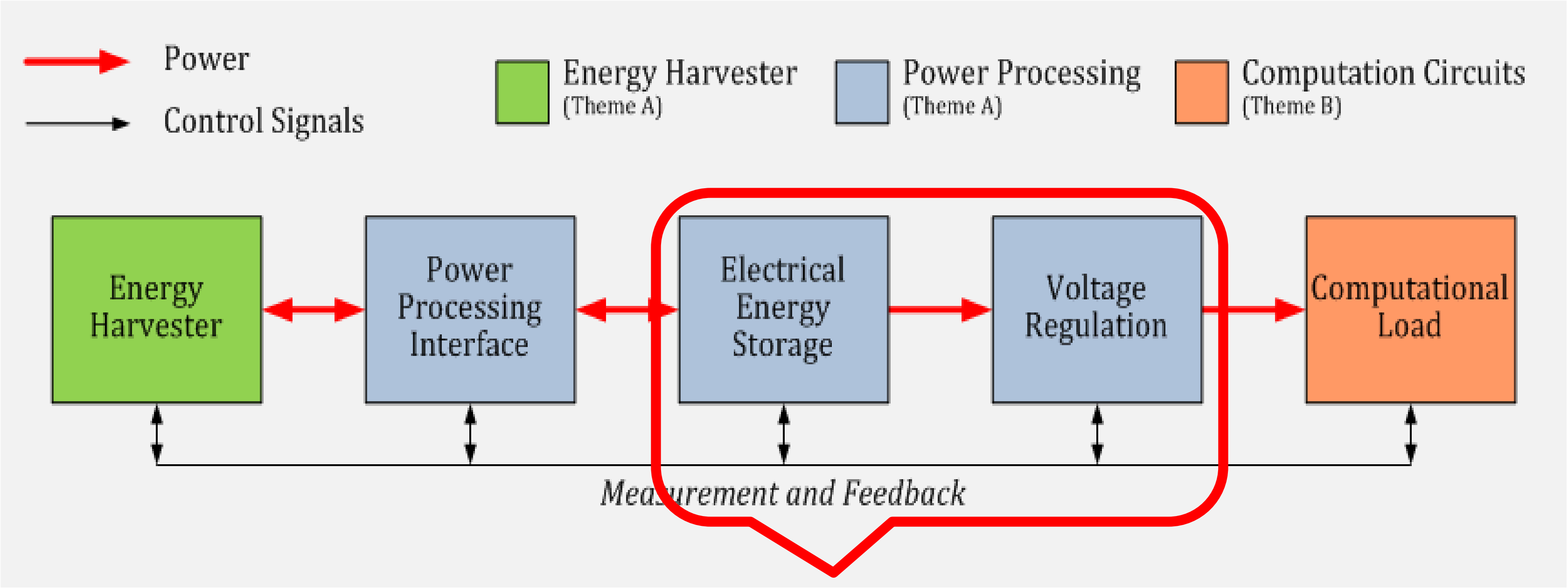


(b) Synchronous implementation

TODAY

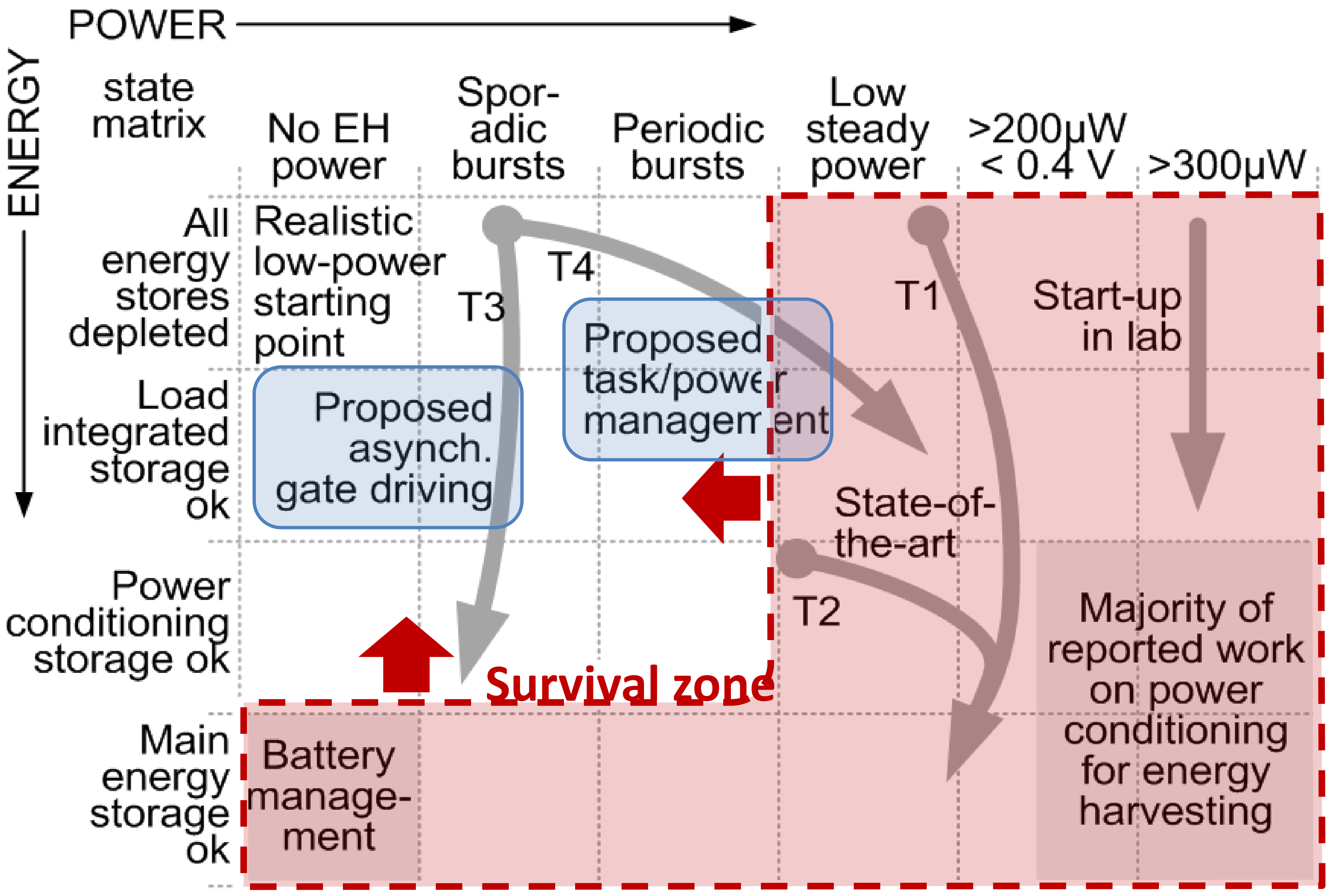
Energy harvesting systems

Using Holistic project vision:

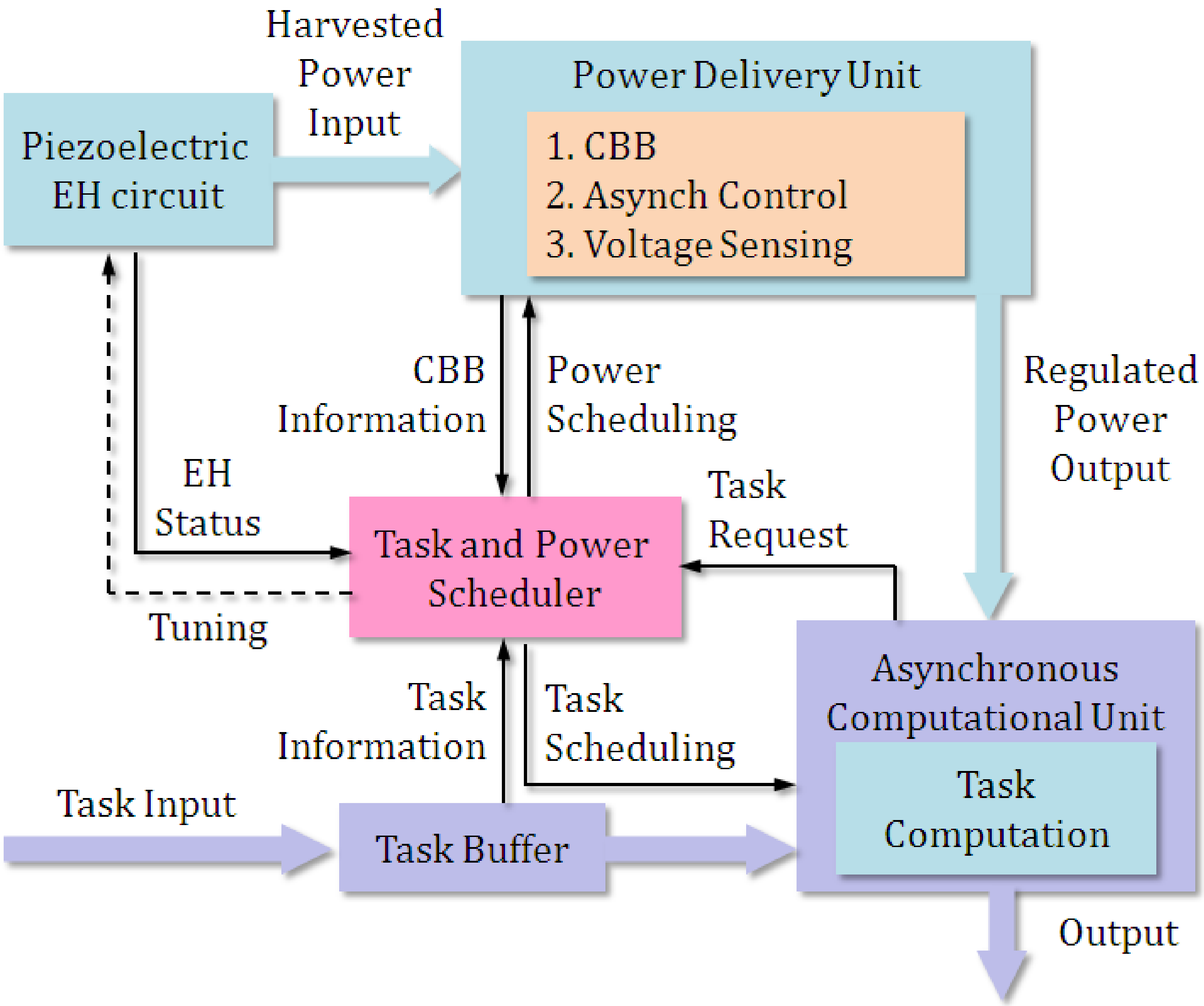

Power
Control Signals
Energy Harvester (Theme A)
Power Processing (Theme A)
Computation Circuits (Theme B)
Energy Harvester
Power Processing Interface
Electrical Energy Storage
Voltage Regulation
Computational Load
Measurement and Feedback

Sporadic source of energy does not allow for fancy power processing and therefore large storage

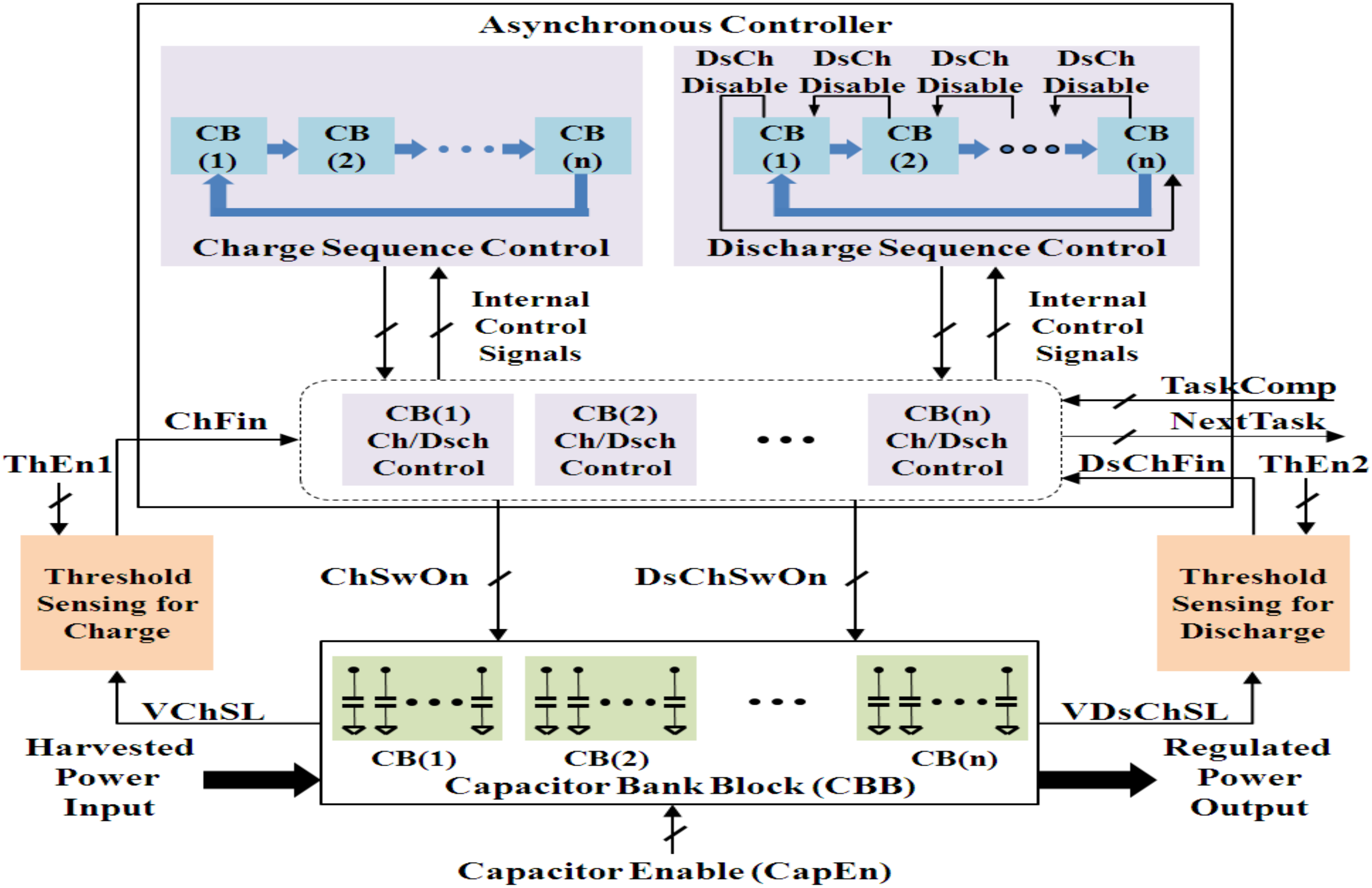
Staying alive in variable, intermittent, low-power environments (Savvie Project)


POWER
ENERGY
state matrix
No EH power
Spor-adic bursts
Periodic bursts
Low steady power
>200µW < 0.4 V
>300µW
All energy stores depleted
Load integrated storage ok
Power conditioning storage ok
Main energy storage ok
Realistic low-power starting point
T3
T4
T1
T2
Start-up in lab
Proposed asynch. gate driving
Proposed task/power management
State-of-the-art
Survival zone
Battery manage-ment
Majority of reported work on power conditioning for energy harvesting

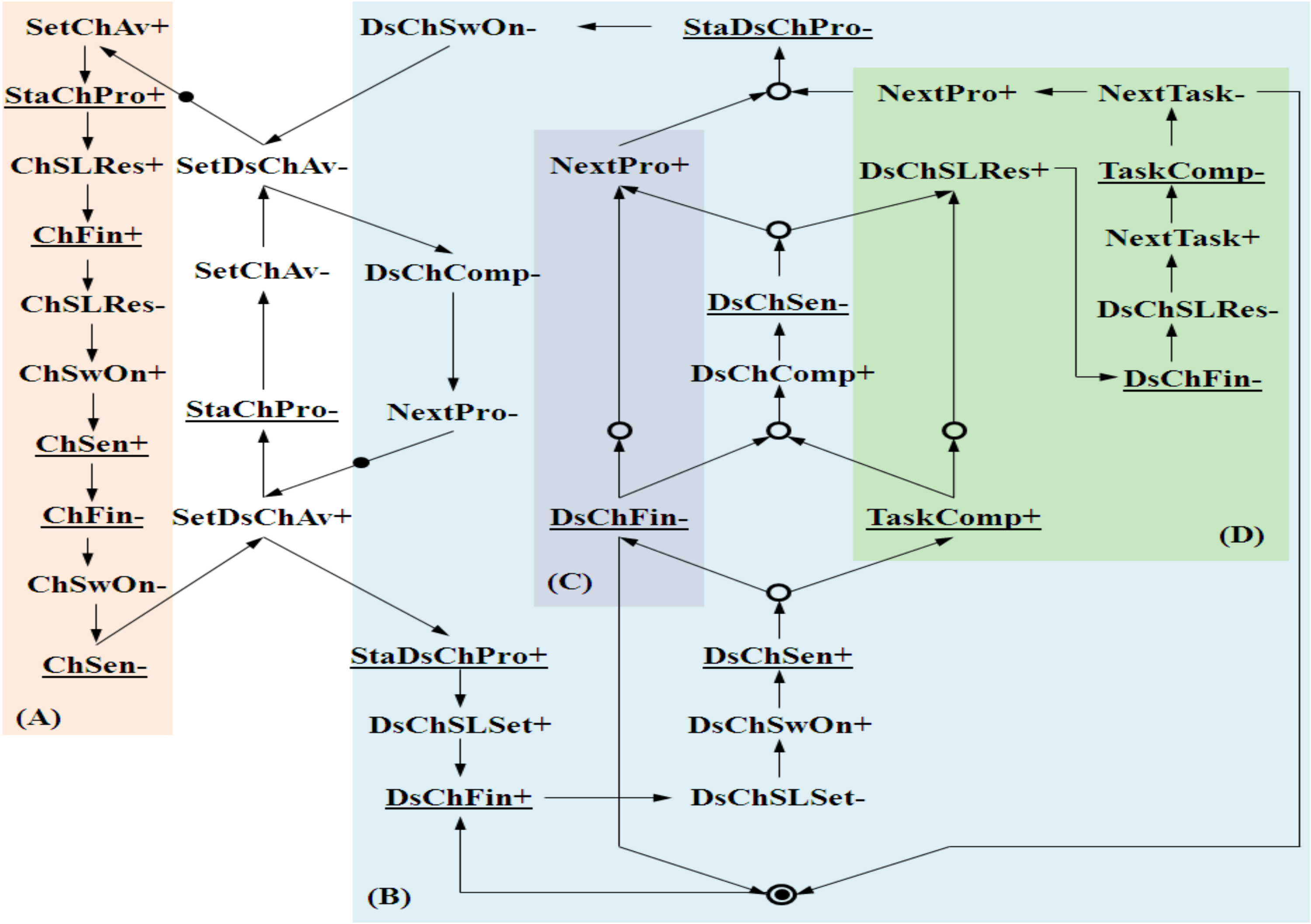
Our View on EHA Systems



Asynchronous Control for Capacitor Banks

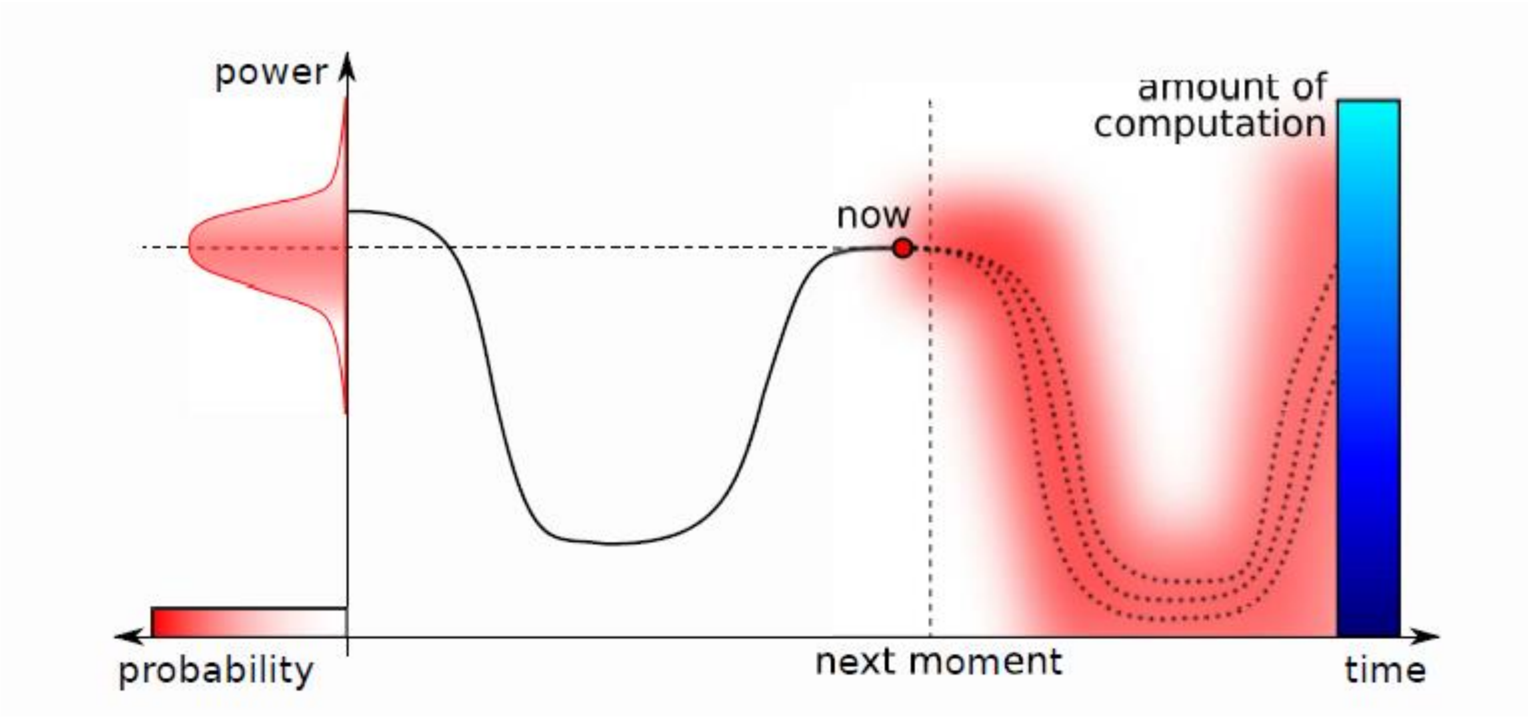


Asynchronous Controller: Signal Transition Graph Spec



Power-modulation and uncertainty

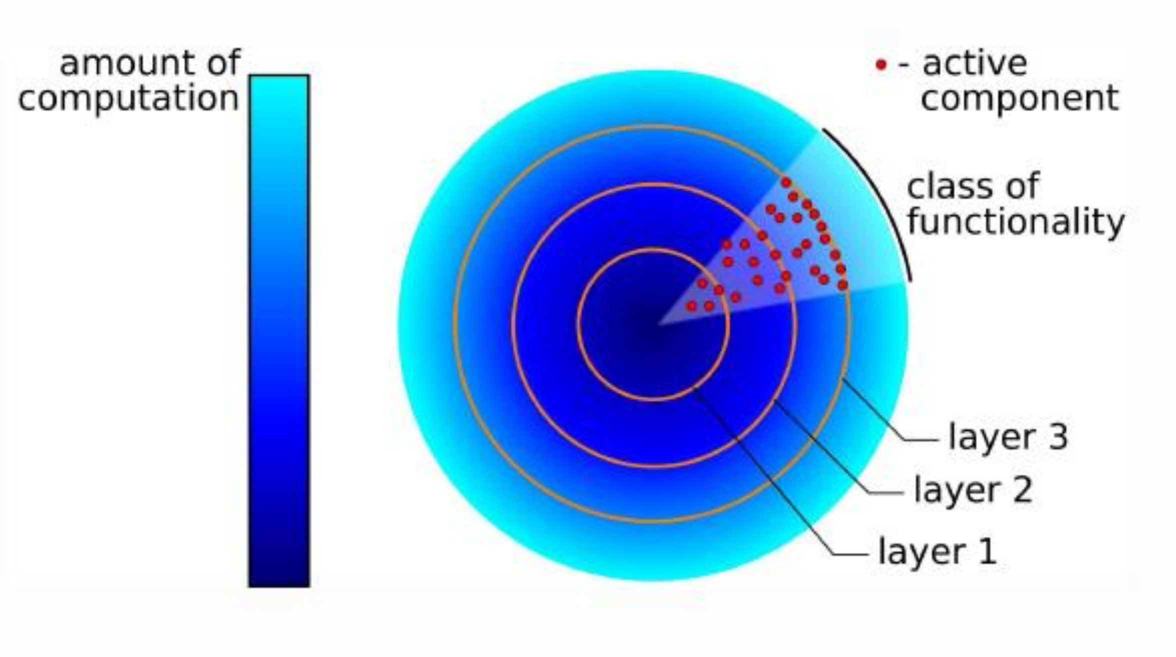
- Localised prediction, from every moment at present
- Power has a certain profile (time trajectory) in the past and uncertain future
- Power-proportional and power-adaptive systems …



TOMORROW

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



Acknowledgements

- My colleagues at Newcastle and outside: Andrey Mokhov, Danil Sokolov, Fei Xia, Delong Shang, Maxim Rykunov, Reza Ramezani, Alex Kushnerov (Ben Gurion Uni), Bernard Stark (Bristol Uni) and many others – see http://async.org.uk
- EPSRC support: Dream Fellowship, projects: Holistic, PRiME, Power-Prop, Savvie, A4A
- Industrial support: ARM CASE studentship, Dialog Semiconductor

THANK YOU!