EE249
Embedded System Design: Models, Validation and Synthesis – Introduction, Part 2

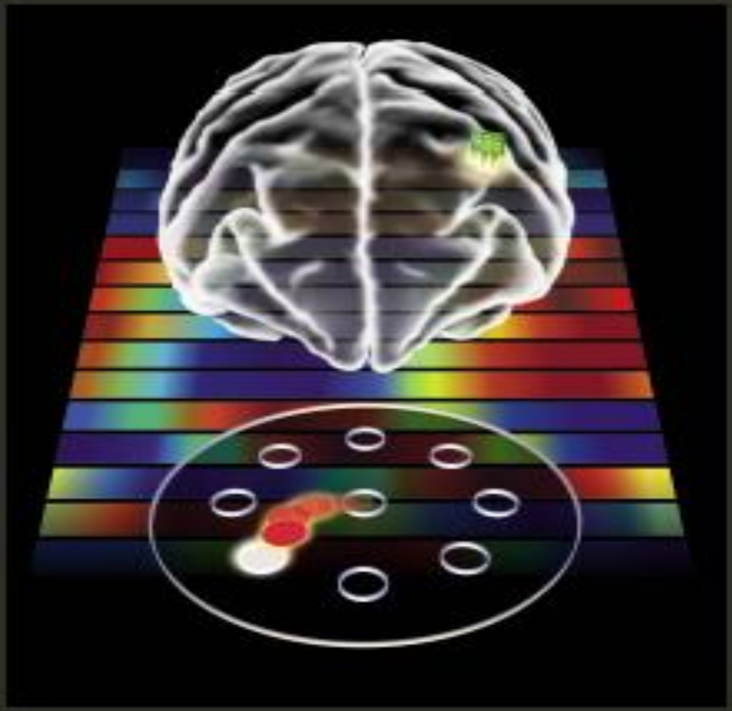
Alberto Sangiovanni Vincentelli

Outline

- Evolution of IT Systems
- Cyber-physical Systems
  - Societal Scale Systems
  - Automobile of the future
  - Smart grid and buildings
- The Far Future
  - Bio-Cyber Systems
- Design Challenges

Another One: BioCyber (?) Systems

Linking the Cyber and Biological Worlds

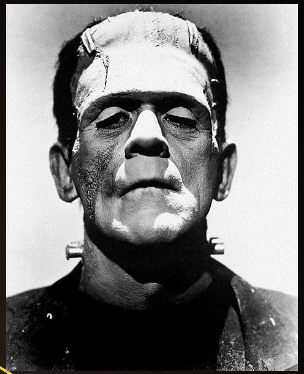




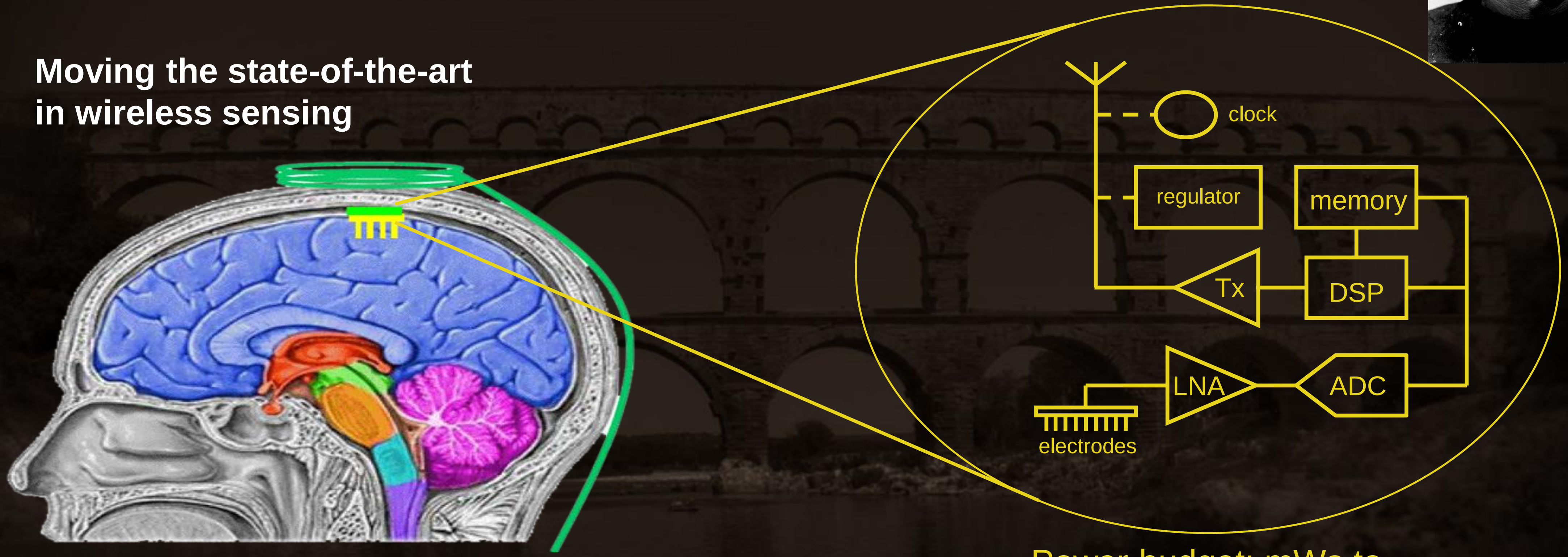
Examples: Brain-machine interfaces and body-area networks

Courtesy: J. Rabaey

Towards Integrated Wireless Implanted Interfaces



Moving the state-of-the-art in wireless sensing

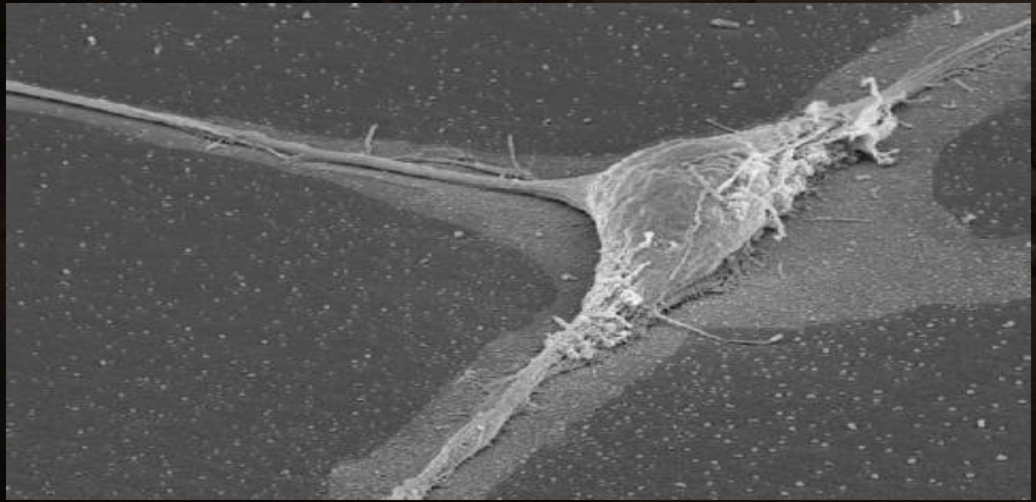


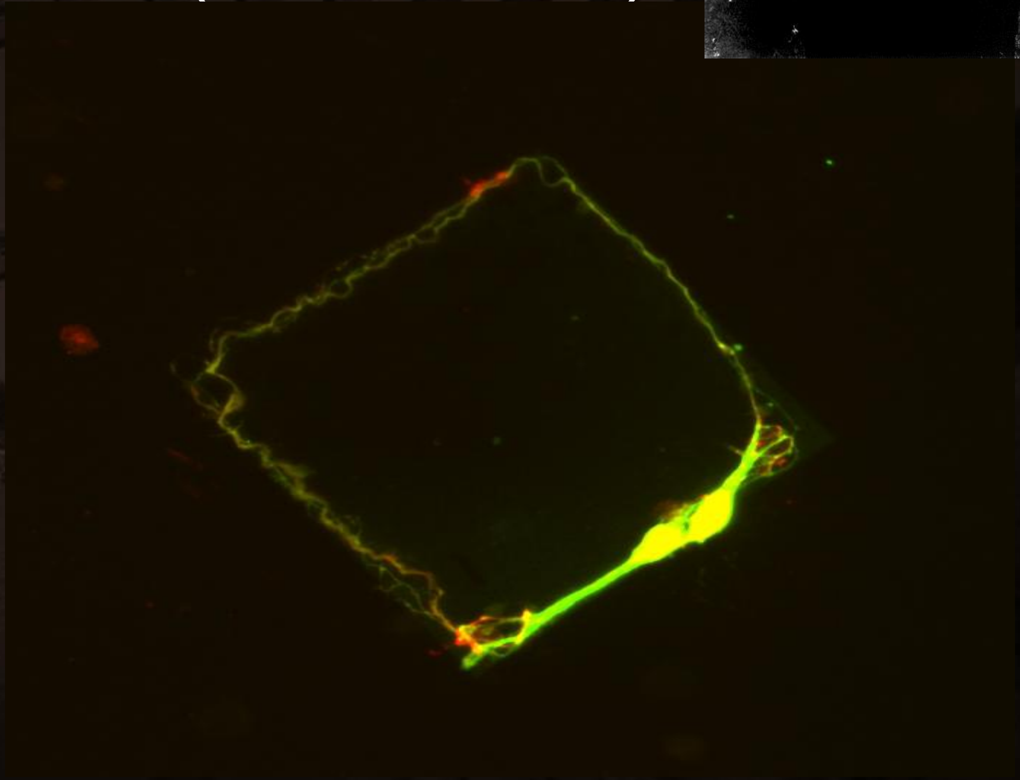
[Illustration art: Subbu Venkatraman]

Power budget: mWs to 1 mW

Engineering Tomorrow's Designs:
Neurons drive Electro-Mechanical Systems
Italian Institute of Technology Genova Central Research Center
The Neuroscience Brain Technology Department – Fabio Benfenati's Group

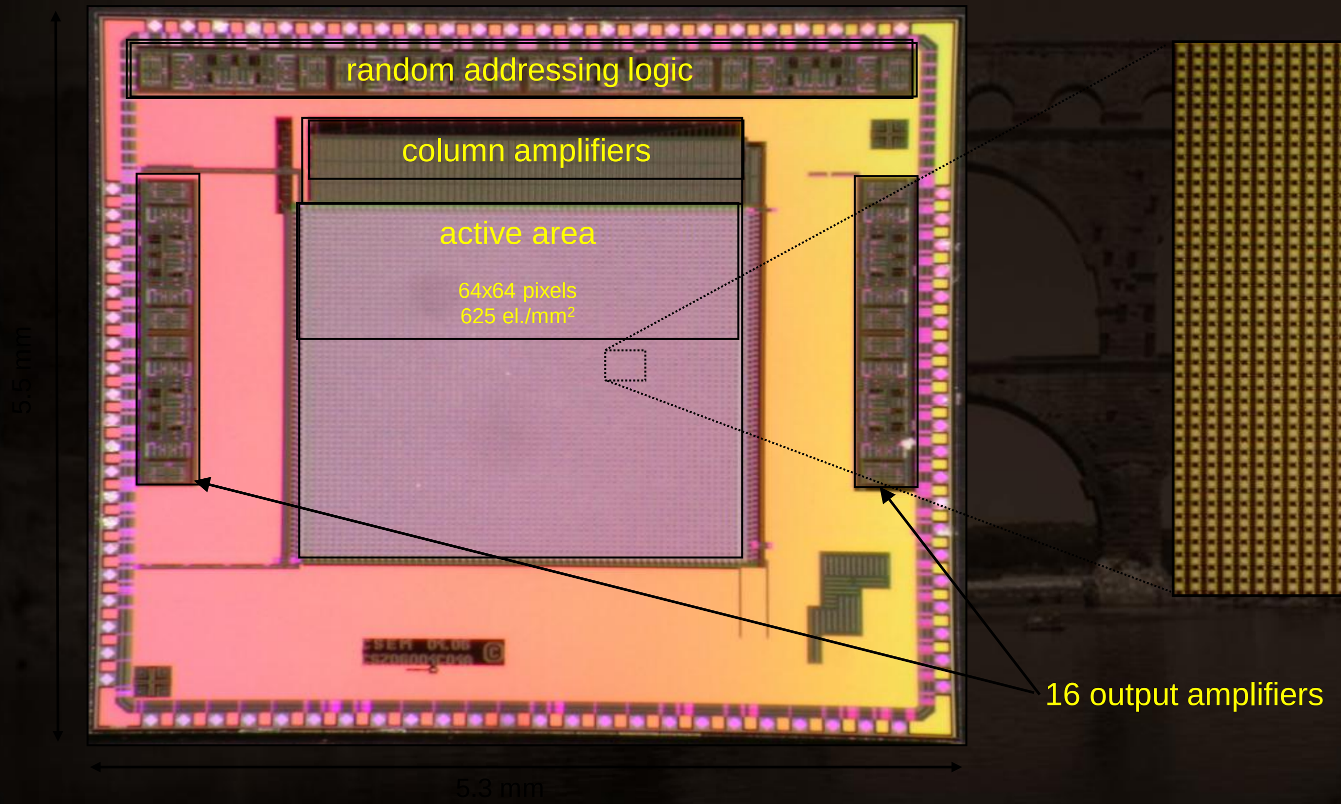
Generate spatially-ordered 2d and 3d neuronal (NON NEURAL) networks

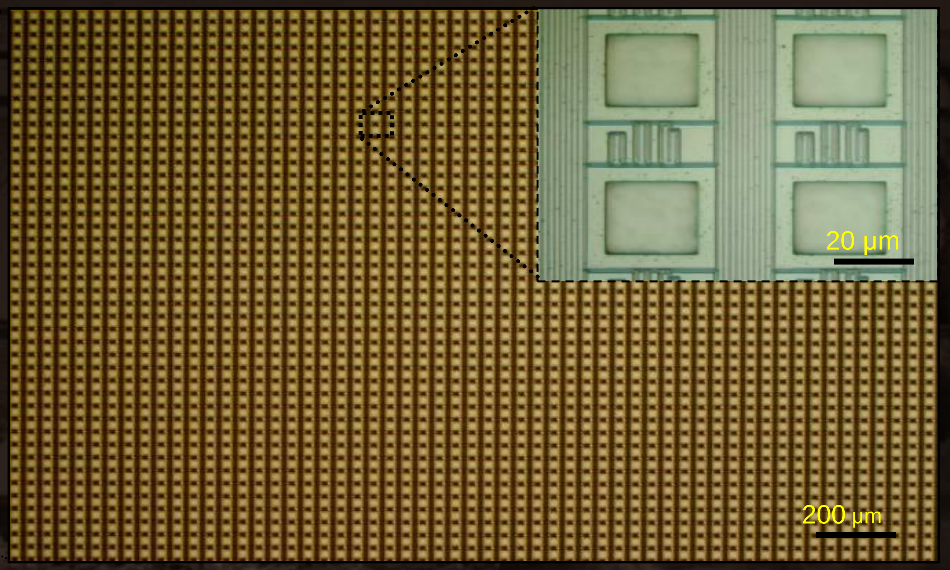




THE HIGH-RESOLUTION NEURON-TO-CHIP INTERFACE

Luca Berdondini

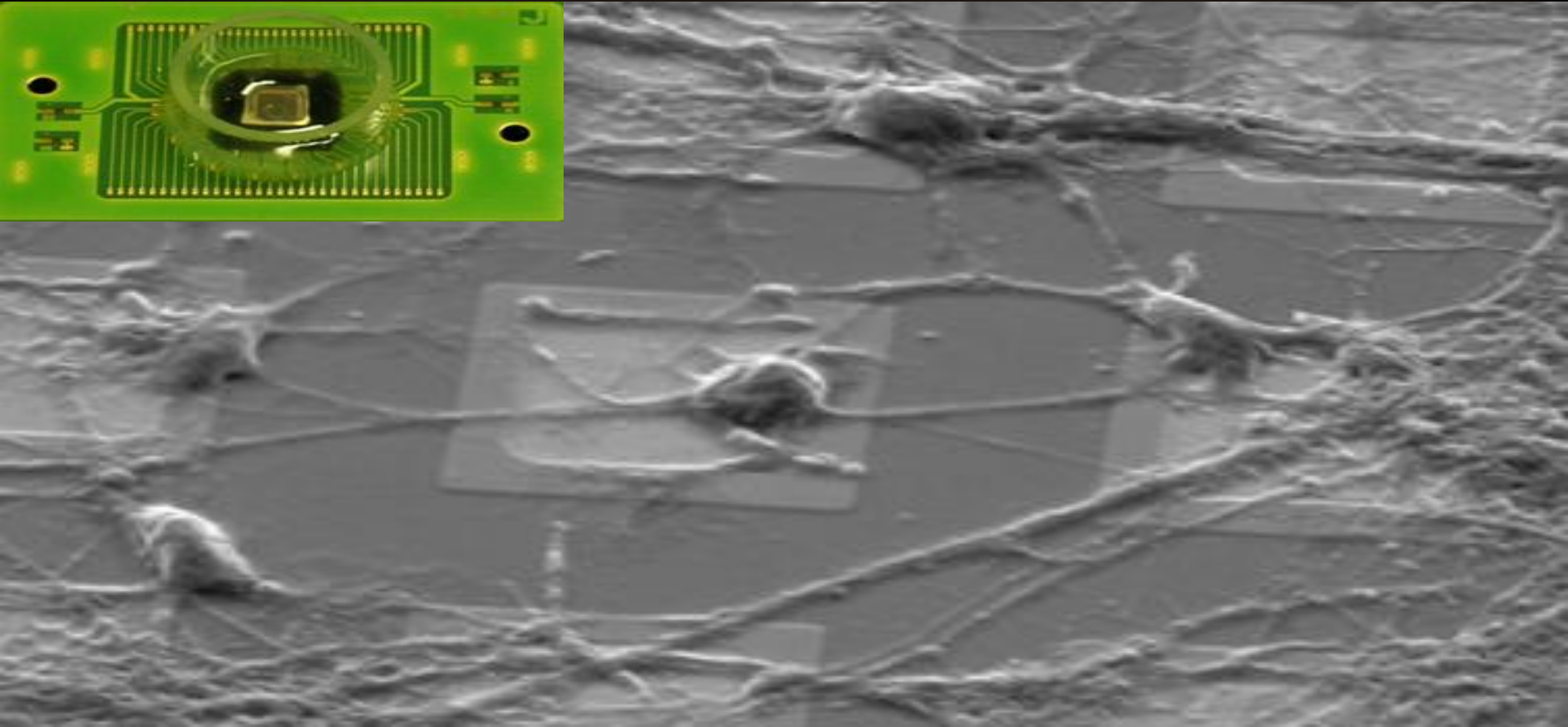




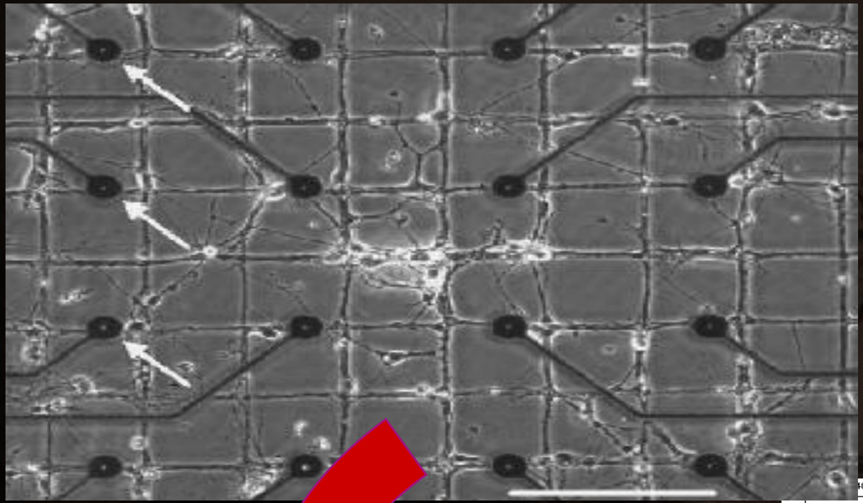
technology: 0.35 µm CMOS (4 metal-layer process by AMS)

- 625 electrodes per mm^2
- inter-electrode separation of 20 µm

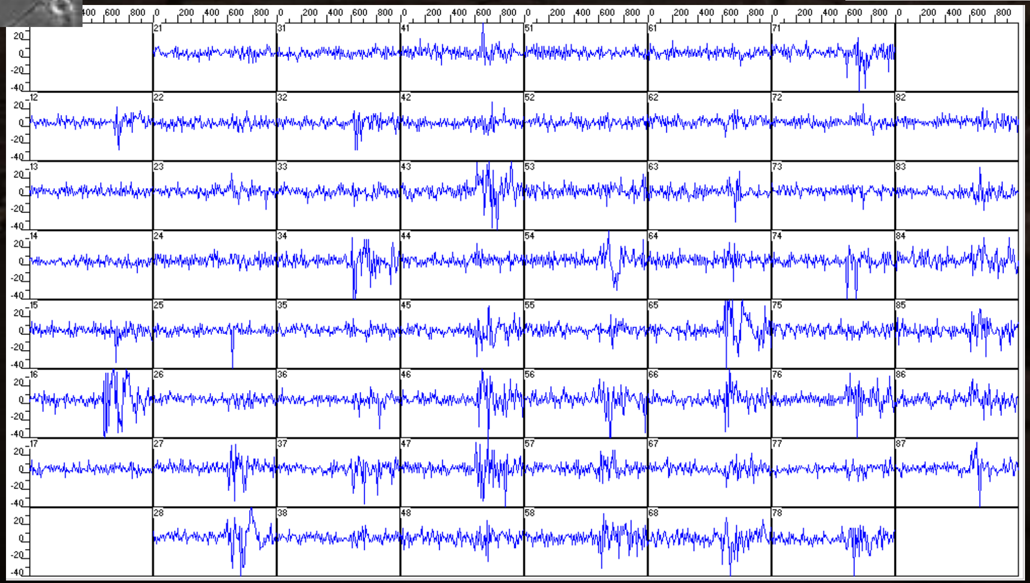
THE 4096 ELECTRODE SPATIAL RESOLUTION



NEURO-ROBOTIC INTERFACES:
from neuronal networks to an external body (Sergio Martinoia)



SENSORY STIMULATION
(experience)

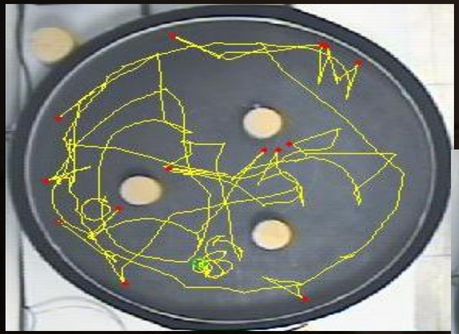


NEURAL COMPUTATION
(adaptation, plasticity, emerging properties)

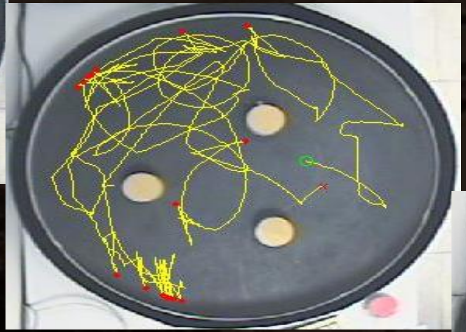
MOTOR COMMANDS
(purposeful behavior)

Obstacle avoidance task

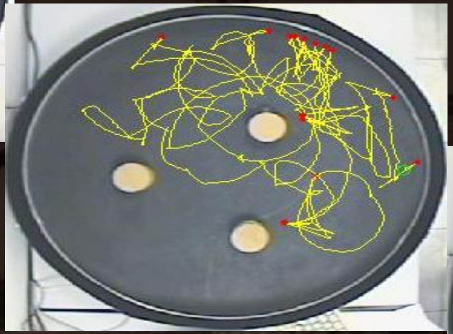
10 min per phase



Phase 1
Free running

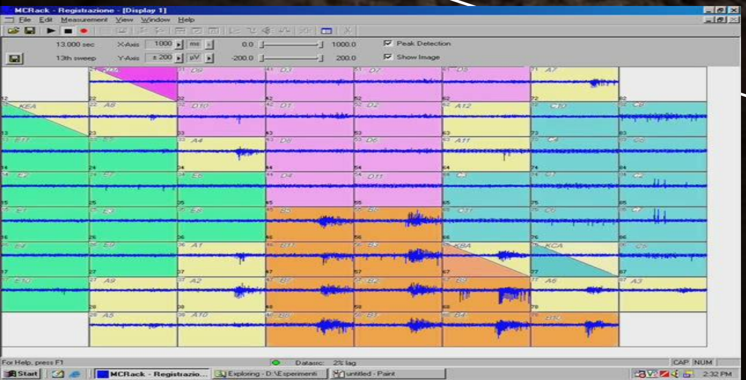


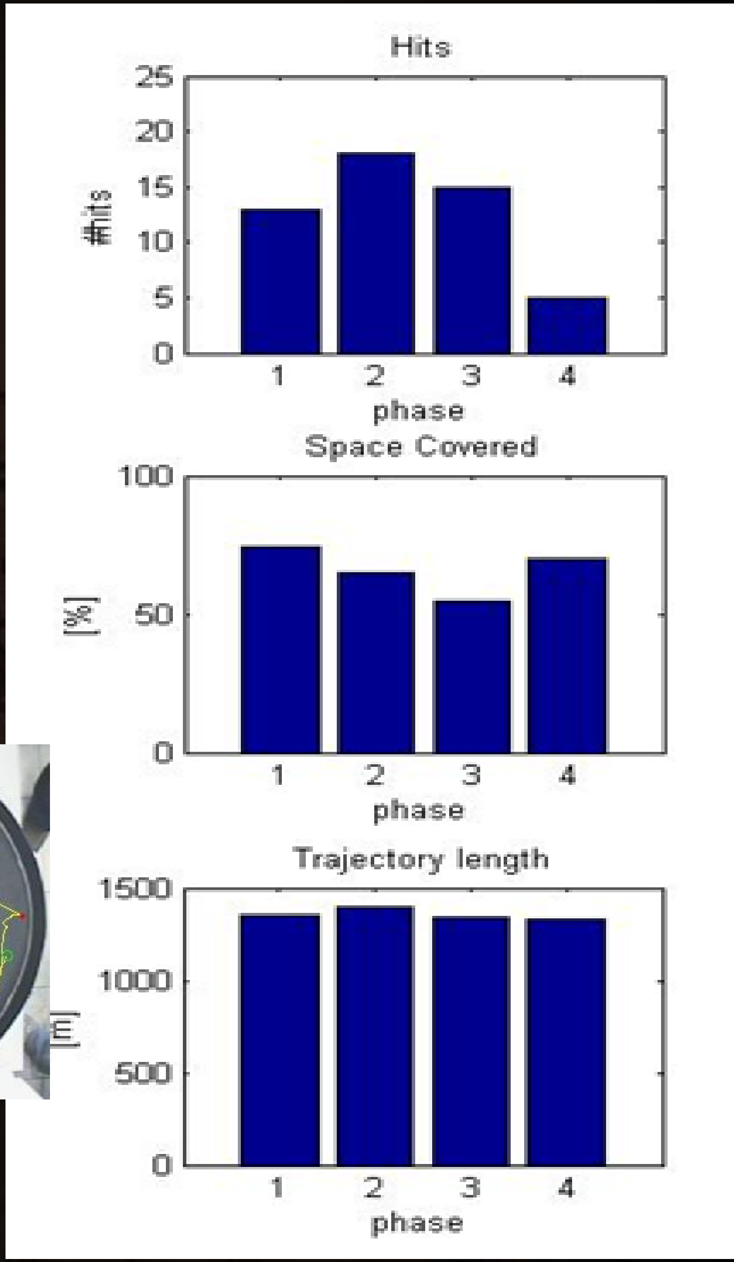
Phase 2
Learning



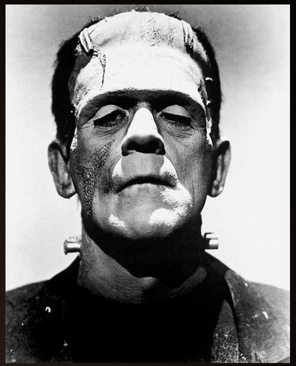
Phase 3,4
Avoidance

Phase 5
Free running

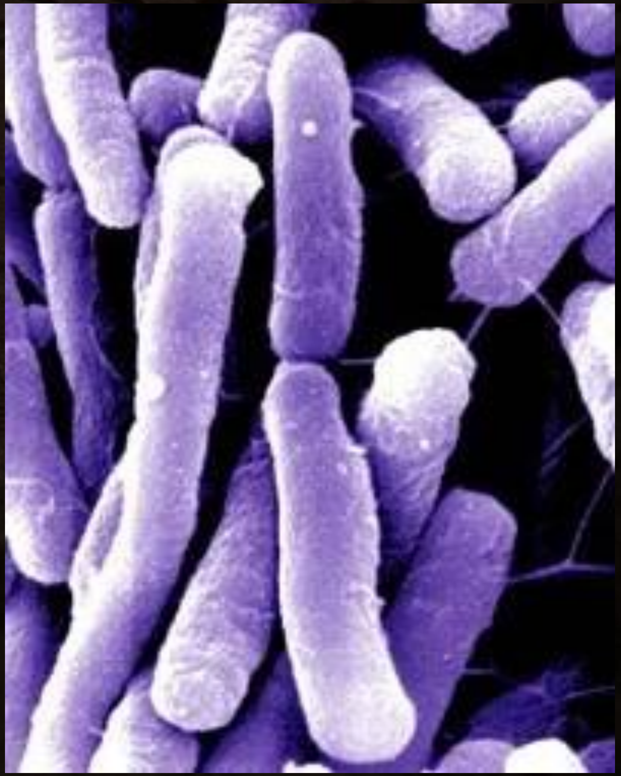




Synthetic Biology



συν•θη•σισ *n.* 1.a. the combination of separate elements to form a coherent whole.



- Synthetic biology seeks, through understanding, to design biological systems and their components to address a host of problems that cannot be solved using naturally-occurring entities
- Enormous potential benefits to medicine, environmental remediation and renewable energy

Engineering Tomorrow's Designs
Synthetic Biology

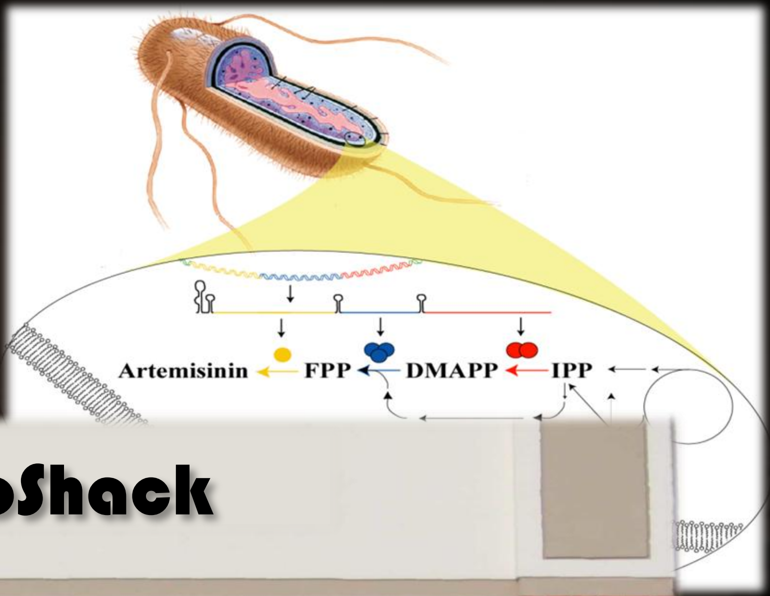
The creation of novel biological functions and tools by modifying or integrating well-characterized biological components into higher-order systems using mathematical modeling to direct the construction towards the desired end product.

Building life from the ground up (Jay Keasling, UCB), Keynote presentation, World Congress on Industrial Biotechnology and Bioprocessing, March 2007.

Development of foundational technologies:

- Tools for hiding information and managing complexity
- Core components that can be used in combination reliably

Microbial Synthesis of Artemisinin



Off-the-shelf parts?

HMGS

atoB

tHMGR

MK

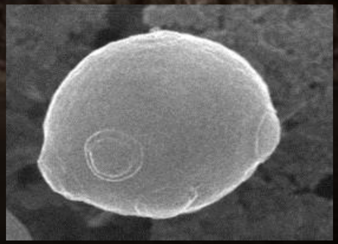
AcCoA

HMG-CoA

AcAcCoA

Mev

Mev



CPR

Artemisinin

Courtesy: Jay Keasling

Applications of Synthetic Biology

Energy Crop

- Water saving
- No fertilizer
- Doubled photosynthetic efficiency

Biodiesel and bio-jet fuel

- No compromise
- Fully compatible with existing infrastructure

Natural product drugs

- Capture all of the chemistry in nature
- Construct a microbe that can produce any natural product

Courtesy: Jay Keasling

Amyris

- Amyris had its technological foundation in 2001 in the Keasling lab at Berkeley.
- “Keasling’s magic bug, genetically enhanced from a soup of DNA obtained from bacteria and the plant world, is a five-carbon base chemical and a high-value target in the world of what is now known as the field of renewable chemicals — its a path to isoprenoids, which are themselves a family of some 50,000 molecules that have applications or pathways for pharmaceuticals, fragrances, cosmetics and fuels.”
- Keasling filed the patent in 2001, and Amyris itself was eventually formed and funded by 2006 with $14.1 million in Series A investments from Kleiner Perkins and Khosla Ventures among other early backers.

IPO in 4° Quarter 2010
From 680Mil cap to 1.265Bil today

renewable products for the world

AMYRIS IS APPLYING AN INDUSTRIAL SYNTHETIC BIOLOGY PLATFORM TO PROVIDE HIGH-PERFORMING ALTERNATIVES TO PETROLEUM-SOURCED FUELS AND CHEMICALS

Total and Amyris Partner to Produce Renewable Fuels

Total and Amyris strategic partnership expanded to accelerate development and marketing of renewable fuels

PARIS, France and EMERYVILLE, Calif.-- November 30, 2011 - Total (CAC: TOTF.PA) and Amyris, Inc. (NASDAQ: AMRS) signed agreements to expand their current R&D partnership and form a joint venture to develop, produce and commercialize a range of renewable fuels and products.

Total and Amyris have agreed to expand their ongoing research and development collaboration to accelerate the deployment of Biofene® and develop renewable diesel based on this molecule produced from plant sugars. The ambitious R&D program, launched in 2010 and managed jointly by researchers from both companies, aims to develop the necessary stages to bring the next generation renewable fuels to market at commercial scale. Total has committed to contribute $105 million in funding for an existing $180 million program.

In addition, Total and Amyris have agreed to form a 50-50 joint venture company that will have exclusive rights to produce and market renewable diesel and jet fuel worldwide, as well as non-exclusive rights to other renewable products such as drilling fluids, solvents, polymers and specific biolubricants. The venture aims to begin operations in the first quarter of 2012.

Engineered Superbugs Boost Hopes Of Turning Seaweed Into Fuel

SCIENCE VOL 335 20 JANUARY 2012

Outline

- Evolution of IT Systems
- What is possible? Cyber-physical Systems
  - Societal Scale Systems
  - Automobile of the future
  - Smart grid and buildings
- The Far Future
  - Bio-Cyber Systems
- Design Challenges

How Safe is Our Design Today?

The Larger Picture

PAGE 14 – SUNDAY, FEBRUARY 6, 2005 – THE NEW YORK TIMES (by Tim Mo

What's Bugging the High-Tech Ca

On a hot summer trip to Cape Cod, the Mills family minivan did a peculiar thing. After an hour on the road, it began to bake the children. Mom and Dad were cool and comfortable up front, but heat was blasting into the rear of the van and it could not be turned off.

Fortunately for the Mills children, their father – W. Nathaniel Mills III, an expert on computer networking at I.B.M. – is persistent. When three dealership visits, days of waiting and the cumbersome replacement of mechanical parts failed to fix the problem, he took the van out and drove it until the oven fired up again. Then he rushed to the mechanic to look for a software error.

Additionally, the study found that altho errors cannot be removed, more than took two minutes for them to hook up diagnostic tool and find the fault," said Mills, senior technical staff member at I.B.M.'s T.J. Watson Research Center Hawthorne, N.Y. "I can almost see the software code; a sensor was bad."

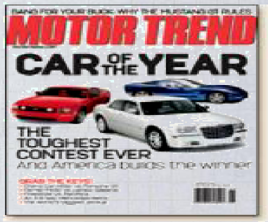
Indeed, the high-tech comfort system confuse the 200 sendin freezin loyal va up. thir billion,

MOTOR T

NHTSA To Probe Reports Of Sudden Engine Stalls In Prius Hybrids

The National Highway Traffic Safety Administration said yesterday it is investigating reports that a software problem can cause the engine of Toyota's Prius hybrid to stall without warning at highway speeds. No accidents have been reported thus far.

NHTSA has received 33 reports of stalling in Prius cars from model years 2004 and 2005, according to the agency's initial report. More than 85 percent of the cars that stalled did so at speeds between 35 and 65 miles per hour.



Toyota Problems

The Washington Post, March 7

Attention has been focused on mechanical and electronic issues with Toyotas, but another possible cause of the runaway acceleration maybe a software glitch. Each vehicle contains layers of computer code that may be added from one model year to next" that control nearly every system, from acceleration to braking to stability. This software is rigorously tested, but t is well-known in our community that there is no scientific, firm way of actually completely verifying and validating software.

The Problem: Typical Car Electrical Architecture

And What About Airplanes?

Airbus Problems

Initial production of the A380 was troubled by delays attributed to the 530 km (330 mi) of wiring in each aircraft. Airbus cited as underlying causes the complexity of the cabin wiring (100,000 wires and 40,300 connectors), its concurrent design and production, the high degree of customization for each airline, and failures of configuration management ... change control... manufactured using aluminum rather than ... rules including non-

Boeing Problems

Boeing had originally planned for a first flight by the end of August 2007 and premiered the first 787 at a rollout ceremony on July 8, 2007, which matches the aircraft's designation in the US-style month-day-year format (7/8/07). Although intended to shorten the production process, 787 subcontractors initially had difficulty completing the extra work, because they could not procure the needed parts, perform the subassembly on schedule, or both, leaving remaining assembly work for Boeing to complete as "traveled work". blaming a shortage of fasteners as well as incomplete software. The company expects to write off US$2.5 billion because it considers the first three Dreamliners built are unsellable and suitable only for flight tests. In August 2010, it was announced that Boeing was facing a US$1 billion compensation claim from Air India due to the delays for the 27 787s it has on order

It’s Not Over Yet!

THE WALL STREET JOURNAL.
WSJ.com

BUSINESS | NOVEMBER 25, 2010

Boeing 787 Is Set Back as Blaze Forces Fix

By PETER SANDERS

How is Embedded Software Different from Ordinary Software?

- It has to work

- One or more (very) limited resources
  - Registers
  - RAM
  - Bandwidth
  - Time

Source: Alex Aiken

Devil's Advocate

- So what's different?

- All software works with limited resources

- We have compiler technology to deal with it
  - Various forms of program analysis

Source: Alex Aiken

Example: Registers

- All machines have only a few registers
- Compiler uses the registers as best as it can
  - *Spills* the remaining values to main memory
  - Manages transfers to and from registers
- The programmer feels she has ∞ registers

Source: Alex Aiken

The Standard Trick

- This idea generalizes
- For scarce resource X
  - Manage X as best as we can
  - If we need more, fall back to secondary strategy
  - Give the programmer a nice abstraction

Source: Alex Aiken

The Standard Trick

- This idea generalizes

- For scarce resource X
  - Manage X as best we can
  - *Any correct heuristic is OK, no matter how complex*

  - If we need more, fall back to secondary strategy
  - *Focus on average case behavior*

  - *Give the programmer a nice abstraction*

Source: Alex Aiken

Examples of the Standard Trick

- Compilers
  - Register allocation
  - Dynamic memory management
- OS
  - Virtual memory
  - Caches

Summary: abstract and hide complexity of resources

Source: Alex Aiken

What's Wrong with This?

- Embedded systems have limited resources
- Meaning hard limits
  - Cannot use more time
  - Cannot use more registers
- The compiler must either
  - Produce code within these limits
  - Report failure
- The standard trick is anathema to embedded systems
  - Can't hide resources

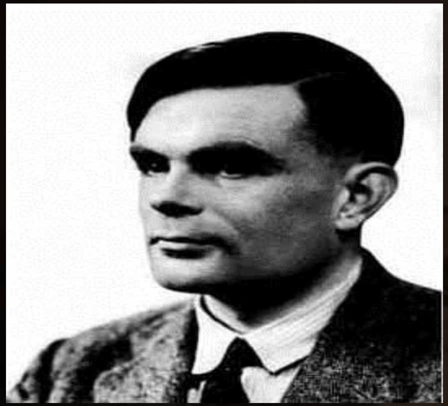
Source: Alex Aiken

Revisiting the Assumptions

- ***Any correct heuristic is OK, no matter how complex***
  - Embedded programmer must understand reasons for failure
  - Feedback must be relatively straightforward
- ***Focus on average case behavior***
  - Embedded compiler must reason about the worst case
  - Cannot improve average case at expense of worst case
- ***Give the programmer a nice abstraction***
  - Still need abstractions, but likely different ones

Source: Alex Aiken

Another Traditional Systems Science - Computation, Languages, and Semantics



Alan Turing

Everything "computable" can be given by a terminating sequential program.

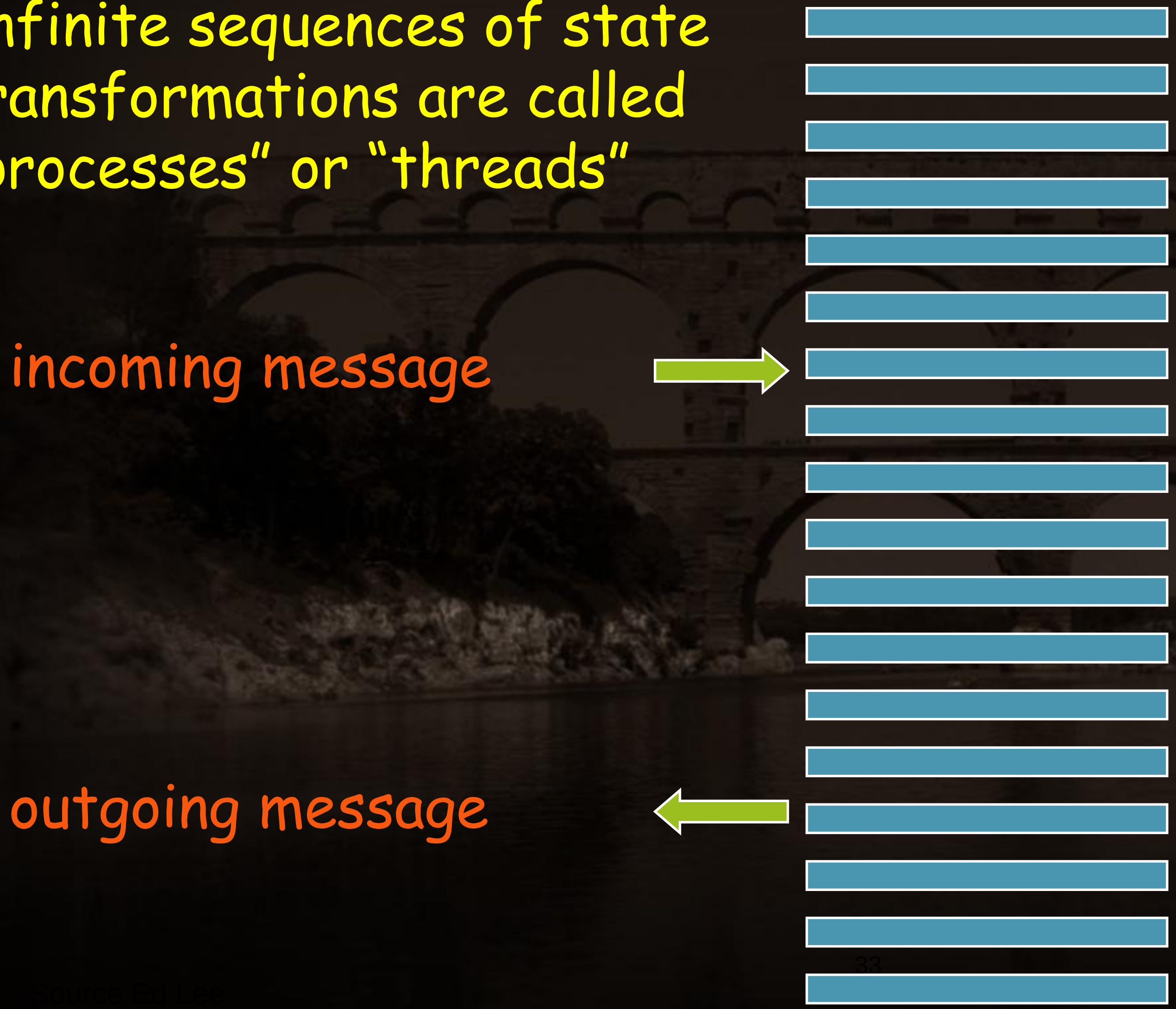
- Functions on bit patterns
- Time is irrelevant
- Non-terminating programs are defective

$States = Bits^*$

Source Ed Lee

Processes and Process Calculi

Infinite sequences of state transformations are called "processes" or "threads"



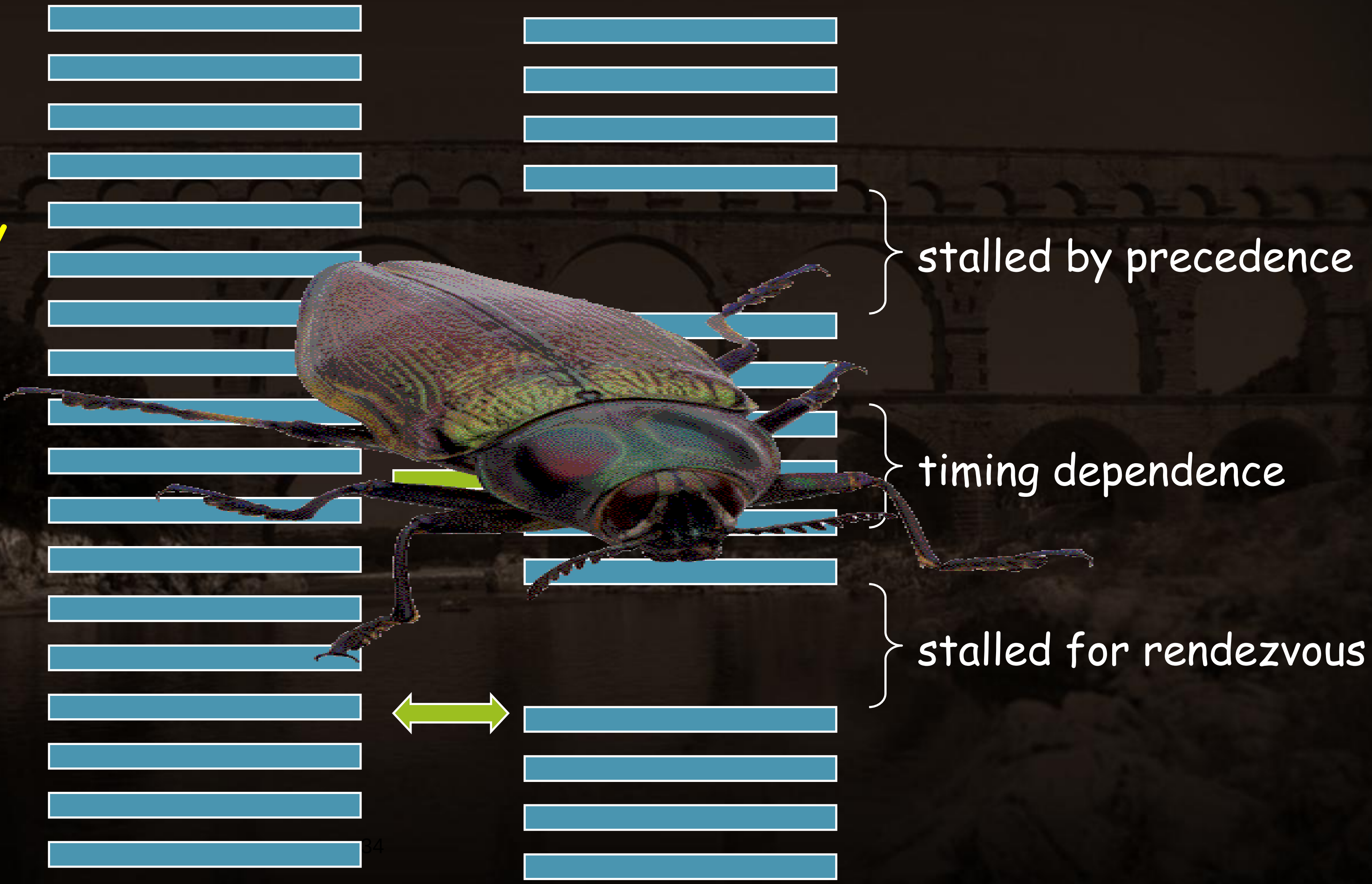
Various messaging protocols lead to various formalisms.

In prevailing software practice, processes are sequences of external interactions (total orders).

And messaging protocols are combined in ad hoc ways.

Source Ed Lee

Interacting Processes – Concurrency as Afterthought



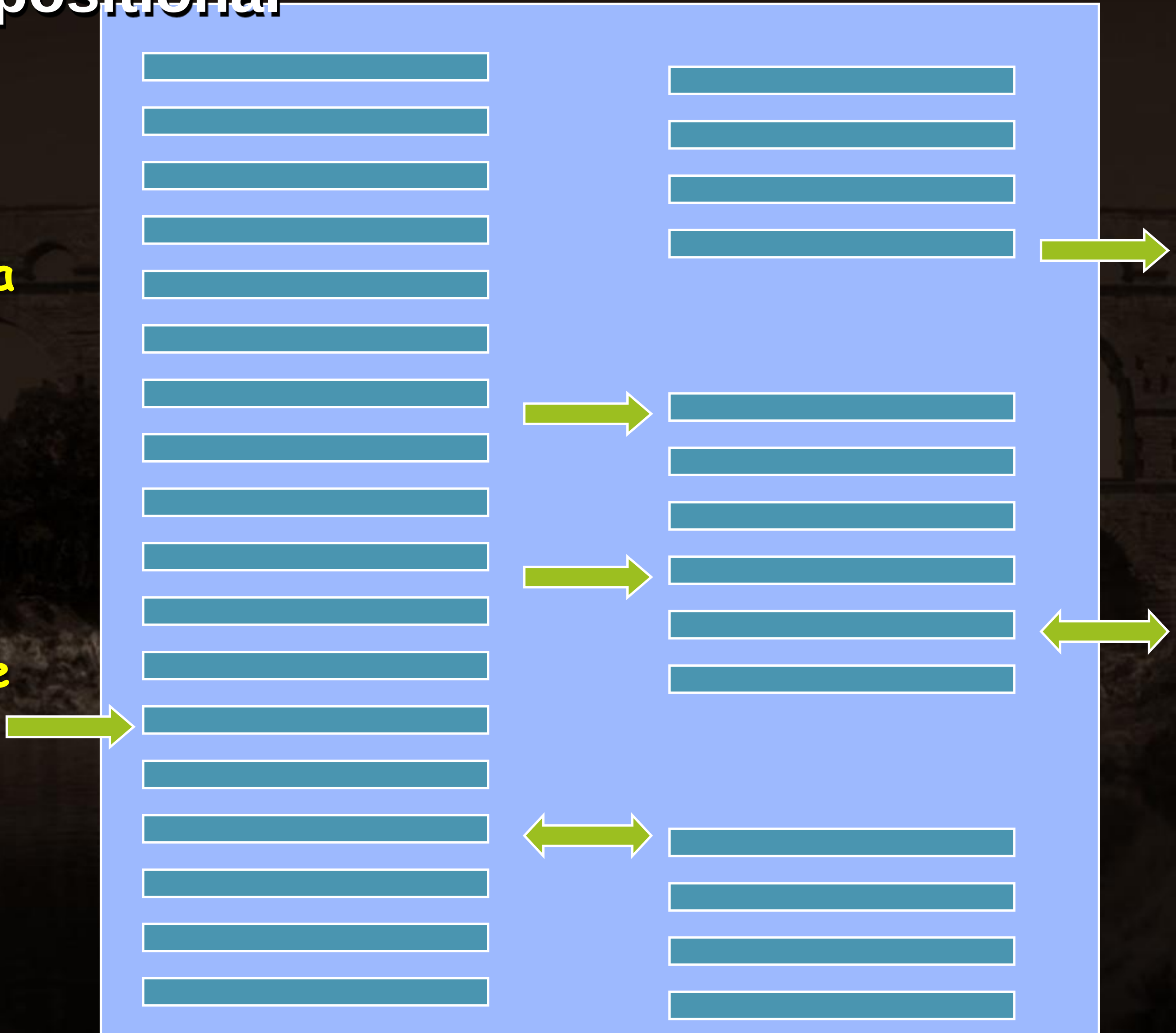
Software realizing these interactions is written at a very low level (e.g., semaphores). *Very hard to get it right.*

Source: Ed Lee

Interacting Processes – Not Compositional

An aggregation of processes is not a process (a total order of external interactions). What is it?

Many software failures are due to this ill-defined composition.



Source Ed Lee

Compositionality

Non-compositional formalisms lead to very awkward architectures.

What About Real Time?

"Make it faster!"

First Challenge on the Cyber Side:
Real-Time and Power-aware Software

Correct execution of a program in C, C#, Java, Haskell, etc. has nothing to do with how long it takes to do anything. All our computation and networking abstractions are built on this premise.

Timing of programs is not repeatable, except at very coarse granularity.

Programmers have to step outside the programming abstractions to specify timing and power behavior.

Second Challenge on the Cyber Side: Concurrency

Threads dominate concurrent software.

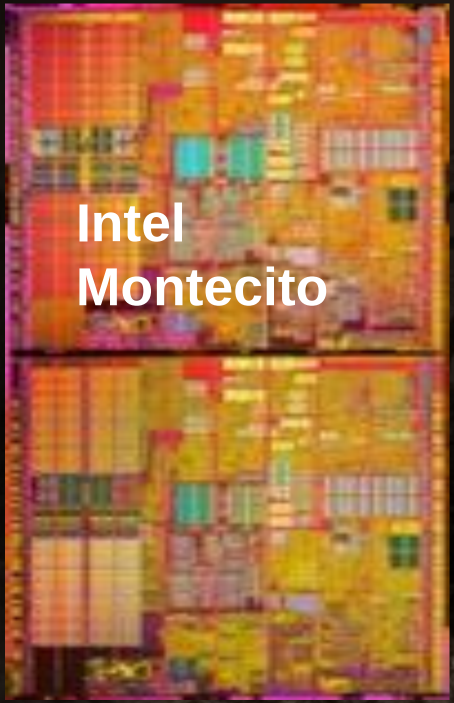
- *Threads*: Sequential computation with shared memory.
- *Interrupts*: Threads started by the hardware.

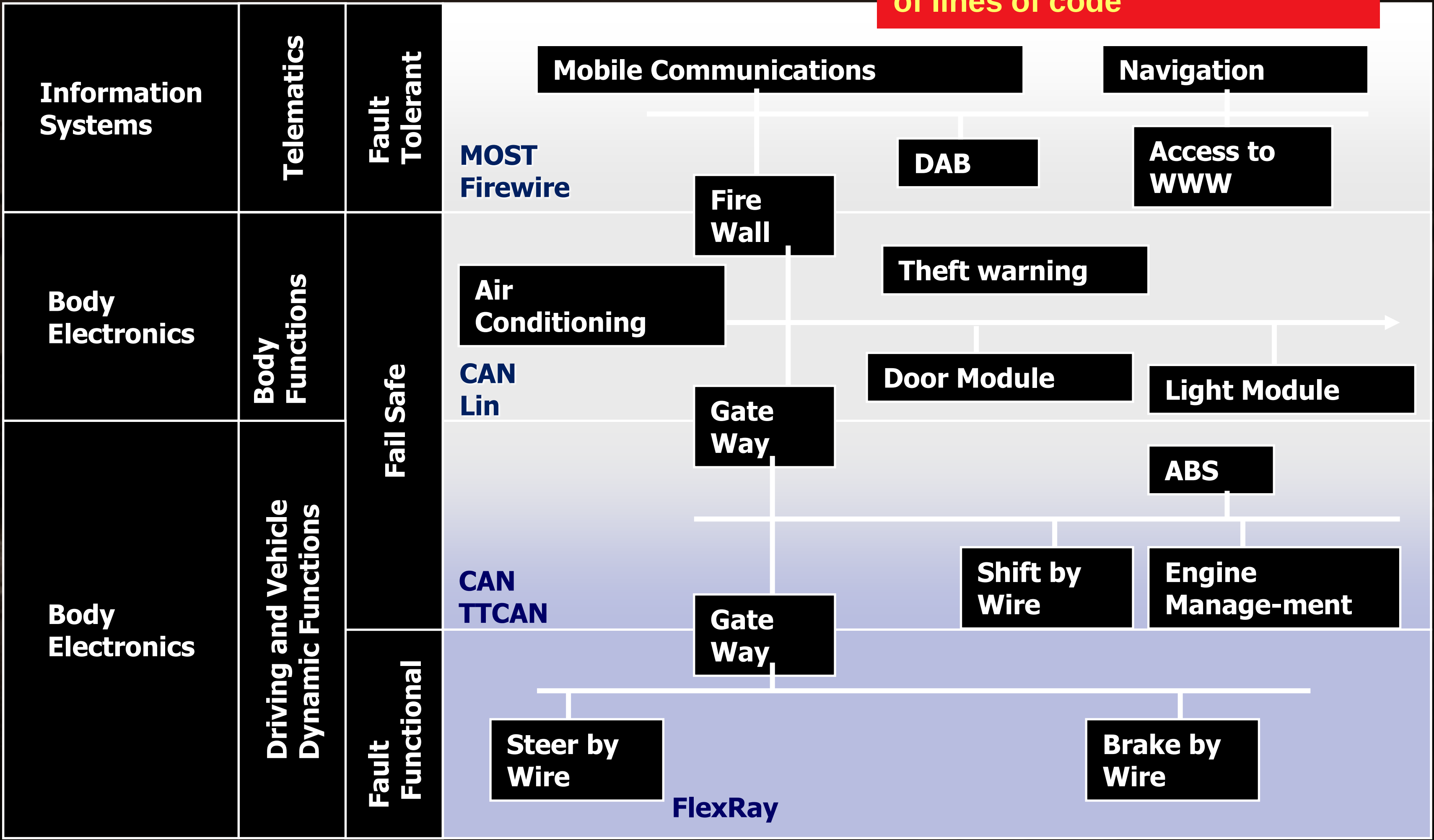
Incomprehensible interactions between threads are the sources of many problems:

- Deadlock
- Priority inversion
- Scheduling anomalies
- Nondeterminism
- Buffer overruns
- System crashes

Concurrency and Heterogeneity

Today, more than 80 Microprocessors and millions of lines of code





Source: Bosch

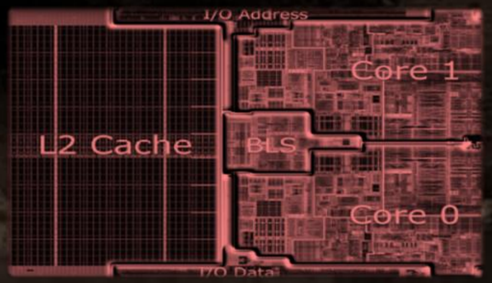
Challenge: Power

Energy = upper bound on the amount of available computation

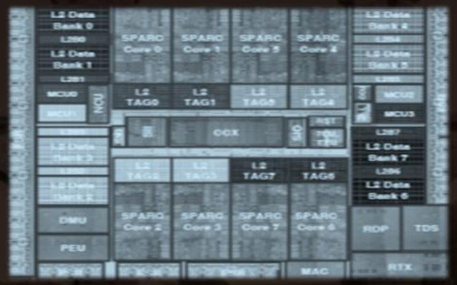
- Total Energy of Milky Way Galaxy: 10^{59} J
- Minimum switching energy for digital gate (1 electron@100 mV): 1.6 10^{-20} J (limited by thermal noise)
- Upper bound on number of digital operations: 6 10^{78}
- Operations/year performed by 1 billion 100 MOPS computers: 3 10^{24}
- Energy consumed in 180 years assuming a doubling of computational requirements every year.

Challenge: Parallel Architectures

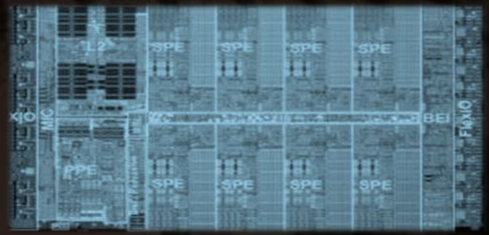
Scaling enabled integration of complex systems with hundreds of millions of devices on a single die



Intel KEROM dual core
ISSCC 07, 290M trans.



SUN Niagara-2
ISSCC 07, 500M trans.



IBM/Sony Cell
ISSCC 05, 235M trans.

Challenge: Manage the Design and Supply Chain

Semiconductor insights

Collaborating to Create the iPhone

SST
SST25VF080B
1 MB Serial Flash

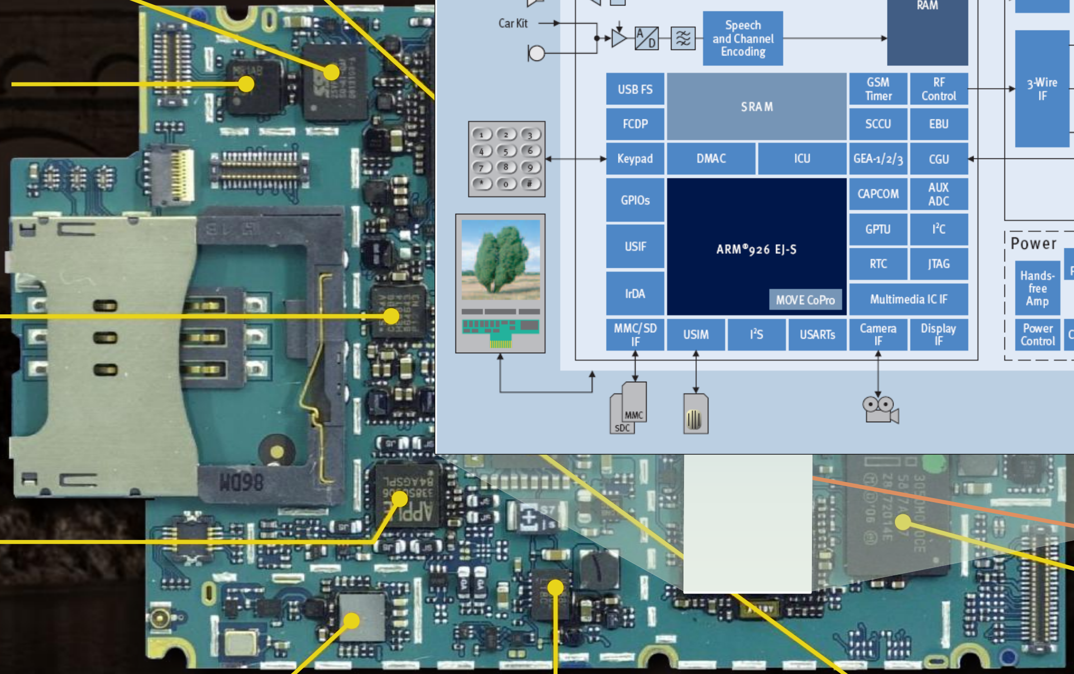
SAMSUNG
Application Processor and DDR SDRAM

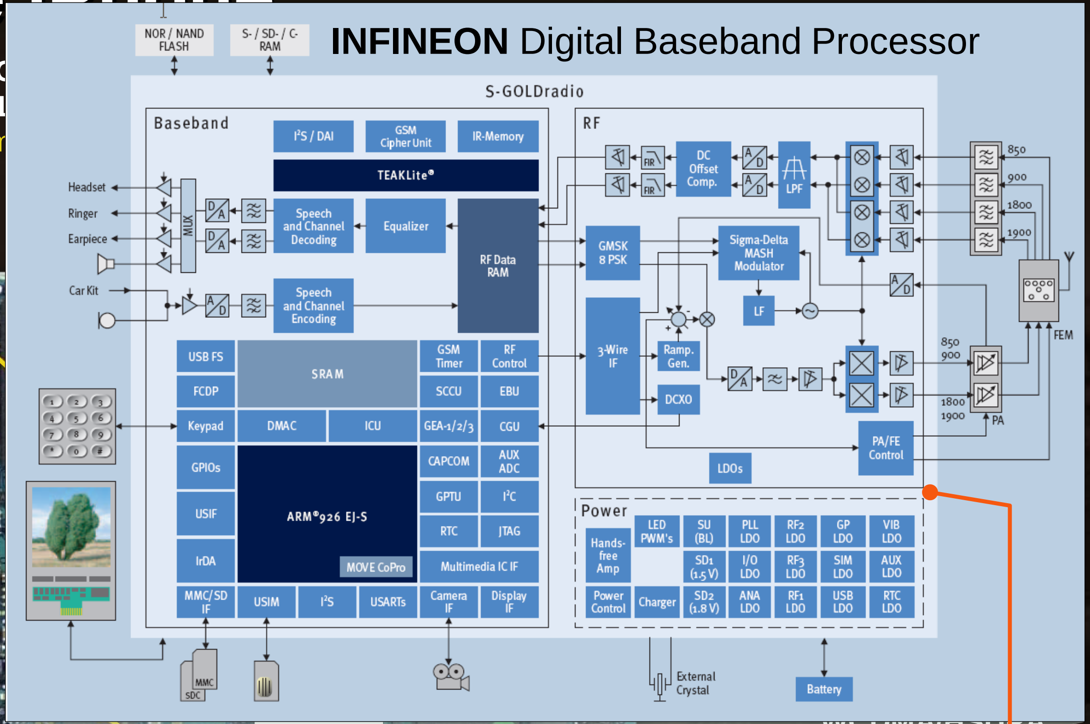
ST MICRO
LIS331 DL
Accelerom

NATIONAL SEMICONDUCTOR LM2512AA
Display Interface

BROADCOM BCM5974
Touchscreen Controller

WOLFSON WM6180C
Audio Codec





WCDMA/HSUPA
Power Amp.

INFINEON
Digital Baseband Processor

INFINEON PMB2525
Hammerhead II GPS

LINEAR TECHNOLOGY LTC4088-2
Battery Charger/ USB Controller

NXP
Power Management

NUMONYX PF38F3050M0Y0CE
16 MB NOR + 8 MB Pseudo - SRAM

Semiconductor insights

General Principles

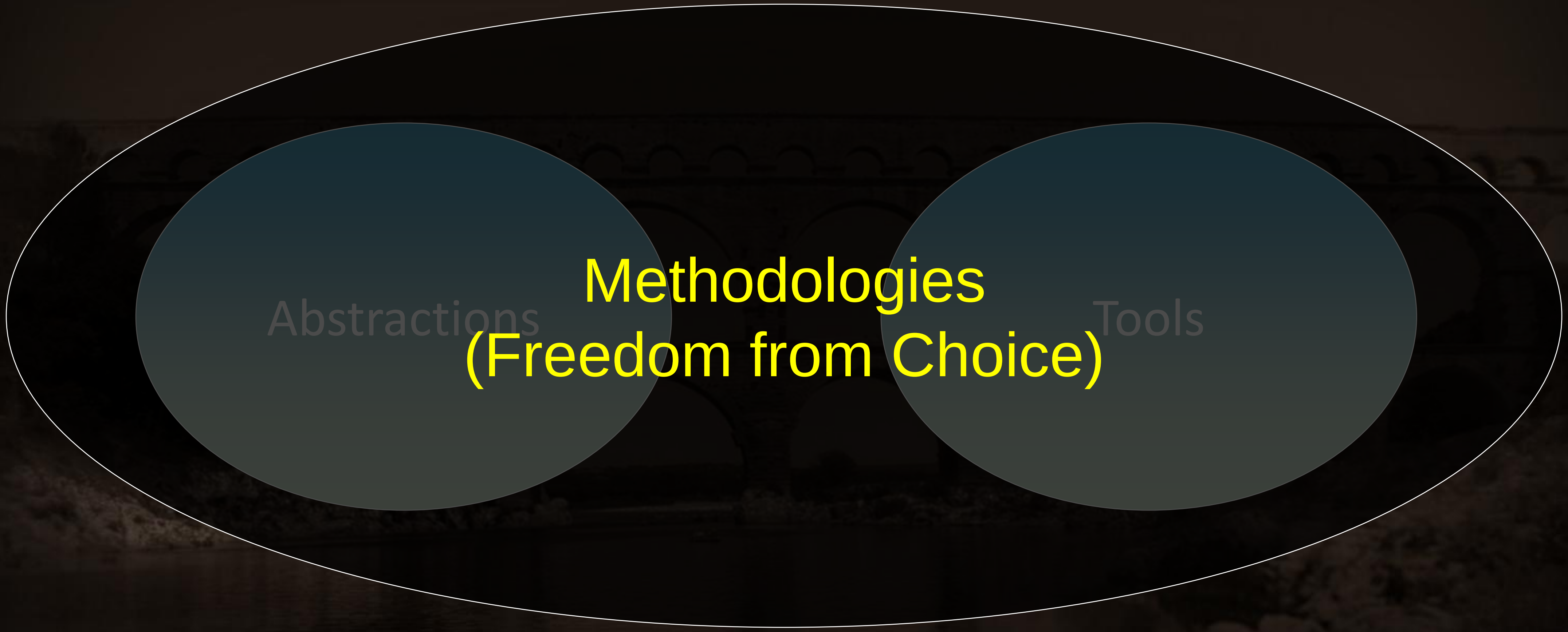
Traditionally complexity has been managed by two basic approaches:

- **Decomposition: reduce the number of items to consider by breaking the design object into semi-independent parts (*divide et impera)***
- **Abstraction: reduce the number of items by aggregating objects and by eliminating unnecessary details with respect to the goal at hand**

Complexity is also managed by "construction"

- **Constrain "artificially" the space (regular layout, synchronous designs)**
- **Start high in the abstraction layers and define a number of refinement steps that go from the initial description to the final implementation**

How did we cope with Complexity in the VLSI Era?



Integration Challenges: Plug and Play?

Plug and Pray!

The Design Integration Nightmare

Specification:

Implementation:

P. Picasso,
Blue Period

P. Picasso
“Femme se coiffant”
1940

Conclusion

We need a design and integration platform

- **To deal with heterogeneity:**
  - **Where we can deal with Hardware and Software**
  - **Where we can mix digital and analog, cyber and physical**
  - **Where we can assemble internal and external IPs**
  - **Where we can work at different levels of abstraction**
- **To handle the design chain**
- **To support integration**
  - **Tool integration**
  - **IP integration**
  - **Team Integration**

Platform-Based Design with Contracts can be the foundation for this platform