Energy Aware Computing Research Group

Ambient Intelligent Systems

Diana Marculescu
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
dianam@ece.cmu.edu

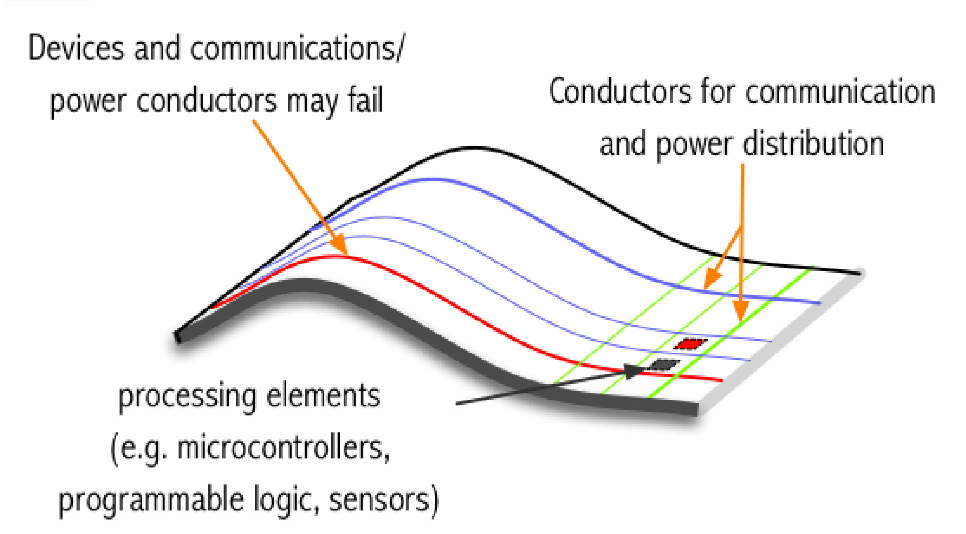
Electrical & Computer ENGINEERING

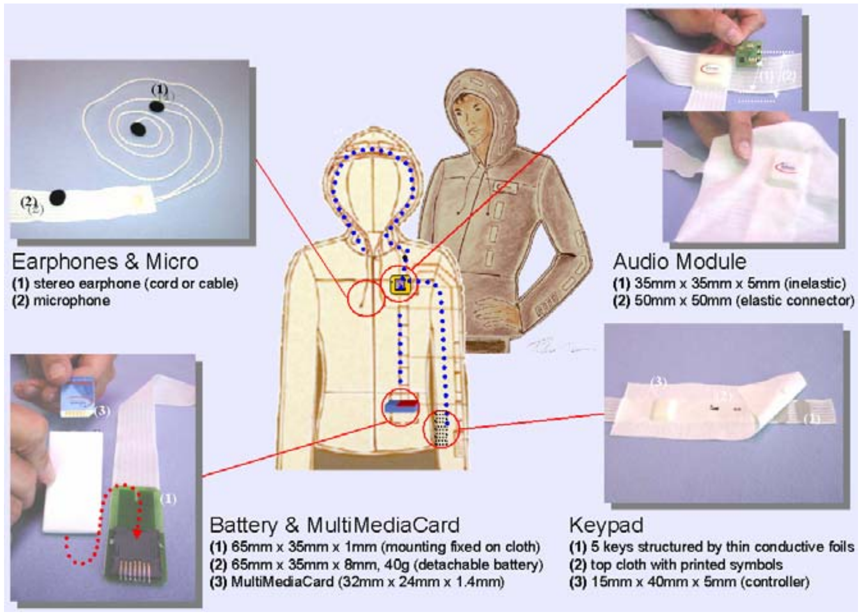
Ambient Intelligent Distributed Systems

- What is Ambient Intelligence?
  - Inconspicuous computing, sensing or actuation that *reacts* and *self-manages* in response to changes in environment or operating conditions.
- Characteristics
  - Distributed on large areas and subject to a variety of external conditions
  - Include a mix of wired and wireless communication
  - Limited in size, computing capabilities, power budget
  - Need to support self-monitoring and self-managing
- Ubiquitous computing taken a revolutionary step forward → "Living Designs"

Why Ambient Intelligent Systems?

- Taking *flexible* substrate computing one revolutionary step forward


Devices and communications/ power conductors may fail
Conductors for communication and power distribution
processing elements (e.g. microcontrollers, programmable logic, sensors)

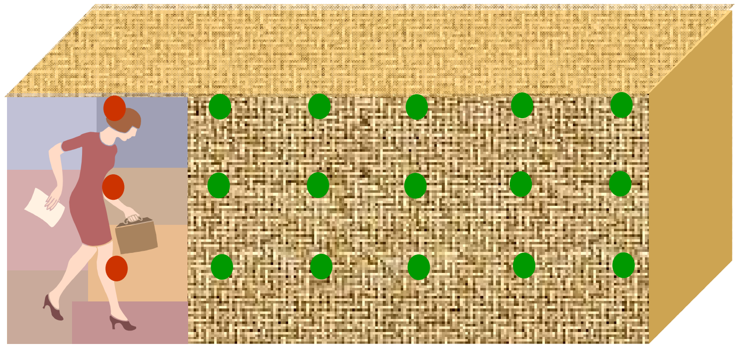

(1)
(2)
Earphones & Micro
(1) stereo earphone (cord or cable)
(2) microphone
Audio Module
(1) 35mm x 35mm x 5mm (inelastic)
(2) 50mm x 50mm (elastic connector)
(3)
(1)
Battery & MultiMediaCard
(1) 65mm x 35mm x 1mm (mounting fixed on cloth)
(2) 65mm x 35mm x 8mm, 40g (detachable battery)
(3) MultiMediaCard (32mm x 24mm x 1.4mm)
Keypad
(1) 5 keys structured by thin conductive foils
(2) top cloth with printed symbols
(3) 15mm x 40mm x 5mm (controller)

Source: Jung et al., 2002

- Target application domains
  - Integrated computing and actuation devices: 100’s of processing elements per m^2 in a flexible substrate

Bigger Picture
Ambient Intelligent (AmI) Distributed Systems

- Large area applications can benefit more from the sheer number of computing devices…

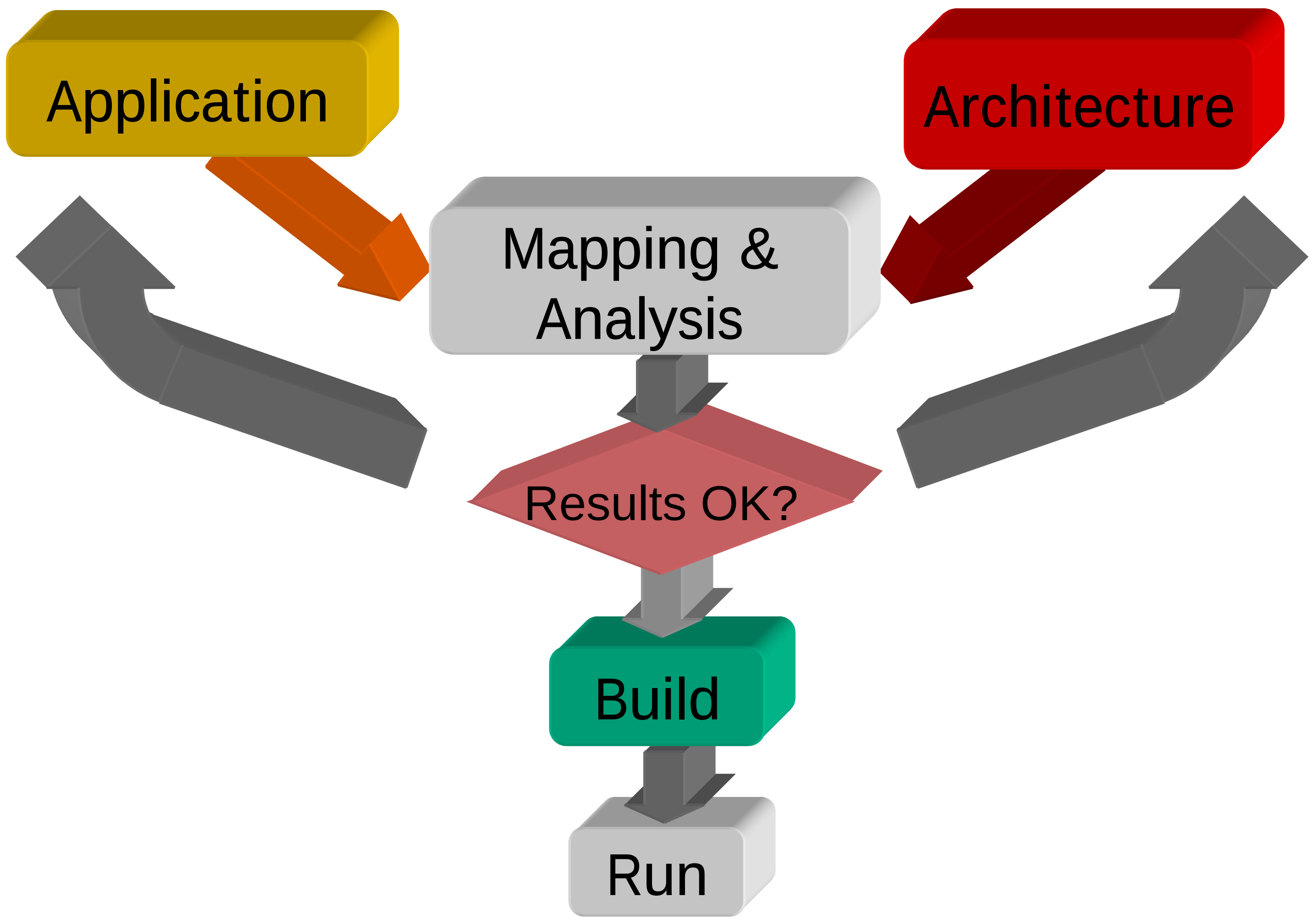


"Smart" rooms/hallways
"Smart" rugs
"Smart" environments

Pom-Pom dimmer
Source: Intl. Fashion Machines

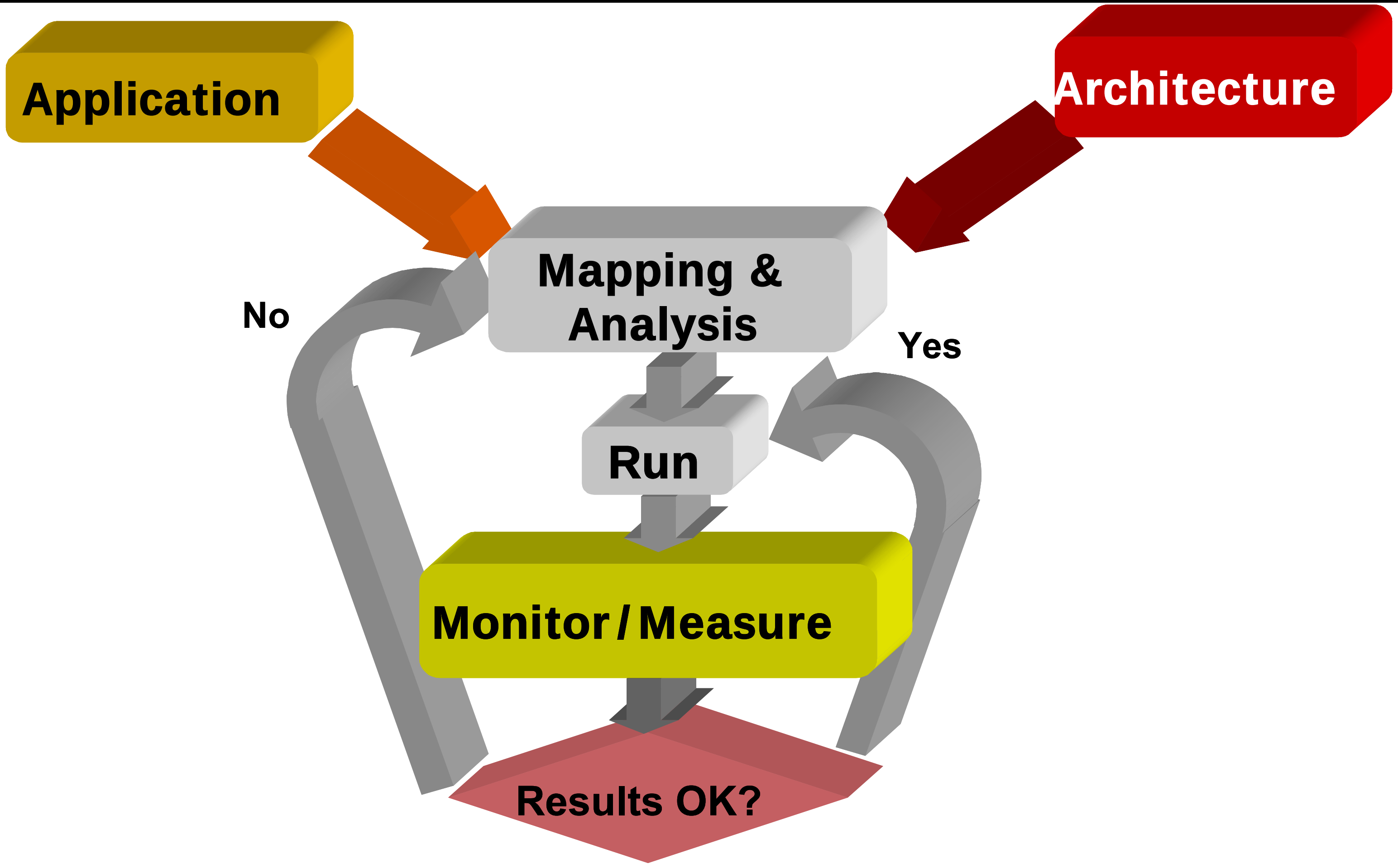
- New paradigm: "Computing-by-the-foot"
  - Textile Area Networks (TANs) that could be configured and re-programmed on-the-fly
- All within "smart" AmI environments

Design Methodology - The Old and the New



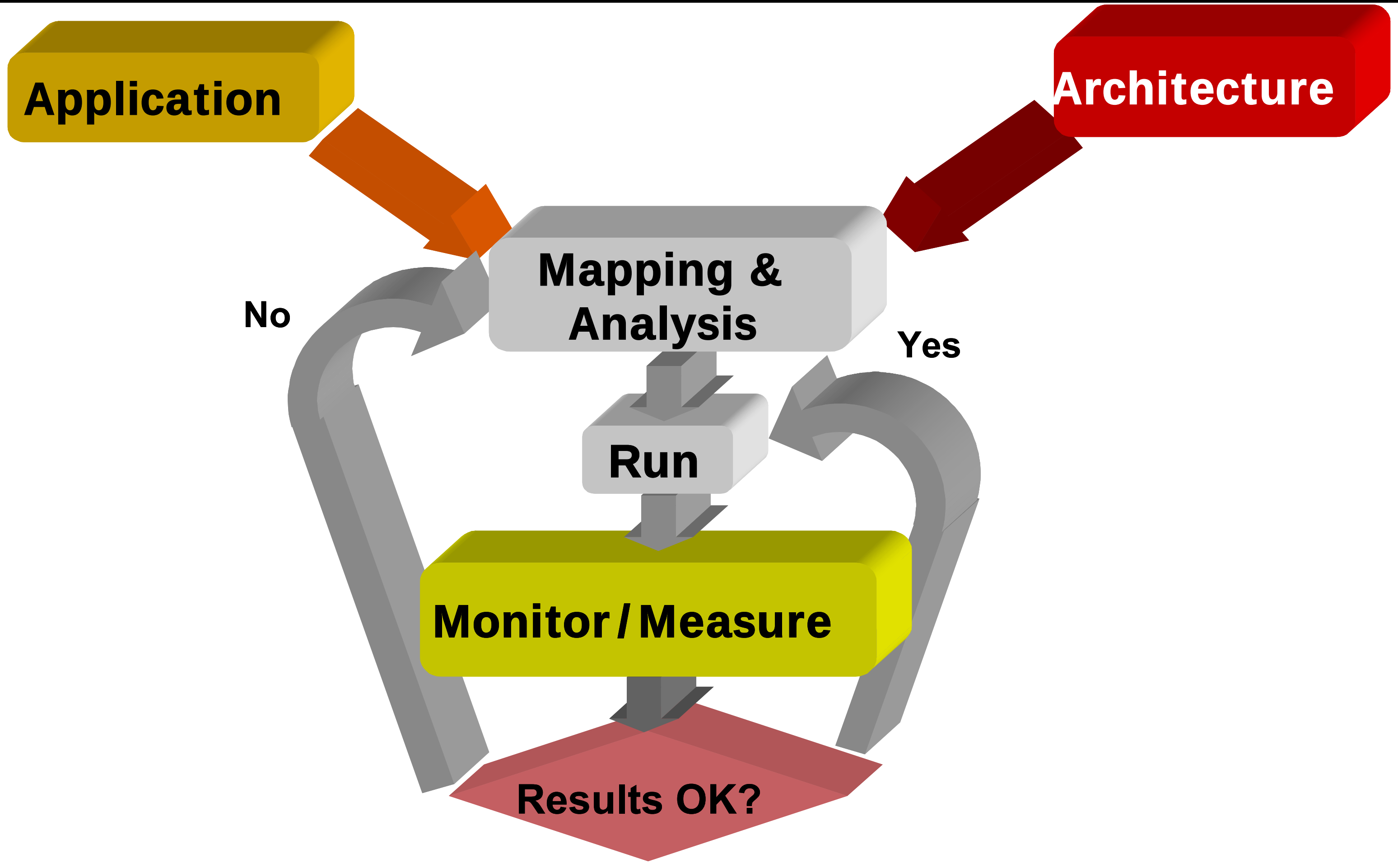
A typical design flow

Design Methodology - The Old and the New



- Move testing/verification to run-time
- Reduce manufacturing costs
- Cope with manufacturing or environment driven failures in a unified way

Design Methodology - The Old and the New



Hence the name Living Designs…

Wish List - What do we need to make this happen?

Technology

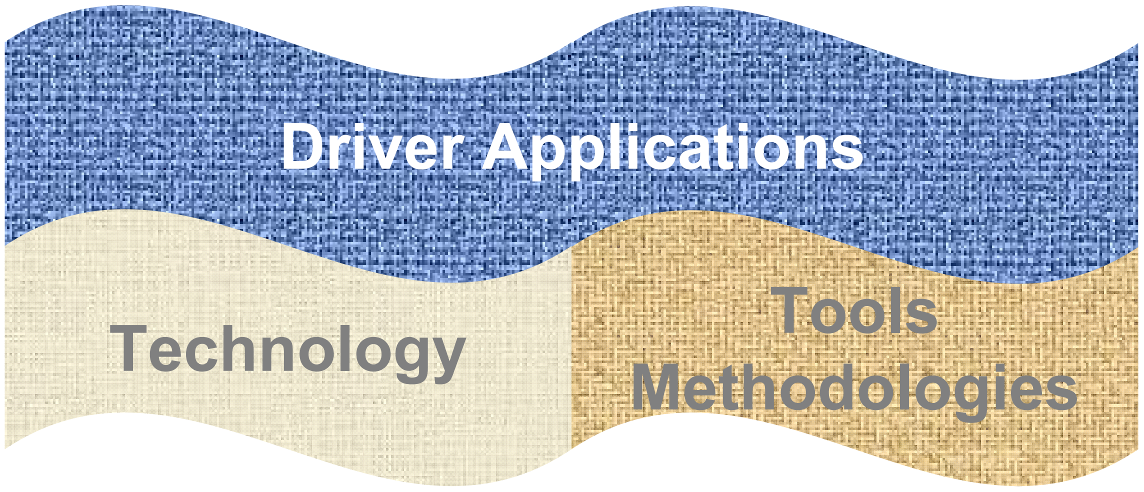
- Technology support
  - On-the-fabric component "plug-and-play"
  - Rigid PCBs -→ Flexible/foldable or fabric based PCBs
  - Weaving ribbons that "compute"
- Battery technology support
  - Energy scavenging (solar, vibration, movement)
  - Tubular or filament-type batteries

Wish List - What do we need to make this happen?

Technology
Tools
Methodologies

- On-the-fly
  - Monitoring (e.g., battery levels, failure rates)
  - Analysis/verification (dynamic management)
  - Dynamic reconfiguration/reprogramming

Wish List - What do we need to make this happen?



- Most likely, there is no single killer application…
  - Safety/security (CMU)
  - Medical (GeorgiaTech)
  - Entertainment/consumer electronics (Infineon, IFMachines)
  - …

Challenges

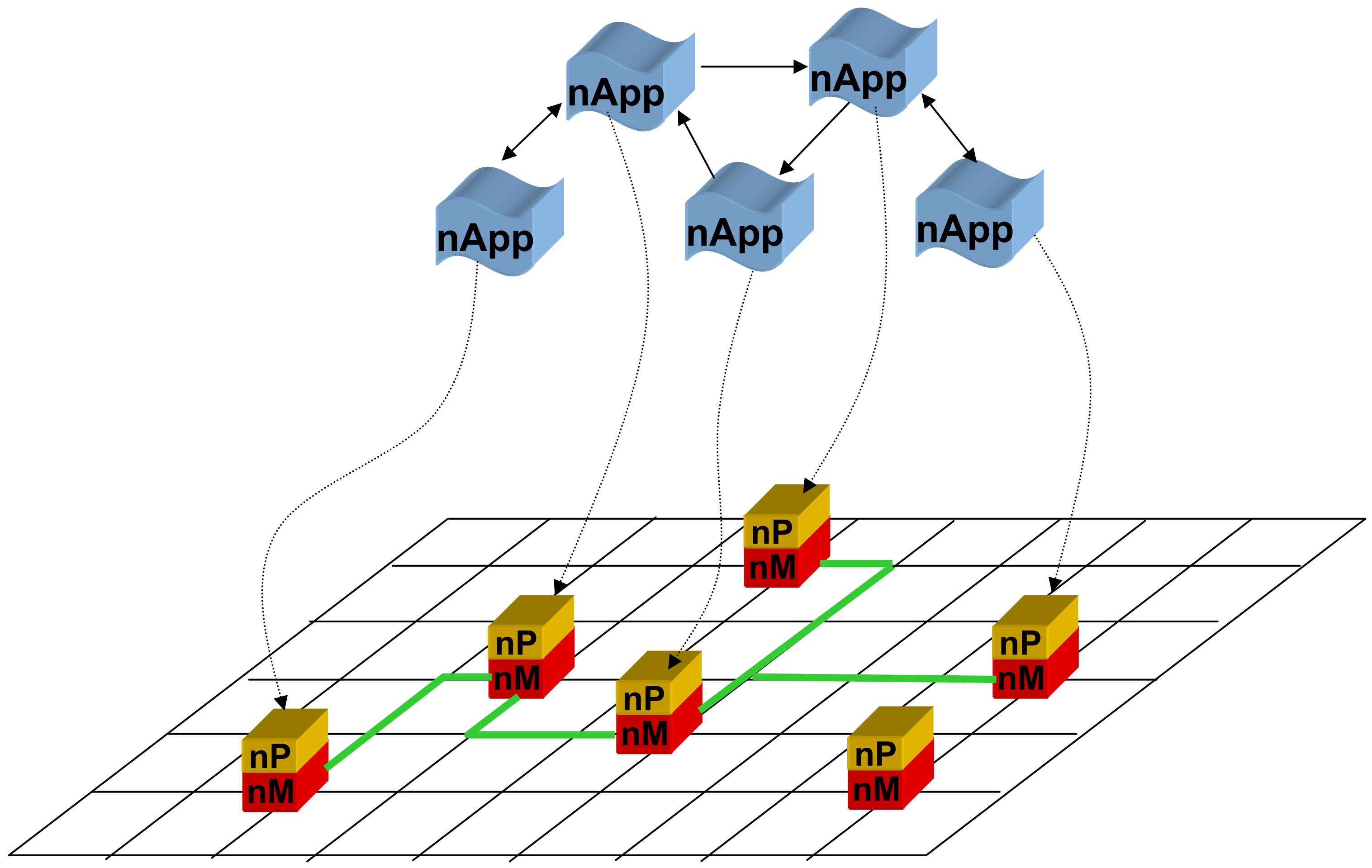
- Due to environmental or operating conditions, achieving reliable computation from large numbers of failure-prone, low-power devices

- Doing so with minimal overhead or cost, under finite battery lifetime constraints

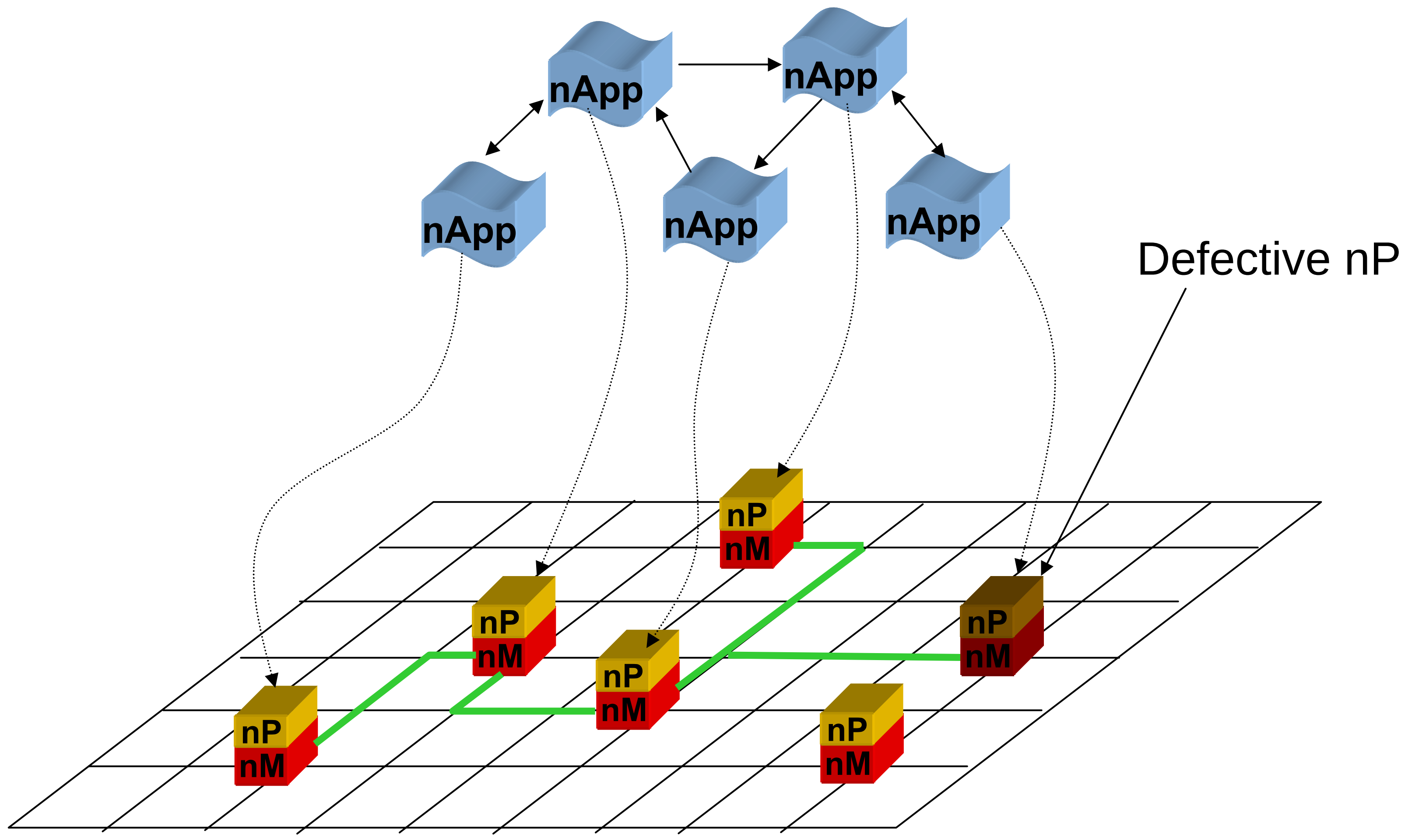
- In the presence of failures
  - Source of failures must be correctly *detected*
  - Fault-tolerance is essential for preserving quality of results or increasing system lifetime
  - Perform monitoring/adaptation without central control!

- Define metrics for characterizing a combination of performance, energy-efficiency, reliability and battery life

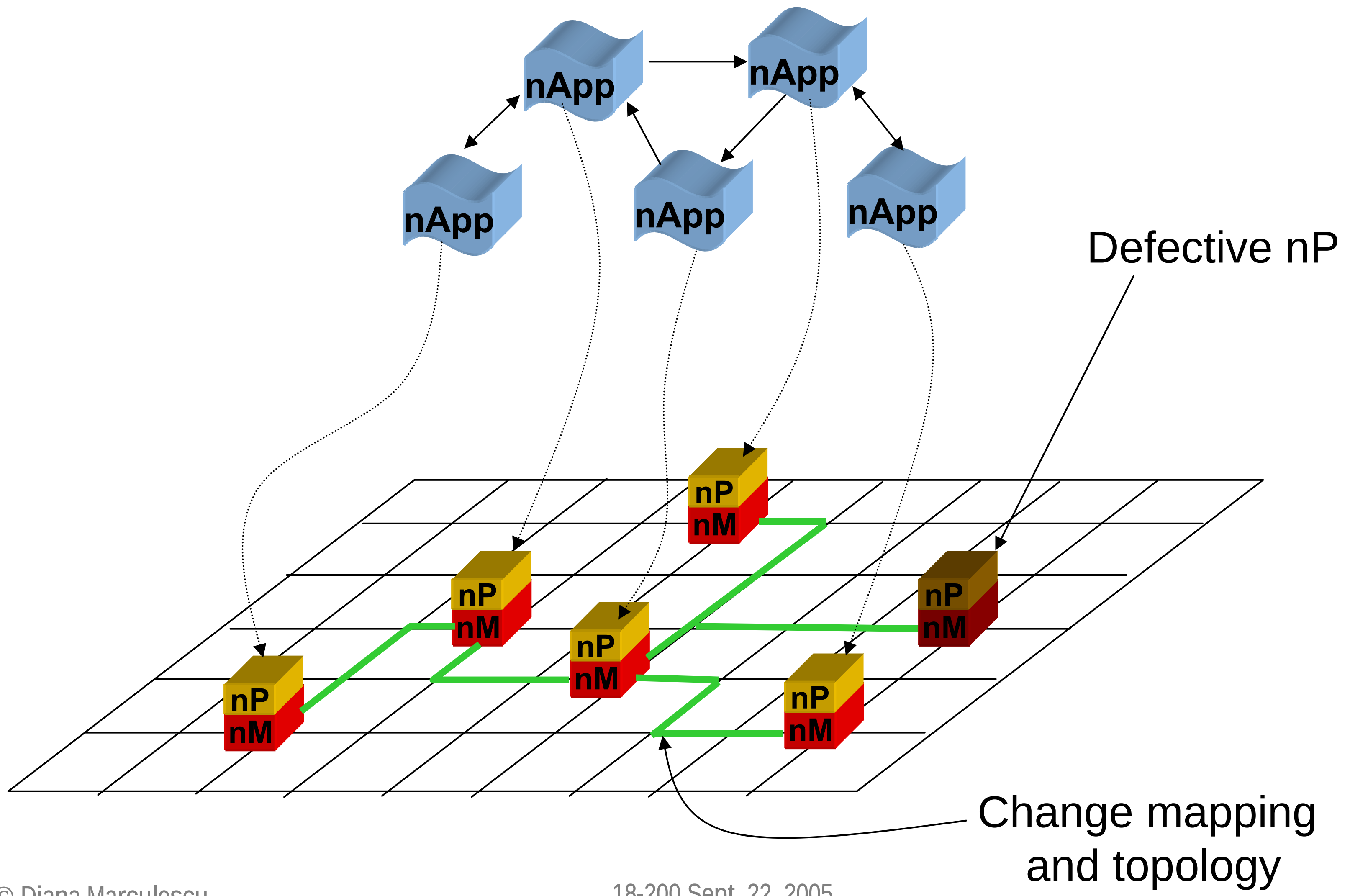
Dealing with Failures: Adaptive Mapping



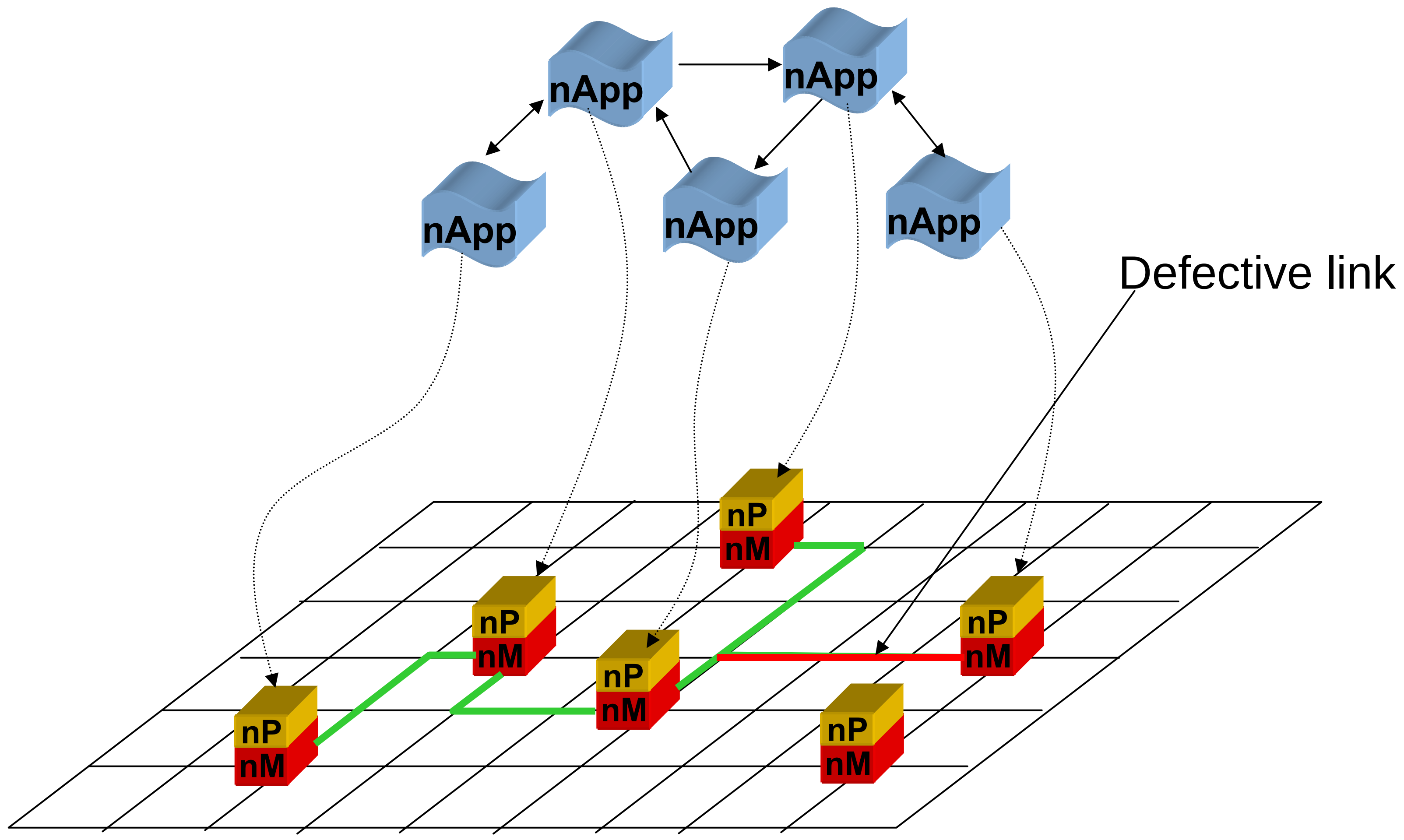
Dealing with Failures: Adaptive Mapping



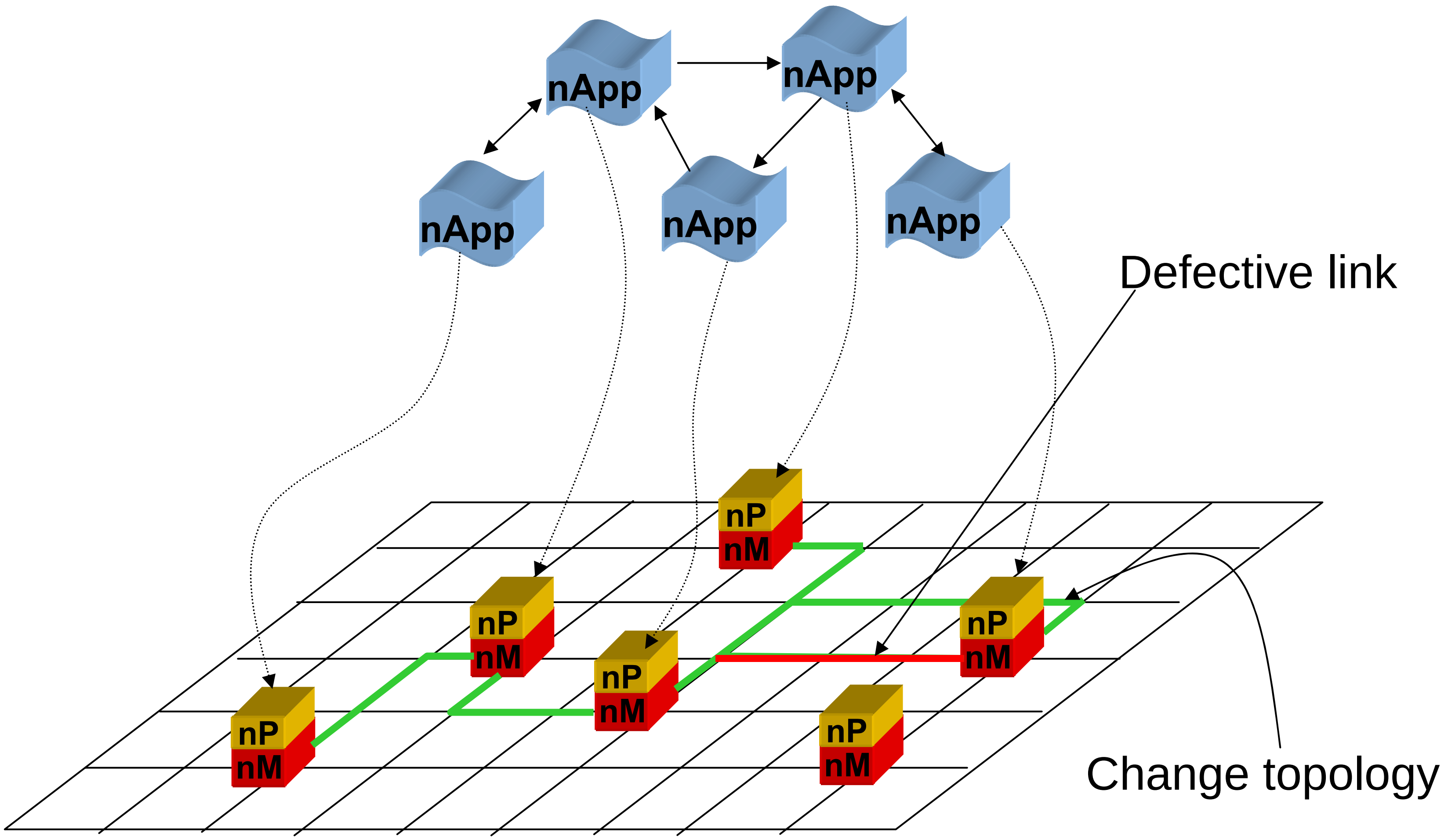
Dealing with Failures: Adaptive Mapping



Dealing with Failures: Adaptive Routing



Dealing with Failures: Adaptive Routing



Exploiting Redundancy

- Devices are energy constrained, failure prone, but…
  - Large numbers of them, networked
  - Idle devices can act as surrogates for failing ones
- Application Remapping
  - Take advantage of redundancy to counteract failures
  - Move executing applications across devices

Slide 17

j10 Although failures in e-textiles will be common, there will also be a very dense node existence on a textile, up to on the order of 100 nodes / square meter. Therefore, there is the opportunity to exploit this redundancy in the nodes available on the textile to provide fault tolerance for the system and applications running on the textile.

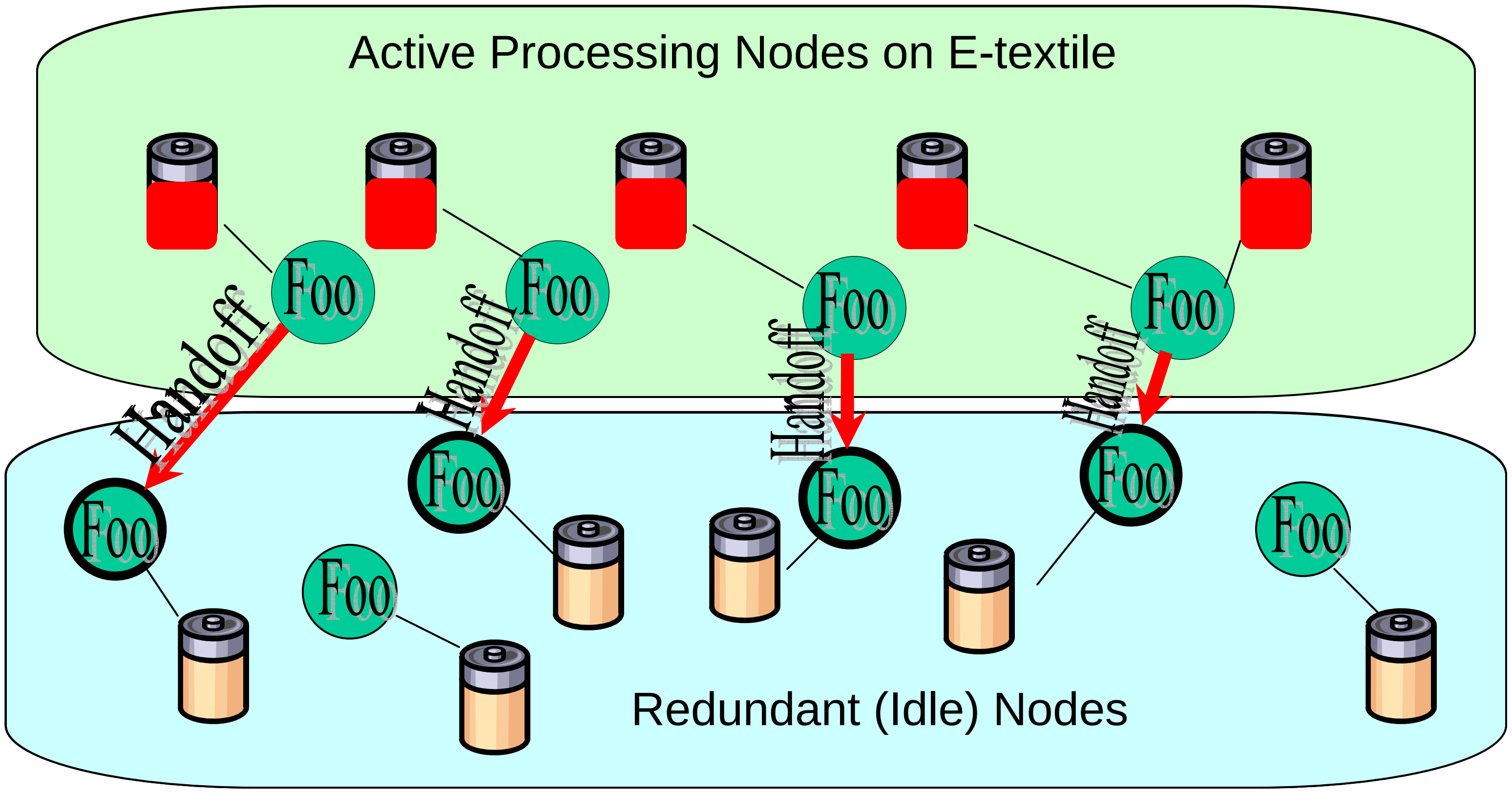
As I said before, battery energy is very limited on an e-textile, so if an application needs to execute despite a depleted battery, then that application needs to be remapped onto redundant nodes at that time.

Remote execution and code migration are two standard techniques to performing application code remapping. We present and propose a new, novel technique for code remapping that we call pre-copying with remote execution, which is a hybrid scheme of code migration and remote execution.

jingcao, 10/5/2002

Remote execution

- Hand-off current state to redundant node when battery is below threshold, B_{low} (code is already residing on the spare device)



Slide 18

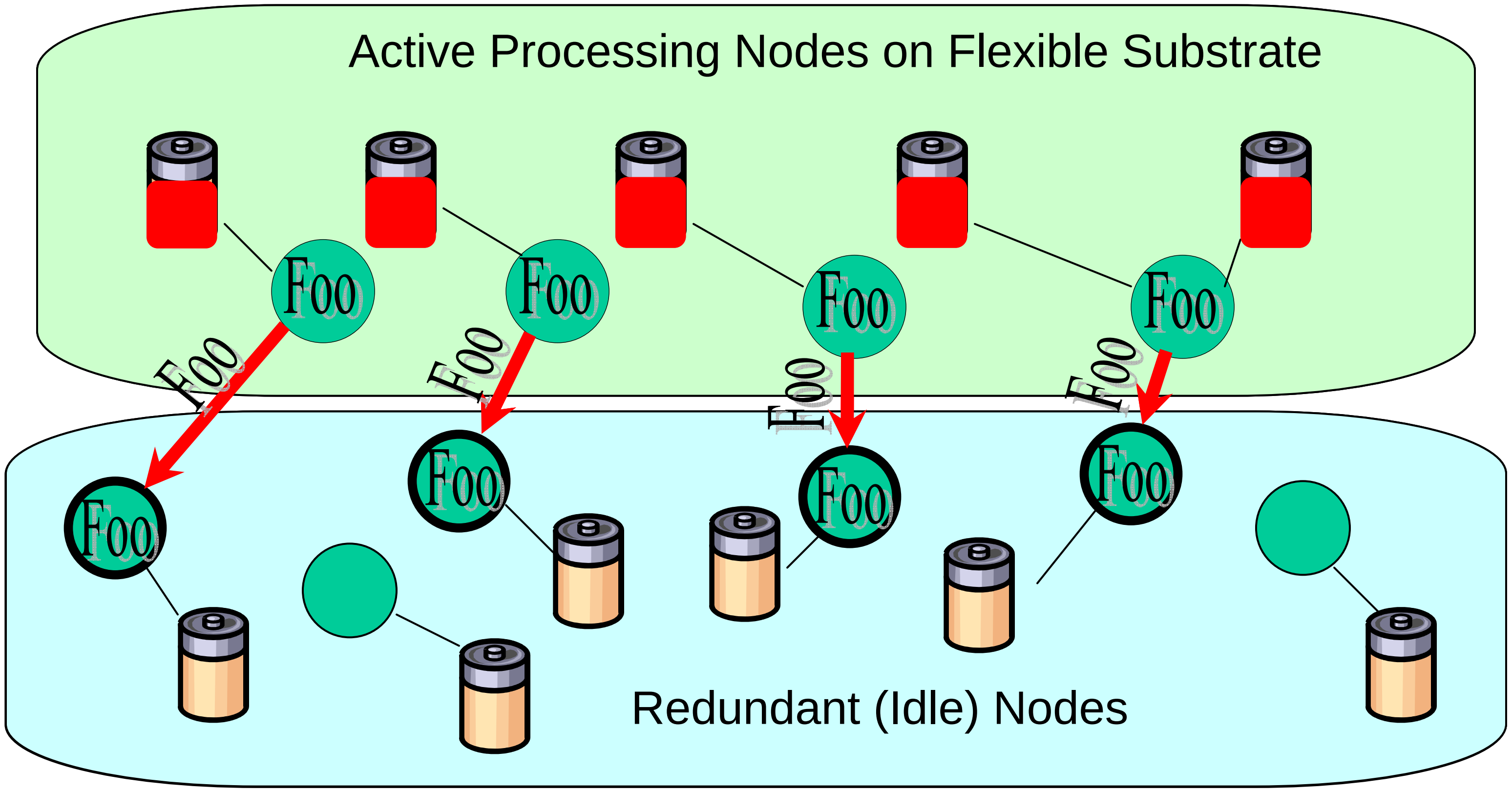
j11 This animation should clarify the process of remote execution. Notice that before execution begins of the application Foo, each node has that application copied into its memory.

When a battery falls down below a critical level, each active node makes a handoff to a redundant node. This handoff includes transmitting the entire current state of the running application. Once the handoff is complete, the application can resume running on the redundant nodes.

jingcao, 10/4/2002

Baseline code migration

- Copy application code and current state to redundant node when battery is below threshold, B_{low}



j12 The other form of code remapping involves code migration, or code copying. With this technique, each active node copies the current state of the application in addition to the executing application code during the remapping process.

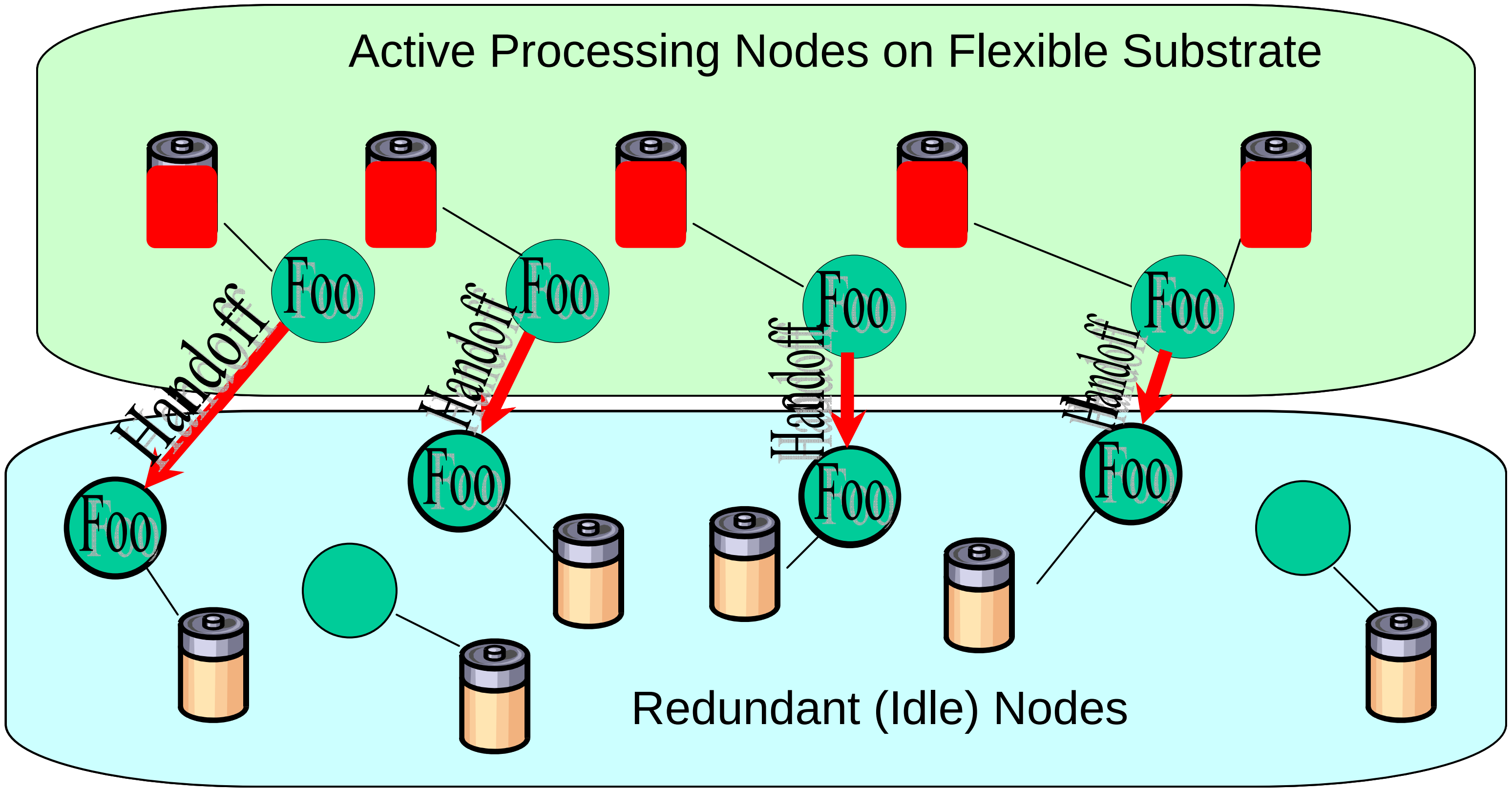
Notice that, as opposed to remote execution before, here only those nodes which begin active have stored in its memory the application which will be run on the e-textile. The baseline code migration scheme we considered simply performs this code remapping when the batteries fall to a certain level.

And because the size of the application might be somewhat large, the battery level at which you should choose to begin the re-mapping needs to be conservative enough to ensure that you successfully migrate before running out of energy completely.

jingcao, 10/5/2002

Pre-Copying with Remote Execution (PCRE)

- Copy code and current state to redundant node in non-overlapping turns, and complete handoff when battery is below threshold, B_{vlow}



Slide 20

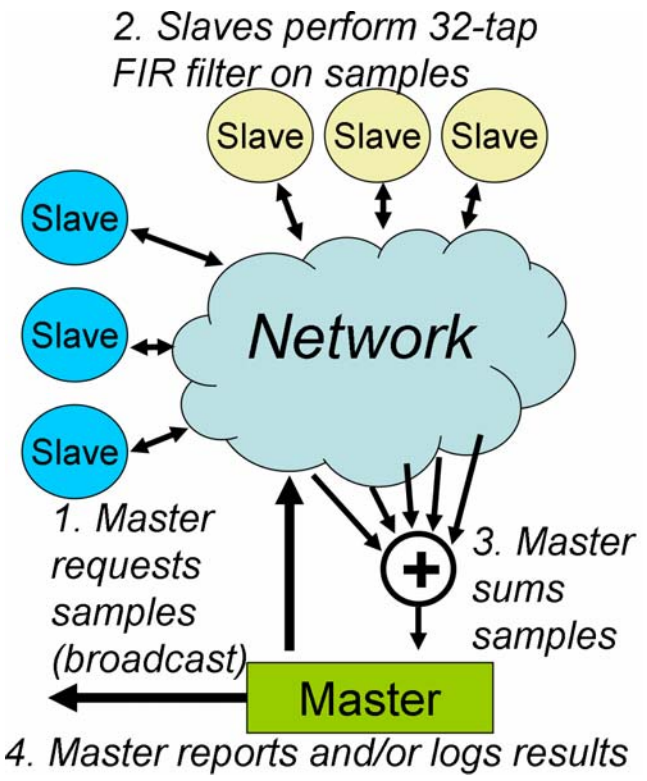
j13 Here is the same diagram as before, now showing the behavior of PCRE.

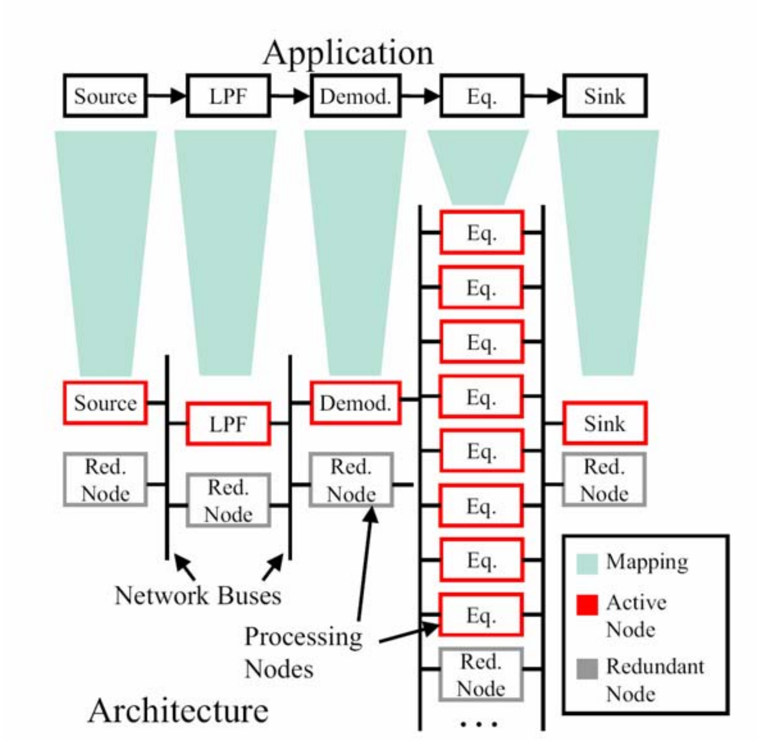
Similar to code migration, only the active nodes are aware of the application Foo before execution begins. In non-overlapping turns, each active node will copy its code to a redundant node elsewhere in the system. After each node has copied its code to a redundant node, they resume normal operation until the battery has fallen to a very low level, at which point they perform the same handoff that occurs with remote execution. After the handoff, the application can resume execution on the redundant nodes.

jingcao, 10/4/2002

Driver Applications

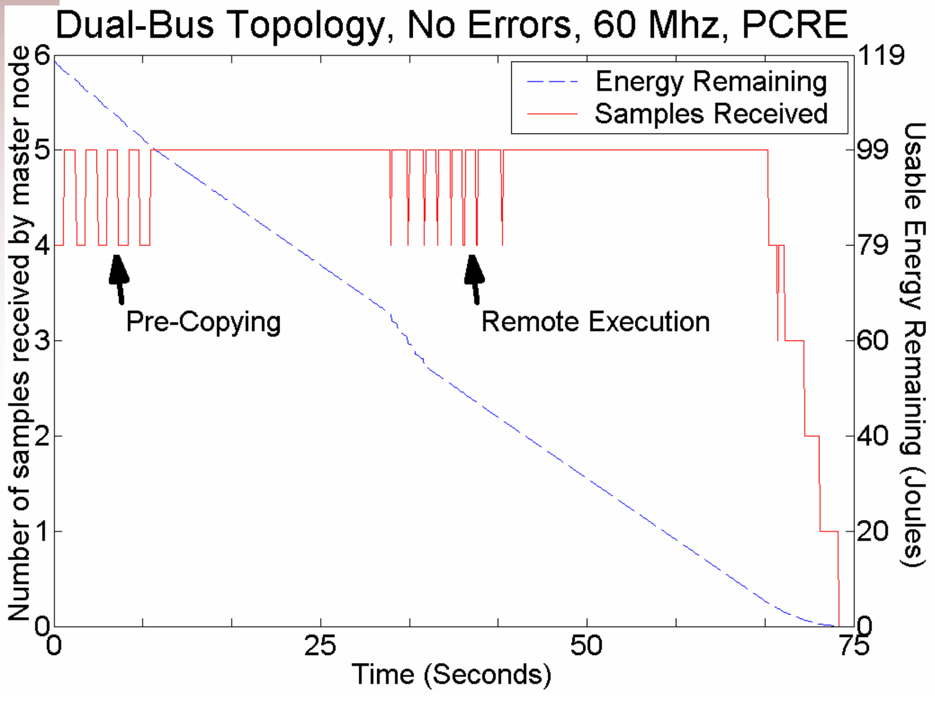
- Applications allowing for graceful performance degradation
  - Acoustic beamforming array
- Applications with single point failures
  - Software defined radio

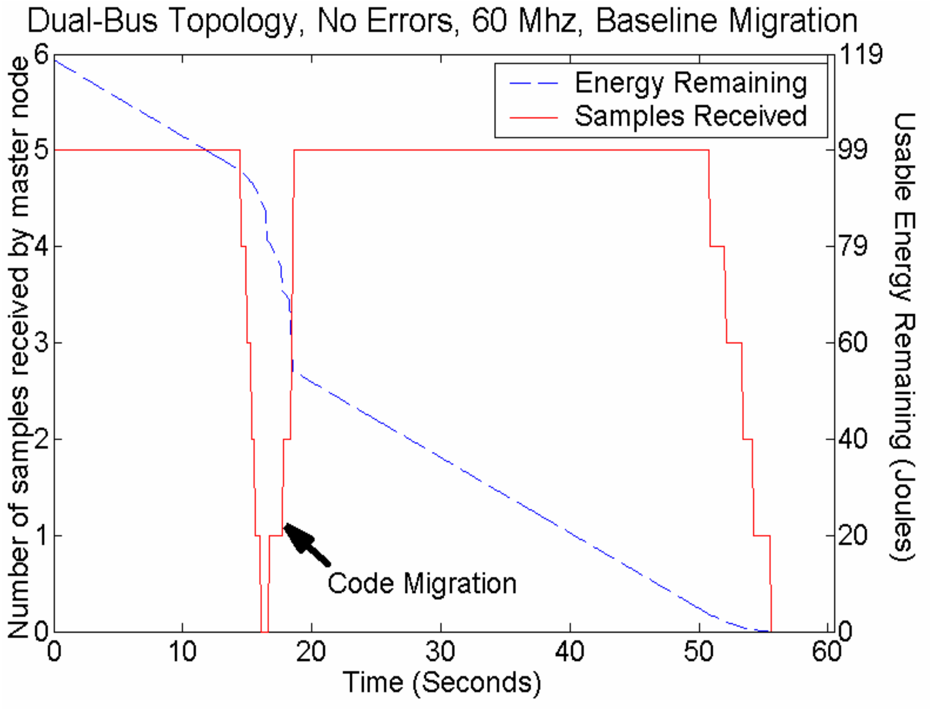

2. Slaves perform 32-tap FIR filter on samples
Slave
Slave
Slave
Slave
Slave
Slave
Network
1. Master requests samples (broadcast)
3. Master sums samples
Master
4. Master reports and/or logs results



Acoustic Beamforming

- Pre-copying permits staging of migration process





72.3 Sec / 115.26 J = 0.627 Seconds/Joule

54.4 Sec / 95.2 J = 0.571 Seconds/Joule

80% lifetime increase when using PCRE

Slide 22

j14

For experiment #1, no errors are present and a dual-bus topology is used. The left graph shows the simulation of the beamformer with PCRE, and the right graph shows the same setup using standard, or the baseline, code migration technique.

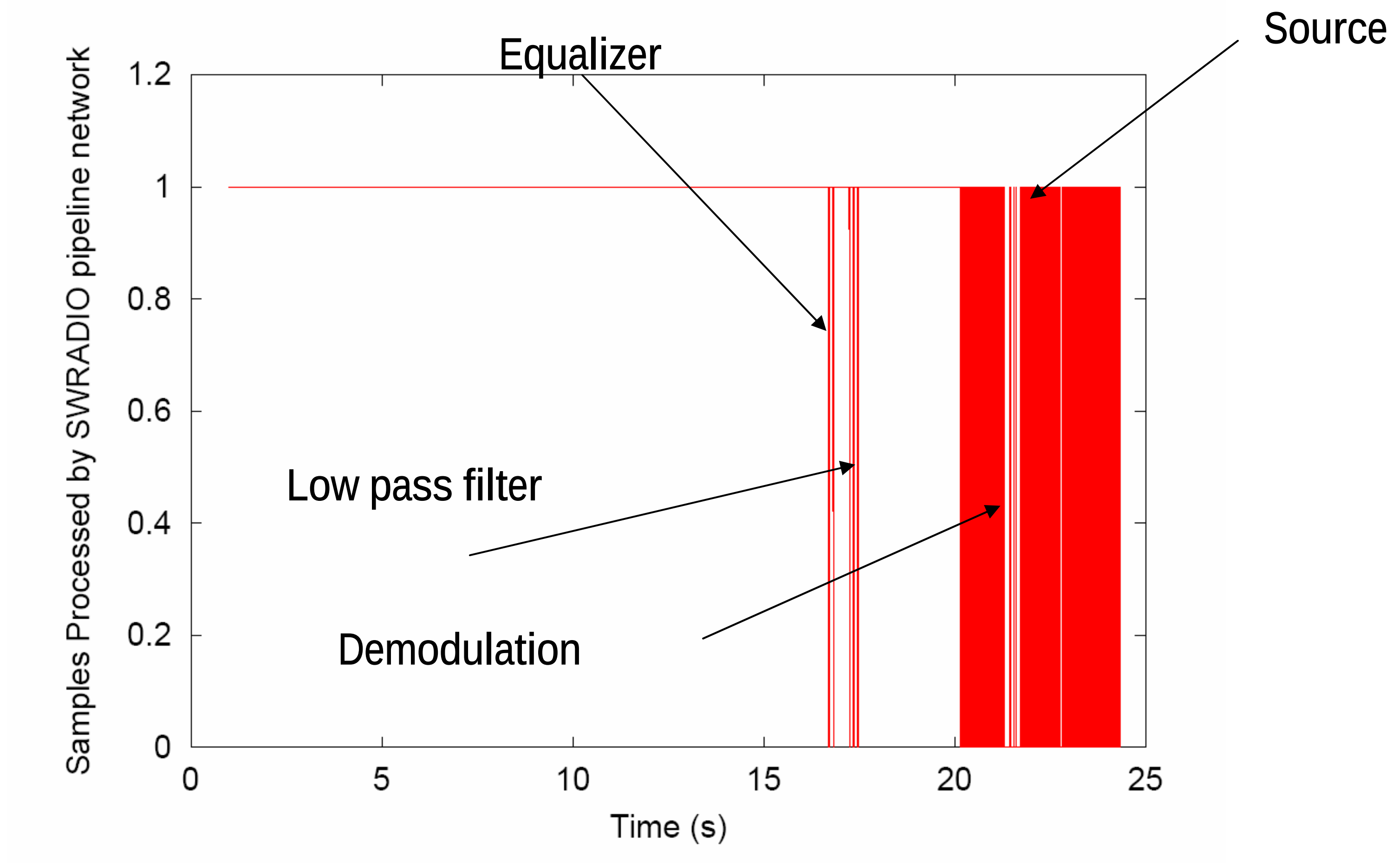
The solid lines show the number of samples the master node receives each beamforming round, and the dotted line shows the amount of battery energy available left for the beamforming application. Looking at the PCRE graph on the left, you can the sawtooth shape at the beginning of the simulation, and this represents each of the 5 slave nodes taking turns copying its code to a redundant node. After copying their code, they resume executing the beamforming application until their batteries reach a critical level, at which point they perform the final handoff and complete the remapping process. Notice that the dotted line decreases at about the same rate throughout this simulation.

Now, looking at the baseline migration technique, you can see the big dip that occurs when each node attempts to begin copying its code to a redundant node. They all successfully migrate to redundant nodes. Notice this time that the dotted line drops sharply at the point where the active nodes complete their migration. This is because those active nodes are essentially abandoning what battery resources they may have left by migrating so early. Like I said before, the point at which this migration occurs needs to be conservative enough to ensure that in the presence of errors with high probability they will successfully migrate.

You can see from the graphs that the PCRE beamforming lives almost 75 seconds, in fact lives 72.3 sec, while it consumes about 115.26 joules, and has an energy efficiency of 0.627 system lifetime per joule consumed. Baseline code migration suffers with only 54.4 Sec lifetime while consuming 95.2 Joules during this time, which is a less efficient 0.571 system seconds / joule consumed ratio. Also, there is that significant dip in performance which occurs when the nodes in the baseline migration case all try to migrate at similar times.

jingcao, 10/5/2002

Software Defined Radio

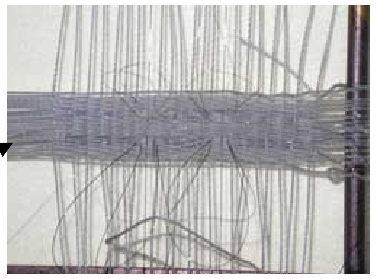


- Migration happens for the equalizer, low-pass filter, demodulation, and source nodes, in this order
- Effectively prolonging application lifetime (by about 30% when using code migration) is possible!

The ECE Connection

As you probably figured out…

- These are *deeply* networked embedded systems, capable of sensing/actuation
- However, they can't use Pentium4-like processors …
- … Nor they are stand-alone portable devices such as PDAs or cell phones
- They might interact with other "intelligent designs" – e.g., robots…
- … but they are mainly characterized by *large numbers* of simple embedded controllers interconnected in a wired or wireless manner…



These yarns can "compute"!

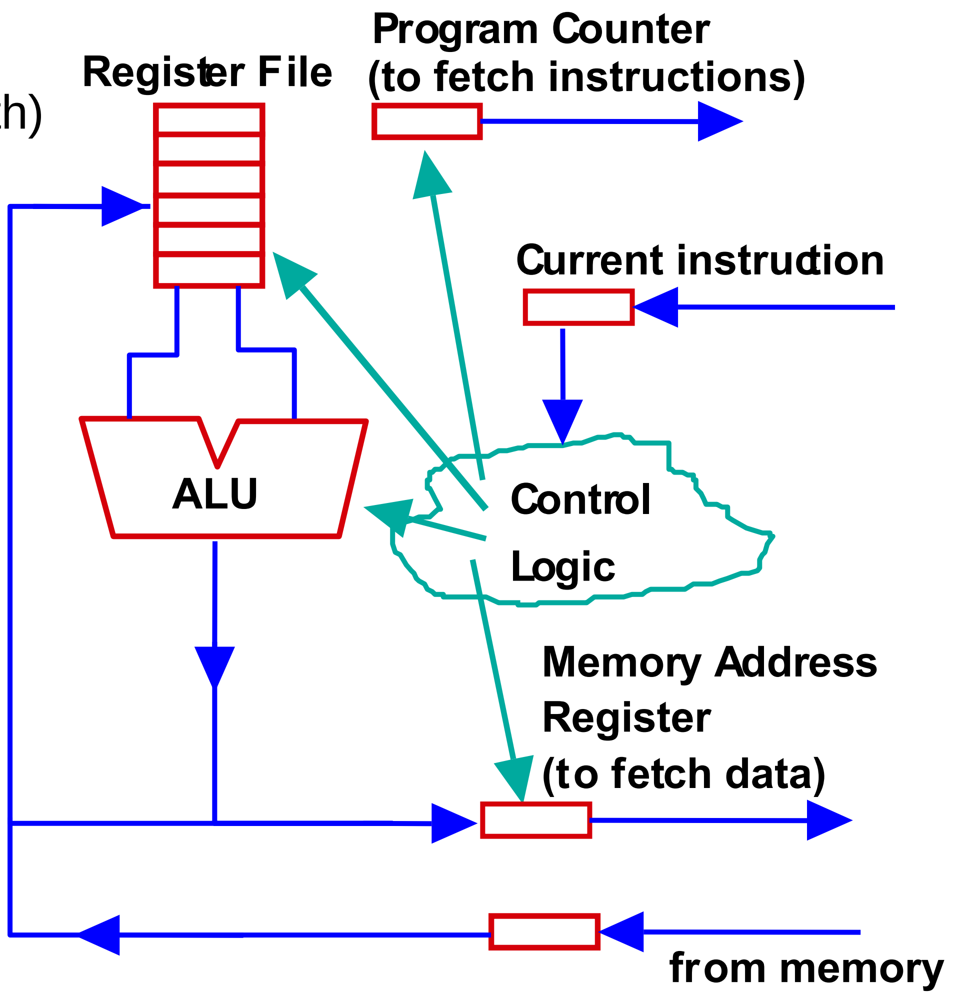
Source: V. Subramanian, UC Berkeley

What's behind such a system?

- Microcontrollers/microprocessors
  - How are they built – what is their internal architecture?
  - These are simple processors – and believe it or not, you need to know ***computer architecture*** to design these!
- Computer-Aided Design tools for
  - Designing the processing nodes (controller + memory + I/O, etc.)
  - Deciding on how the application is mapped
- You can find out about all these by taking ECE classes!

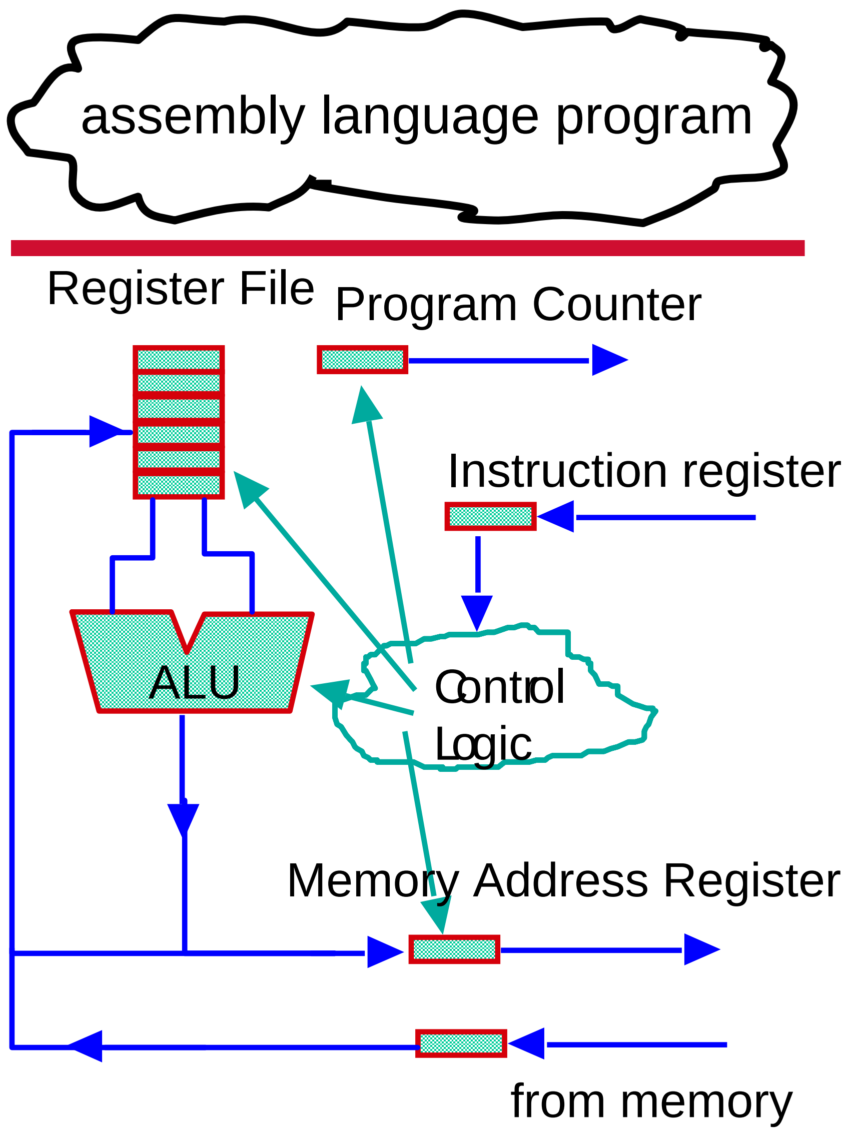
What's Inside a Processor?

- Partitioned into:
  - Control Logic (control path)
  - Datapath
- Datapath includes
  - Register File
  - Arithmetic-Logic Unit
  - Program Counter
  - Memory Register
- Control path uses instructions to manage the datapath

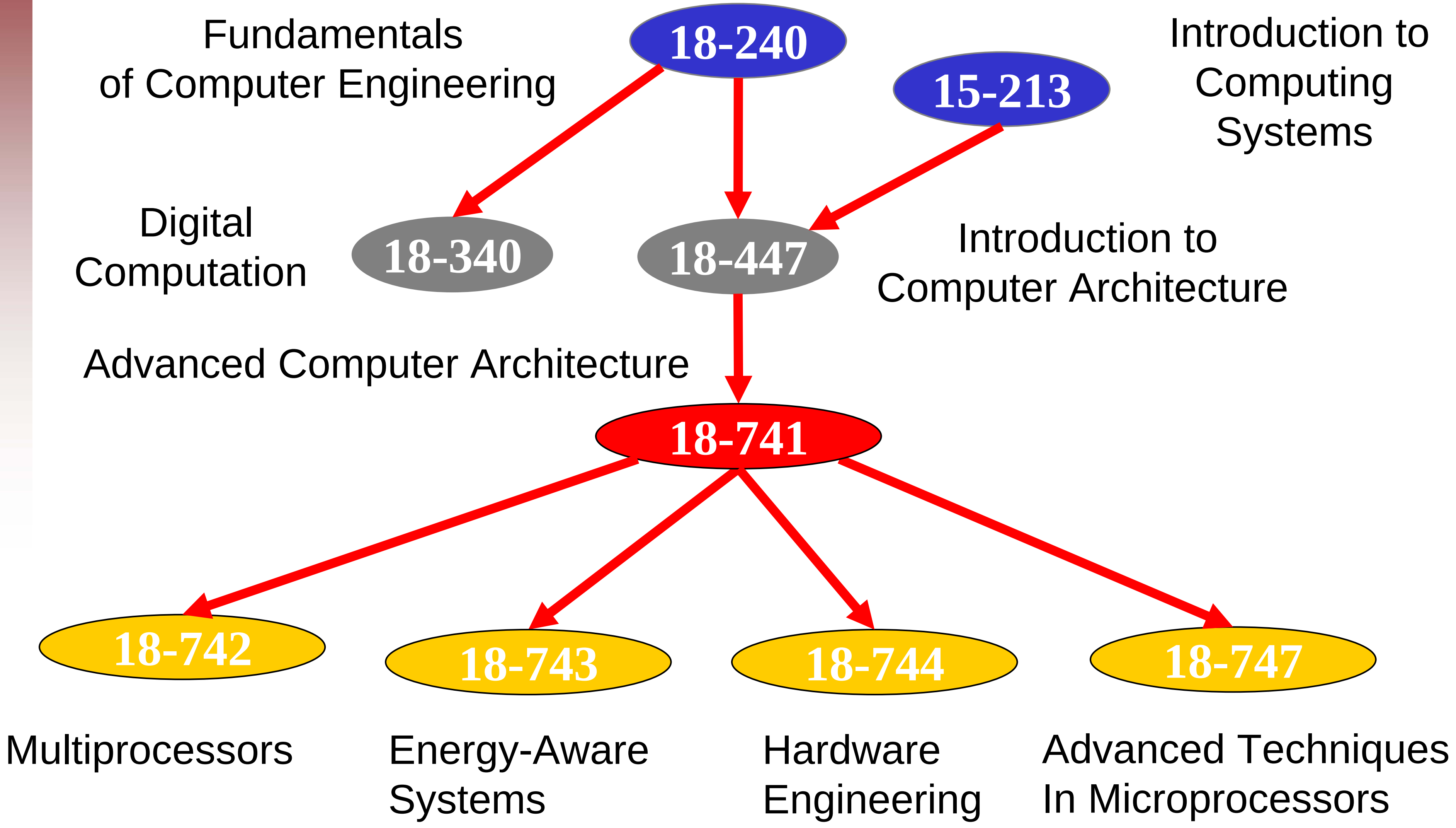


The Big Picture

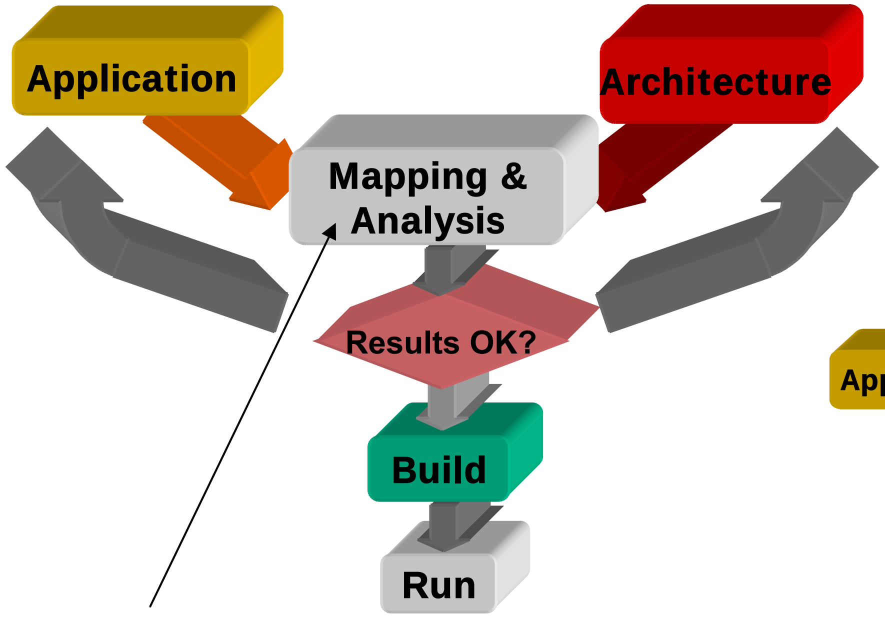
- Instruction Set Architecture
  - ♦ **= the interface** the architecture presents to user, compiler, & operating system
  - ♦ "Low-level" instructions that use the datapath & memory to perform basic types of operations
    - arithmetic: add, subtract
    - logical: and, or
    - data transfer: load, store
    - Control (jump to a location)
- Assembly language
  - ♦ **≠ ISA!**
  - ♦ The ISA is what the assembly language needs to know about the hardware

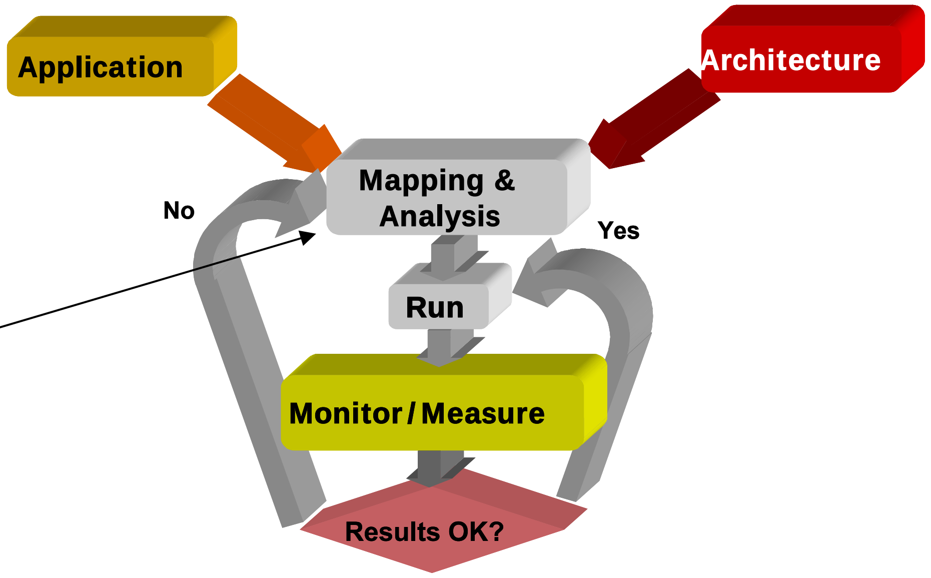


Where can you find more about these things?


Fundamentals of Computer Engineering
18-240
15-213
Introduction to Computing Systems
Digital Computation
18-340
18-447
Introduction to Computer Architecture
Advanced Computer Architecture
18-741
18-742
18-743
18-744
18-747
Multiprocessors
Energy-Aware Systems
Hardware Engineering
Advanced Techniques In Microprocessors

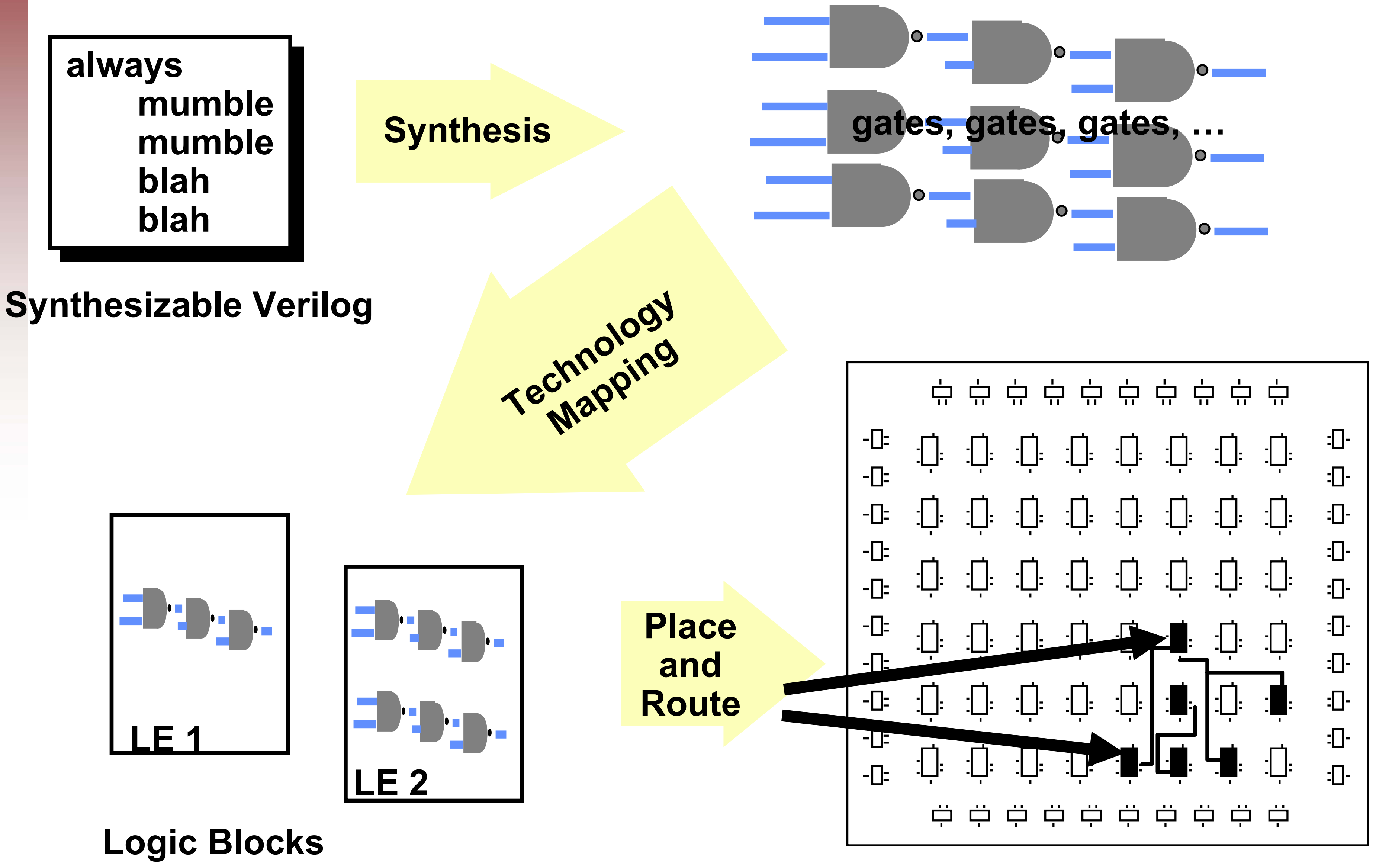
Recall: Two design methodologies



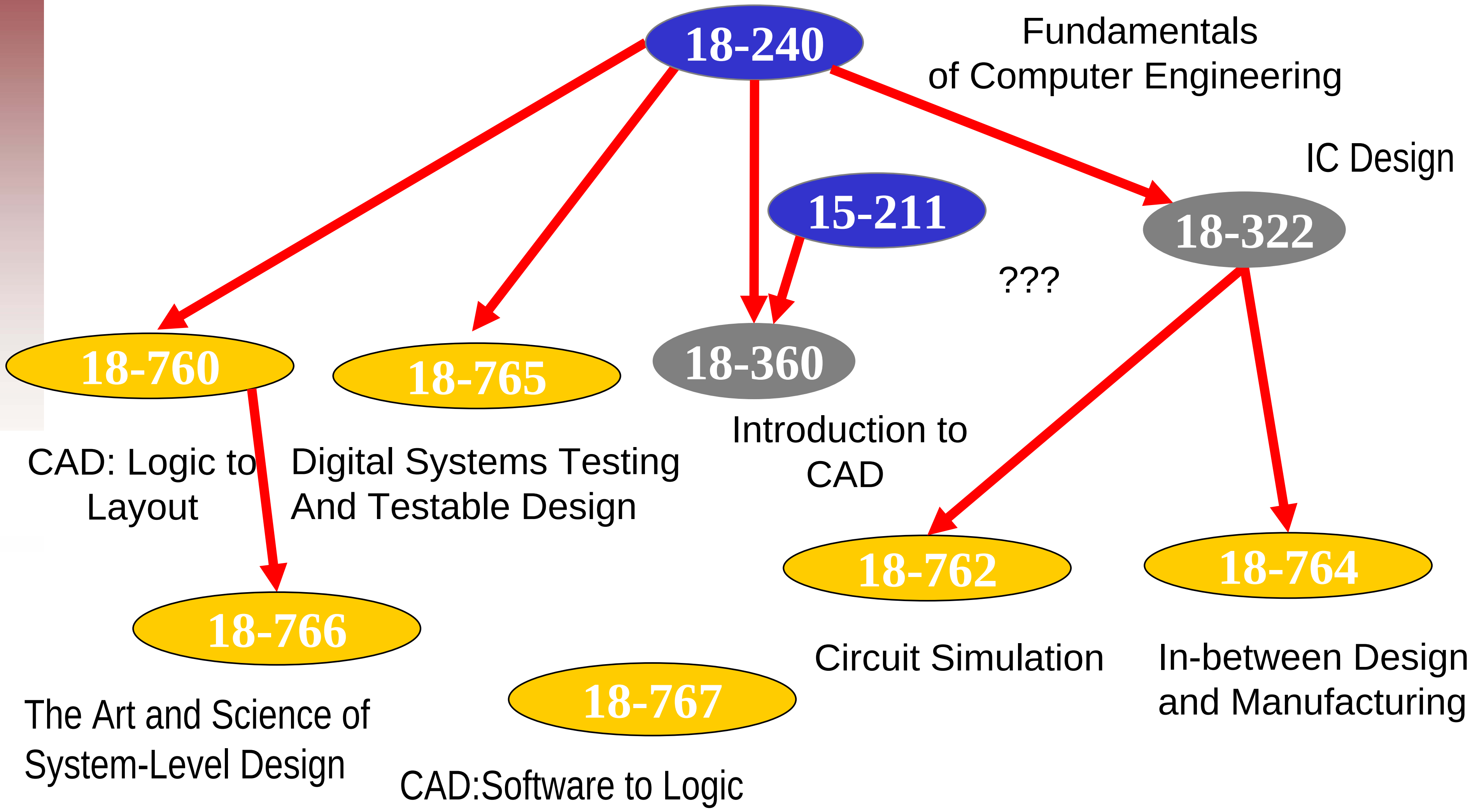


Here's where all design steps happen and design objective are optimized

Typical design methodology

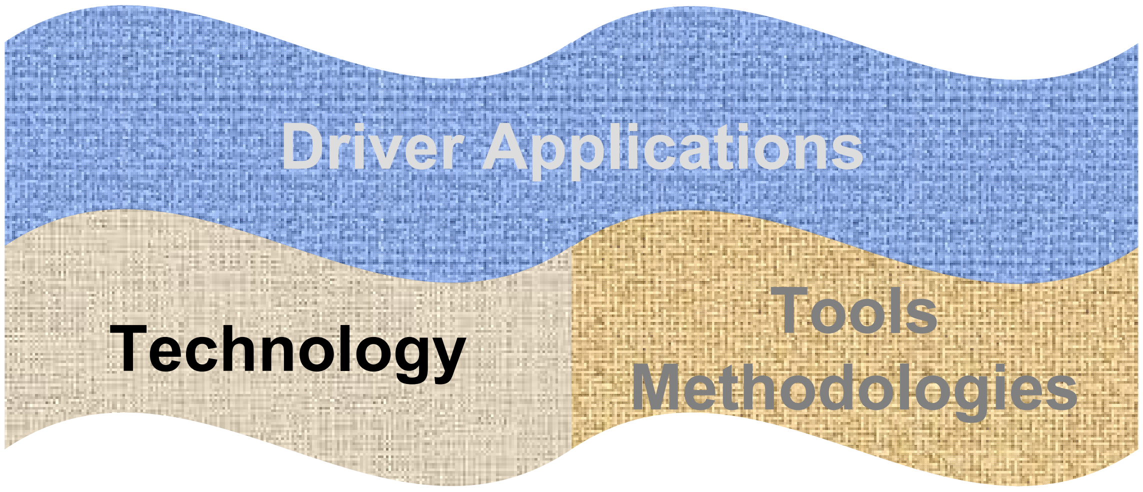


Where can you find more about these things?


18-240
Fundamentals of Computer Engineering
IC Design
15-211
18-322
???
18-760
18-765
18-360
Introduction to CAD
CAD: Logic to Layout
Digital Systems Testing And Testable Design
18-762
18-764
18-766
Circuit Simulation
In-between Design and Manufacturing
18-767
The Art and Science of System-Level Design
CAD:Software to Logic

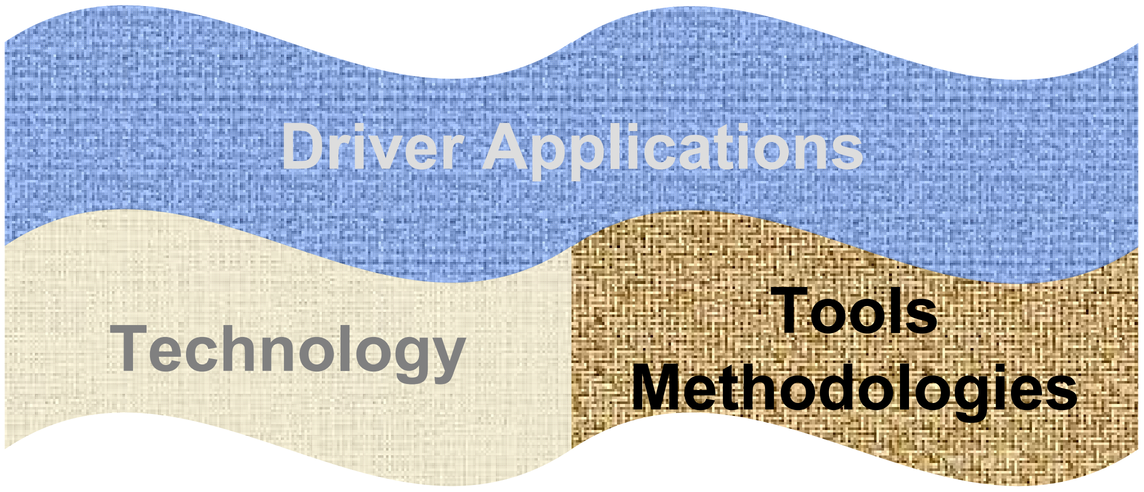
Back to AmI Systems...

How are we doing so far?....



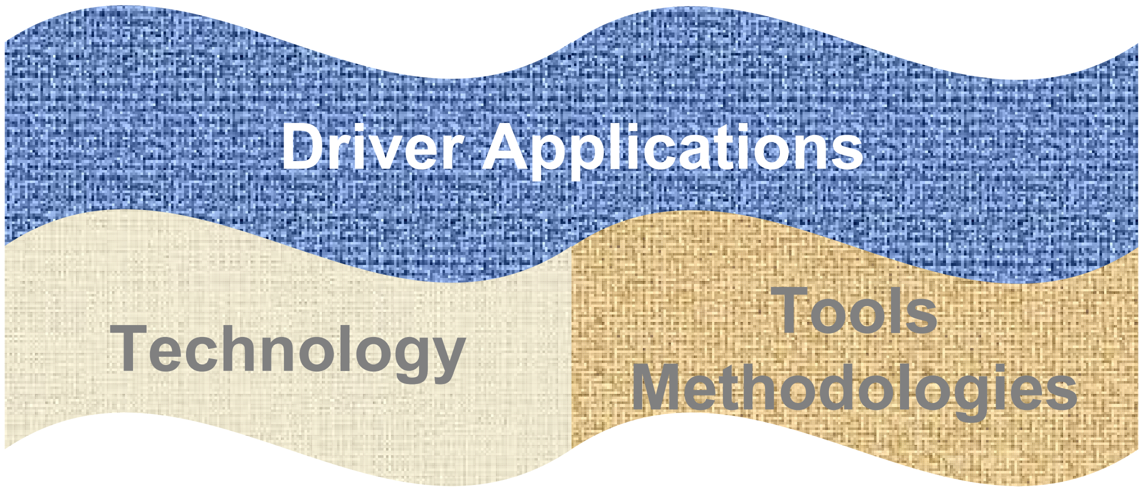
- Promising capabilities for…
  - ♦ Weaving interconnects (Infineon, ETH Zurich)
  - ♦ Battery technology and energy scavenging (UC Berkeley, ITN technologies)
- … Not so well in
  - ♦ Automating the process of embedding off-the-shelf components onto fabric

How are we doing so far?....



- Support for...
  - ♦ Dynamic fault-tolerance and power management (CMU)
  - ♦ Application remapping/reconfiguration (CMU)
- ... but need
  - ♦ Support for automatic application partitioning for a TAN

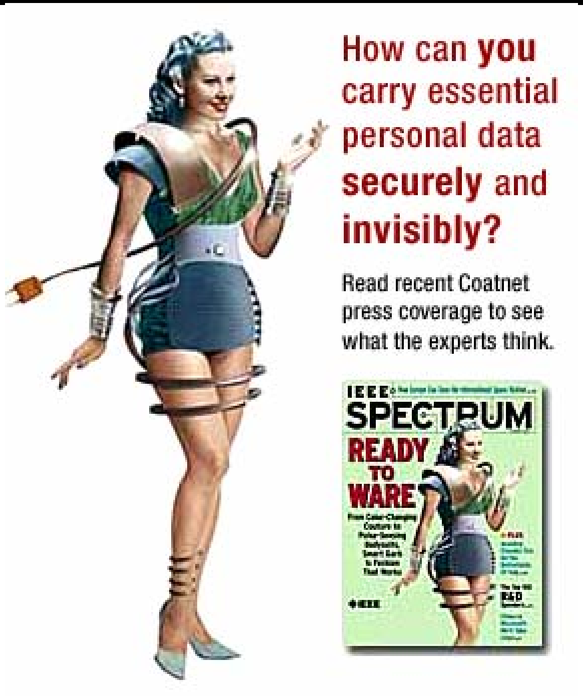
How are we doing so far?....

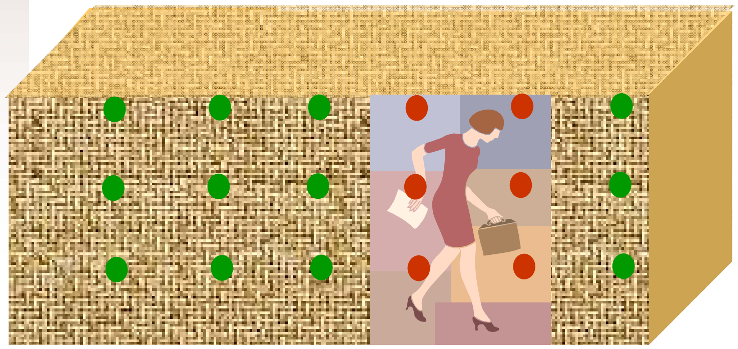


- Existing prototypes for
  - Medical - SIDS monitoring (GeorgiaTech)
  - Safety - Temperature sensor array (CMU)
  - Security – Ultrasonic, acoustic or imaging arrays (CMU)
  - Entertainment - MP3 on a sleeve (Infineon)
- Killer app?

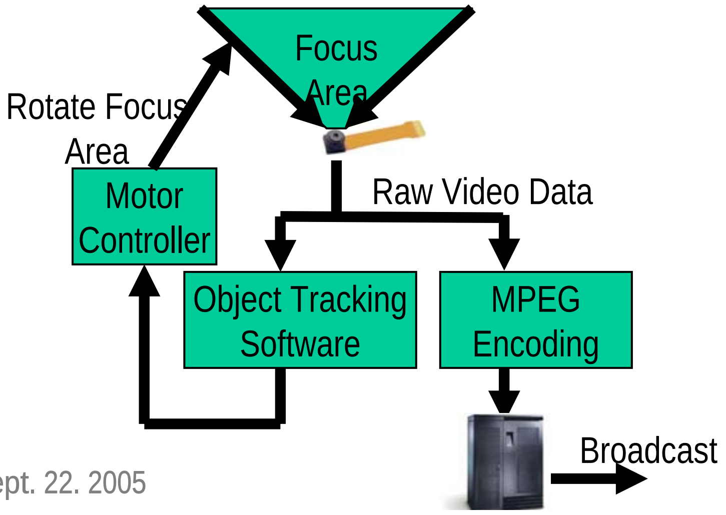
To probe further: CMU's COATNET Research Group

- Read IEEE Spectrum 2003 article!
  - Link available here: http://www.ece.cmu.edu/~etex
- Security/safety applications
  - Utrasonic/acoustic arrays
  - Temperature sensor arrays
  - Light sensor arrays


How can you carry essential personal data securely and invisibly?
Read recent Coatnet press coverage to see what the experts think.
IEEE SPECTRUM
READY TO WARE



- Smart teleconferencing systems
  - Motion tracking cameras – WallWatchers project



...