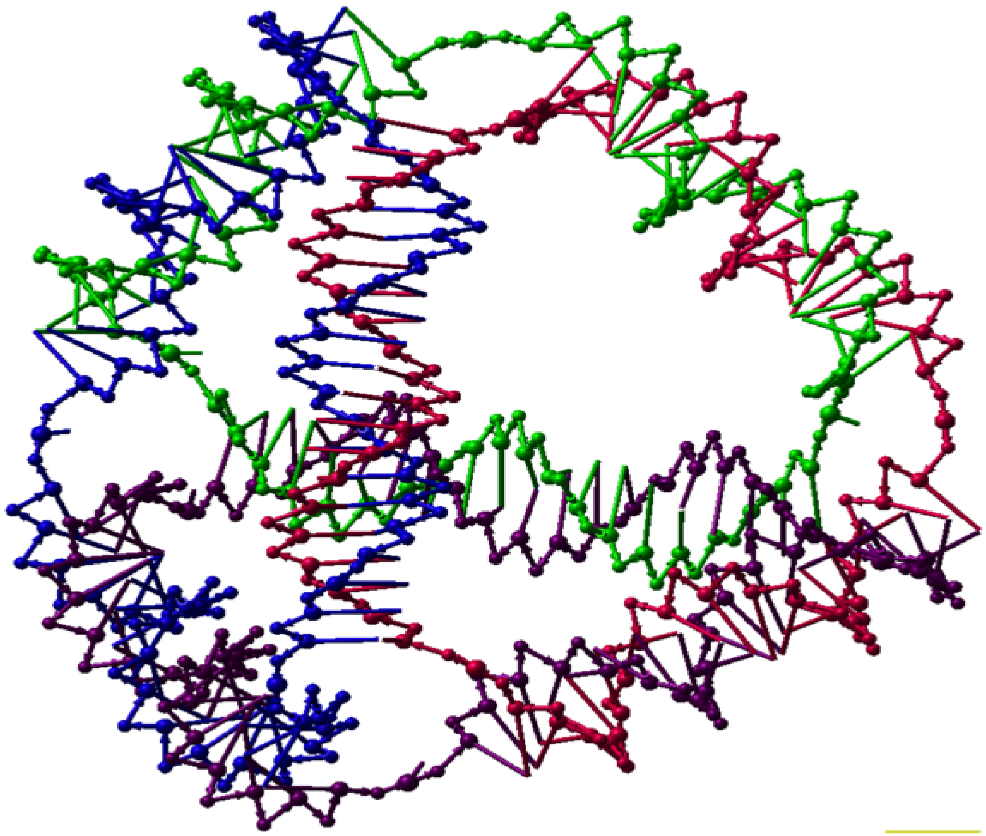
# Nanotechnology & Computer Architectures



Davide Patti (v. Jan 2014.01.15)
davide.patti@dieei.unict.it

# Nano what?!...

- Manipulation of matter at atomic/molecular scales (1 to 100 nm)
  - Smallest life: Mycoplasma bacteria (200 nm)
  - Smallest atom: hydrogen diameter (0.25 nm)
- <u>Quantum effects:</u>
  - new physical properties, not miniaturised versions of larger devices
  - Transparency, solubility, conductivity

# Growing interest

- Application fields: nano-materials, nano-medicine agents, environment (nanofilters), semiconductors, smart food packaging, http://en.wikipedia.org/wiki/List_of_nanotechnology_applications
- Huge research effort: USA 3.7 billion dollars, EU 1.2 billion and Japan 0.75 billions (2012)

# Top-down vs bottom-up Design

- **Top-down**: control on placement of system components (e.g. Photolitography mask to induce a pattern)
- **Bottom-up**: Rely on *local molecular interactions* to build large-scale structures
- *Example:wooden form vs building a wall assembling stones*
- Biology inspired (...please avoid *“grey goo*” due exponentially self-replicant nanorobots)

# In Computer Design...

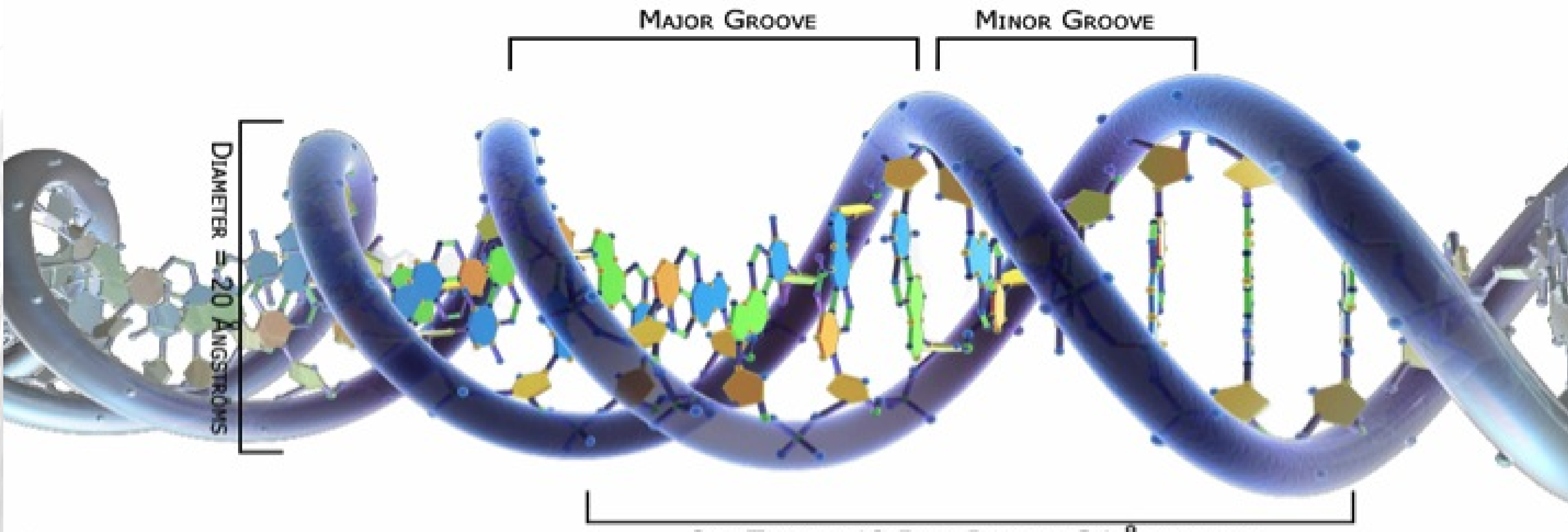
- **Top-down** is the traditional approach: layout mask to specify the computer system structure used by semiconductor industry to place components
- **Bottom-up**: specify only nano components, NOT their placement. The same properties of each component will allow them attach each other, so the system can be defined as "*self-assembled*"

# Self-Assembly

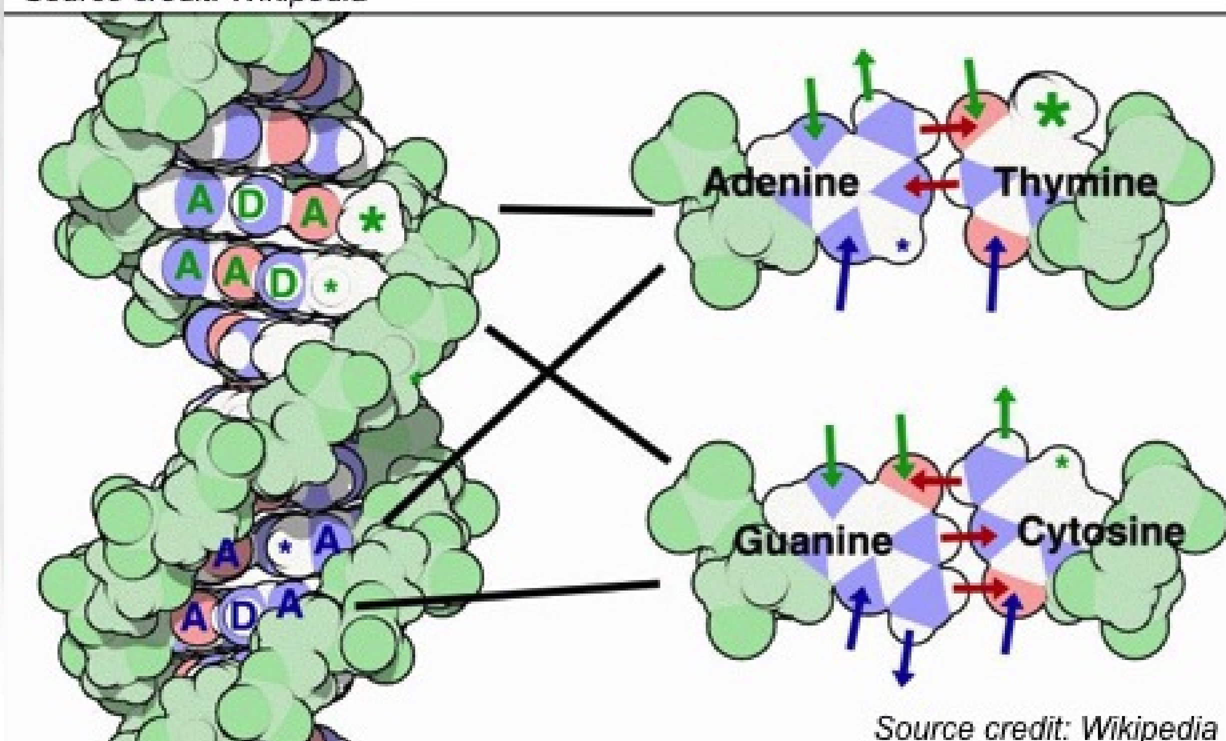
- Using bottom-up approach, elements self assemble to form a complex system
- **Trivial:Random SA** *(everyone can be everywhere...)* → NO imposed order = little customizable complexity (given N components, all random system have similar behaviour)
- **Programmable SA**: specify how components attach to one another, BUT NO where the will be placed

# DNA self-assembly

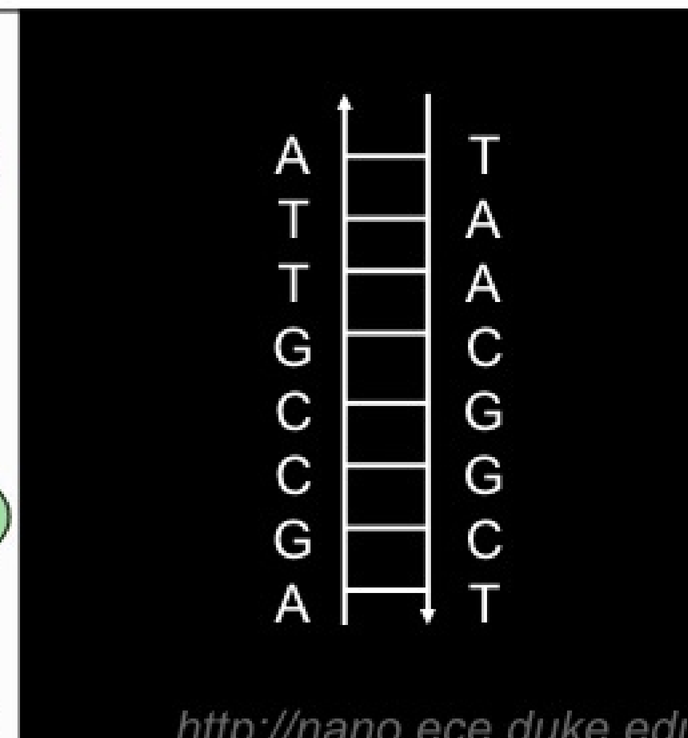
- Sequence of nucleobases A, G, T, C
- Stable structure when complementary nucleobase sequences match, that is:
  - A pairs with T
  - G pairs with C
- The result is an helix of 2nm diameter
- Larger blocks of assembled DNA sequences, called “motifs” can self-assembled to create more complex structures




*Source credit: Wikipedia*

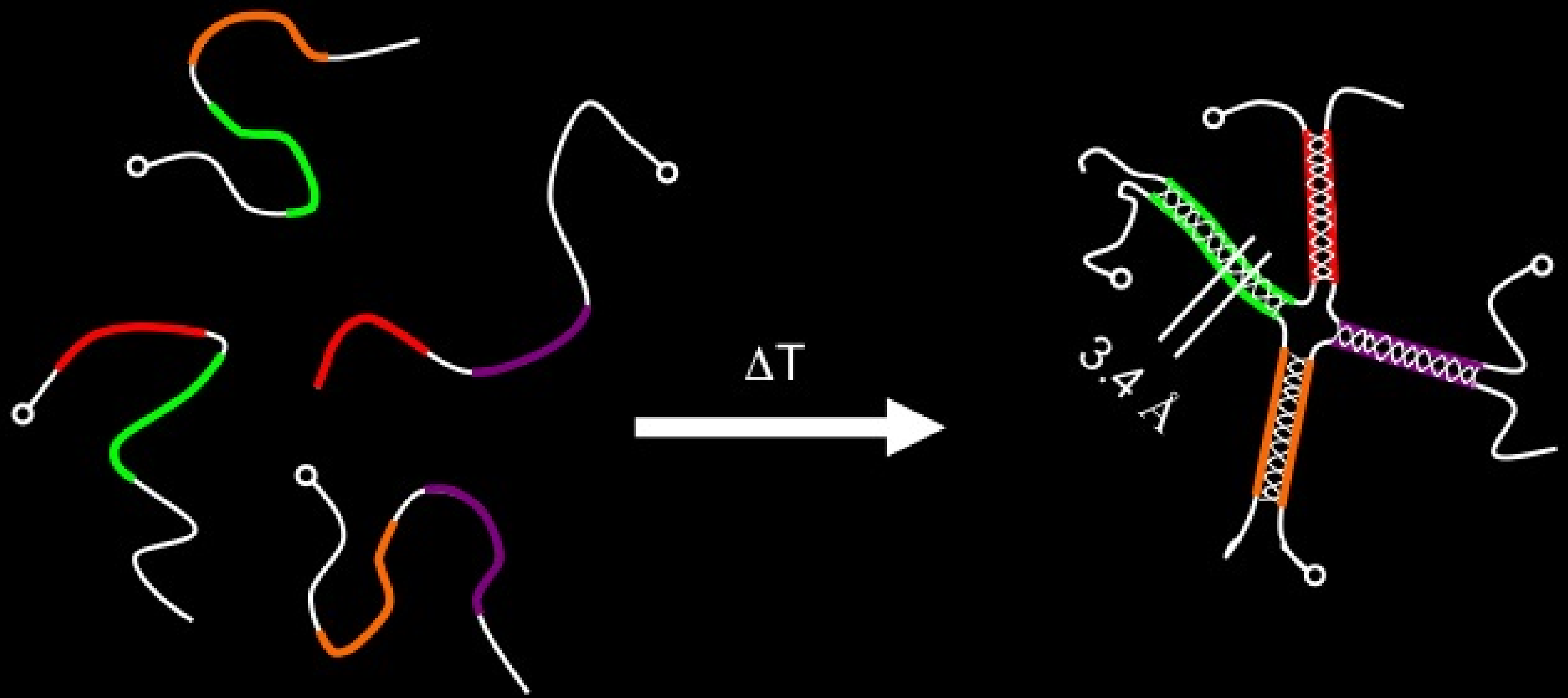



*Source credit: Wikipedia*



# DNA Scaffolds - Geometry

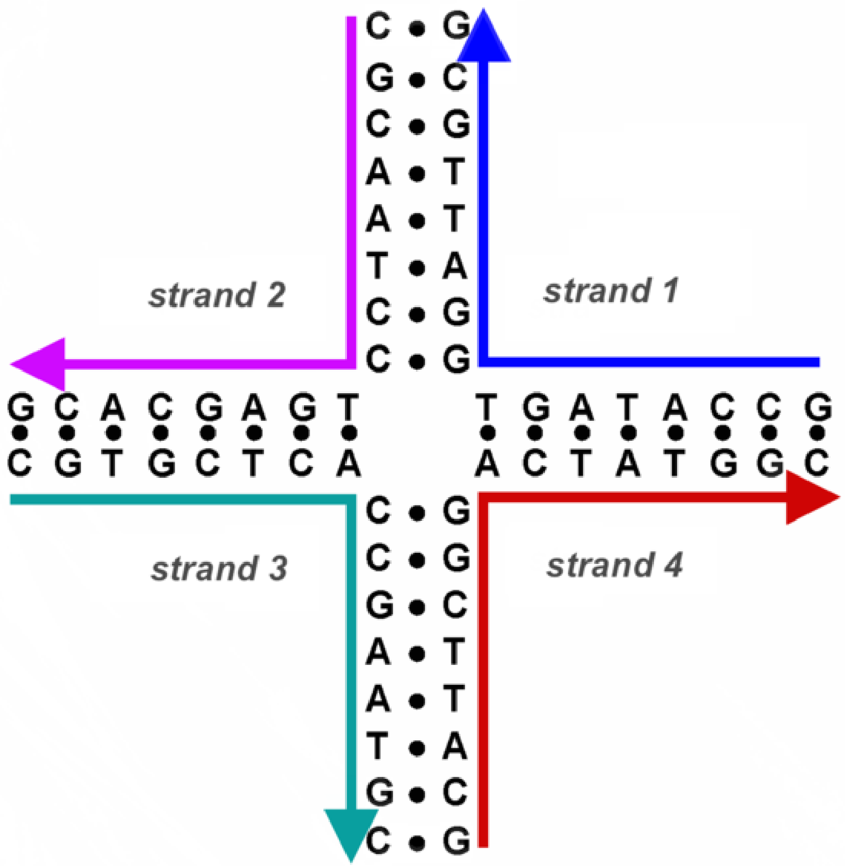
- The geometric properties of double strands can form specific, controlled self-assembled nanostructures:



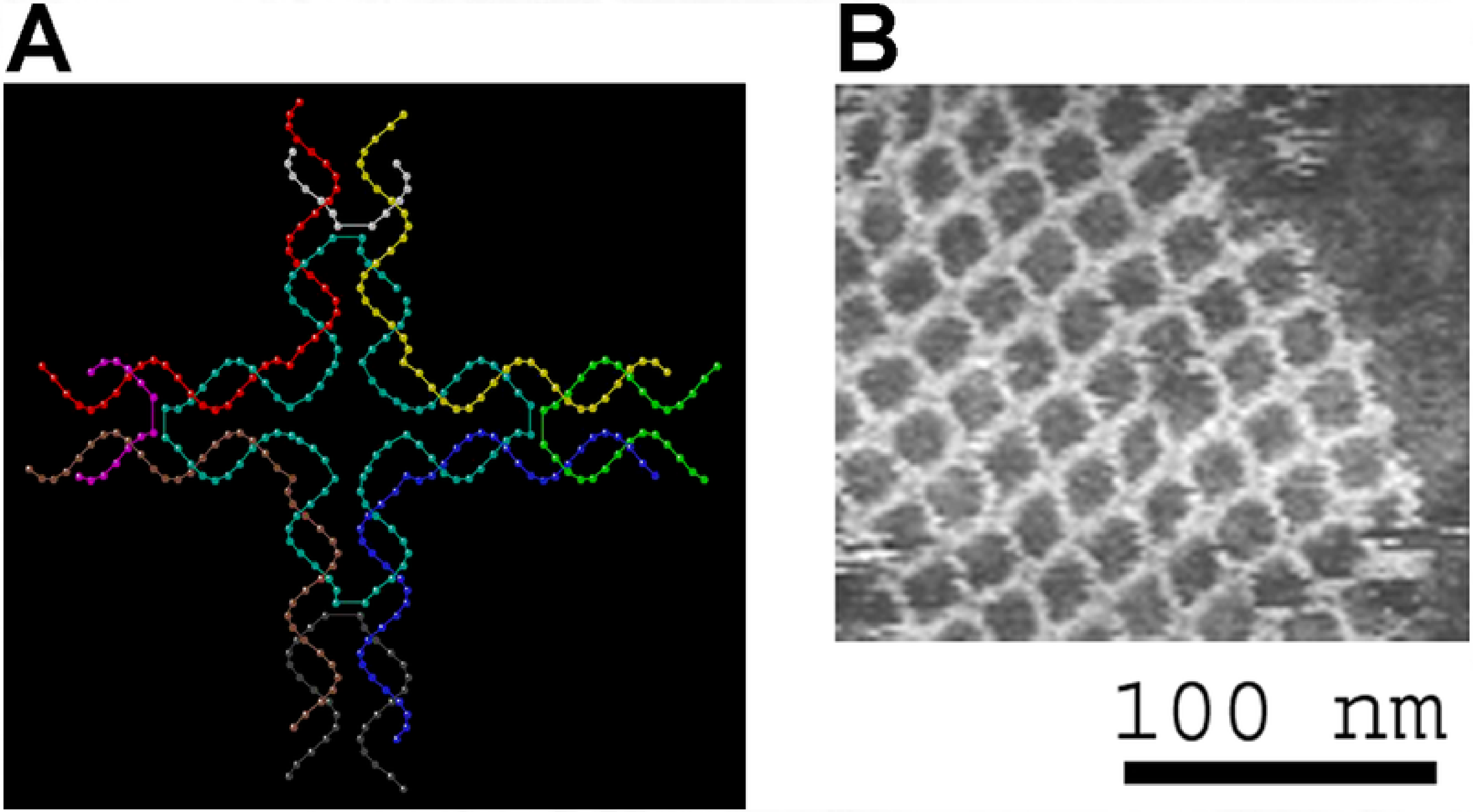

C. Dwyer, Duke University

http://nano.ece.duke.edu

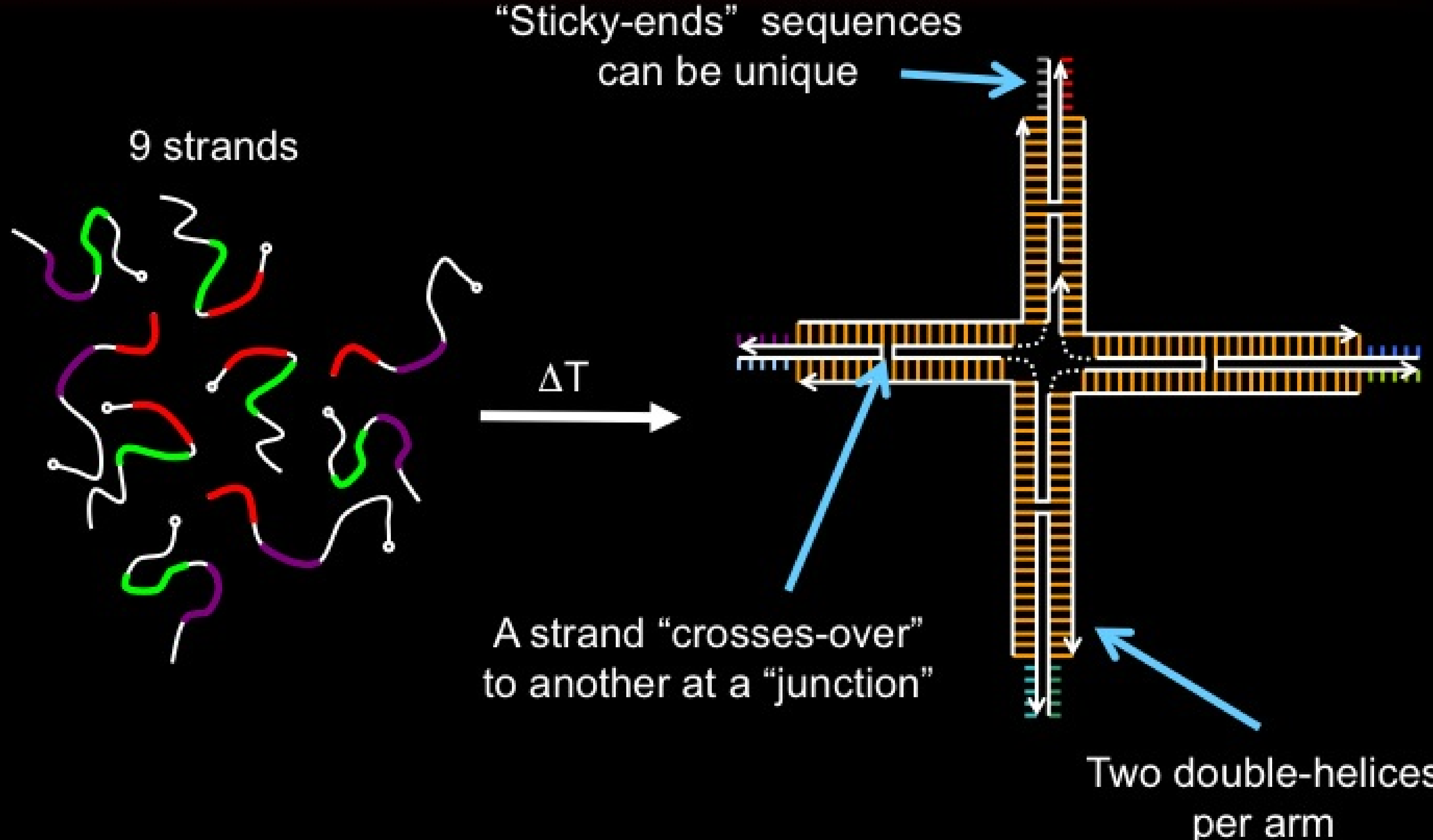
# Simple 4-arm junction



# ...more complex structures

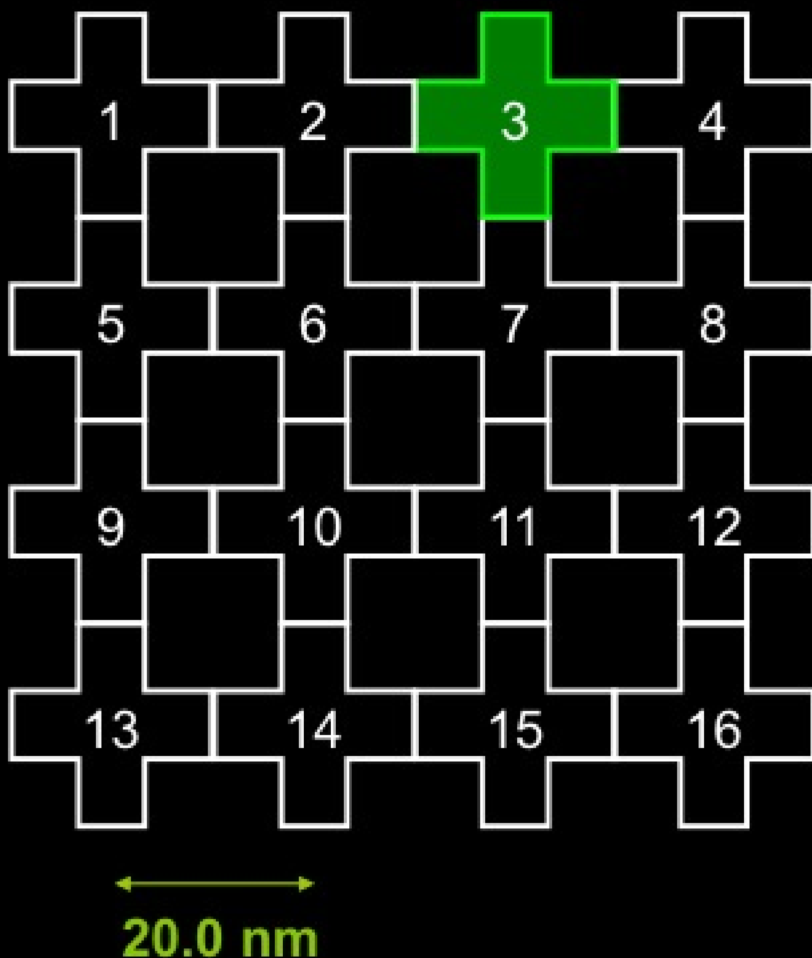


# DNA Motifs



"Sticky-ends" sequences can be unique
9 strands
ΔT
A strand "crosses-over" to another at a "junction"
Two double-helices per arm

# DNA Motifs




Atomic Force Microscopy (AFM) Image
(360nm X 360nm)

C. Dwyer, Duke University

http://nano.ece.duke.edu

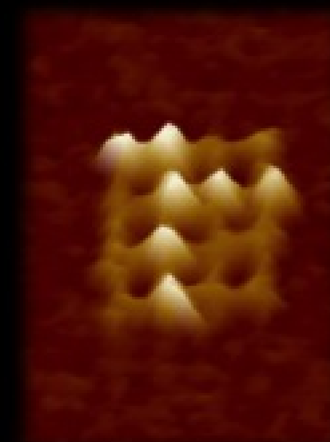
(davide.patti@dieei.unict.it)

# DNA Nano-grids

Self-assembled grids with sub-nm resolution (3.4 Å)

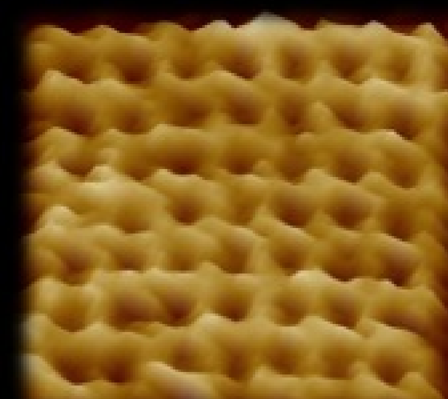
**Multiple DNA grids deposited on flat mica plane**


**16-tiles**



**Patterned grid**

**64-tiles**



**Manufacturing scale: $>10^{15}$ grids/mL**

**(< 8 hours)**

C. Dwyer, Duke University

http://nano.ece.duke.edu

# Placing nano-devices

- **DNA Tag**: a particular sequence of nucleobases (*eg GATTACA, TCGTAAT, etc..*)
- **Nano devices**: *nanotubes wires and CNFET transistor*, each with specific DNA tags
- DNA structures can provide a "scaffold" onto which nano devices can be attached binding to complementary DNA tags
- **Design** = Specify the appropriate DNA tags in order to attach nano device terminals

# Placing nano-devices

- *Example: 2-input NAND → 10 terminals to attach (transistor + wires)*
- Note that a basic useful circuit could consist of thousands NAND gates, that is 10.000s terminals to bind
- Adding complexity:
  - In CMOS: larger masks
  - In DNA SA: more different unique tags
- How many tags ?

# How many tags ?

- Ideally: choose where every single transistor will be placed-> *a different tag for each terminal*
- More different unique tags (of a given lenght):
  - more customizable complexity
  - Tags similar to each other-> ... more probability of improper matching (similar to “hamming-distance”)
- Conflicting goals: defects rate vs complexity

# CMOS vs CNFETs

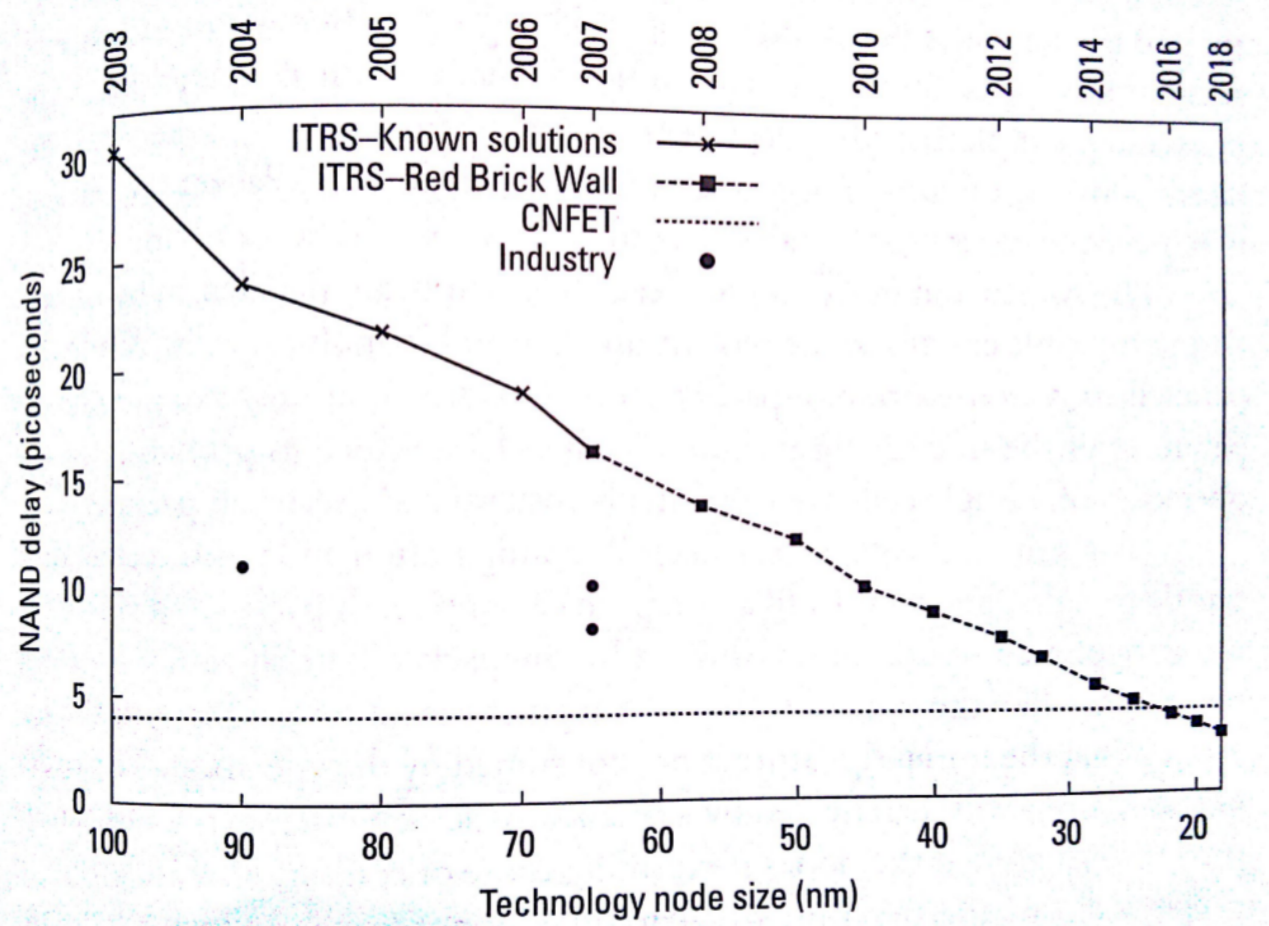



**Figure 7.3** Nanoscale device performance.

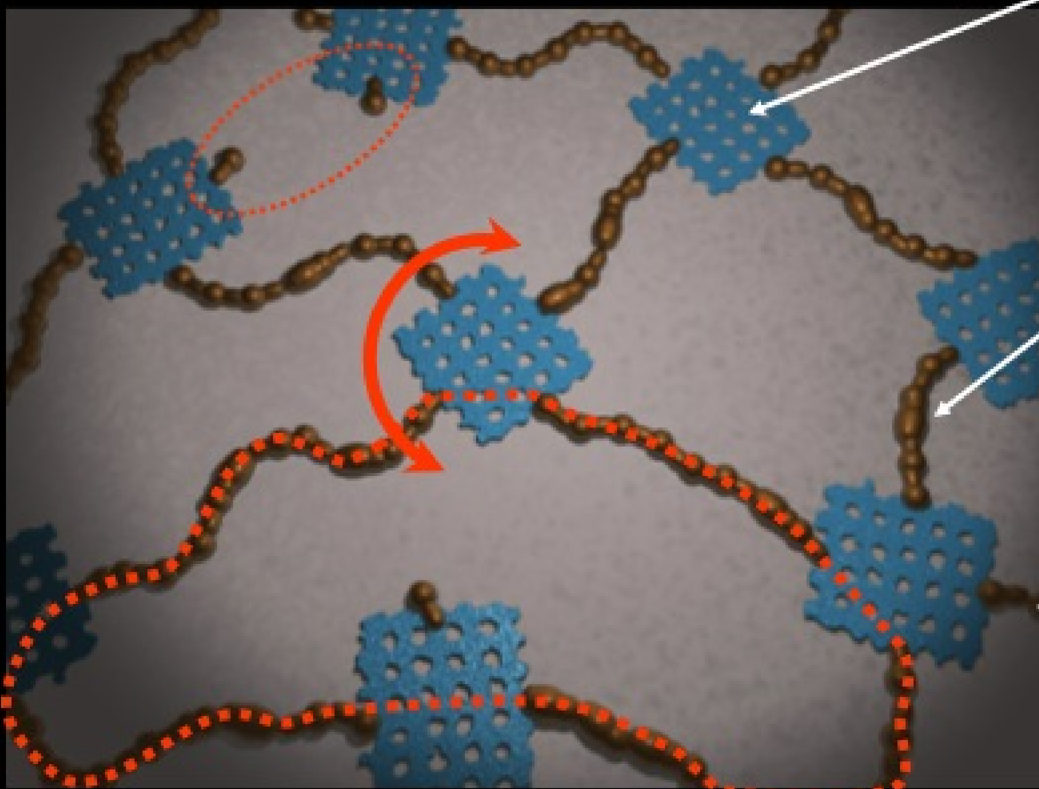
# Defect tolerance

- *Functional*: a devices does not behave like it should (a transistor does not conduct)
- *Positional*: A device is placed where it shouldn't. Typical of DNA self-assembly
- CMOS doesn't have much defect tolerance
- Instead, in self-assembly, the more complexity we need → the more unique tags → more tolerance needed

# Interconnecting nodes

- The size (max 10.000 CNFETs) of blocks is limited by the DNA grid size, due the "defect tolerance/number of tags" conflict discussed above
- So, to increase computing capacity, multiple blocks must be interconnected
- System architects should explicitly partition the designs in smaller functional nodes

# Self-Organizing Architectures



Self-assembled
Computational nodes
*(minimal computational power & defect-prone)*

Self-assembled
Interconnect
*(defect-prone)*

Distant micro-scale
I/O contact
*(low bandwidth)*

Defect model includes:
- Rotation, position
- Connectivity
- Defective devices on nodes

*C. Dwyer, Duke University* *http://nano.ece.duke.edu*

(davide.patti@dieei.unict.it)

# Design flow

- Architectural description
- Behavioural simulator to verify the high-level procedural model (e.g. System C)
- Gate-level modules implementing the system
- Transistor layout is verified
- DNA sequences (using a prefixed number of unique tags)

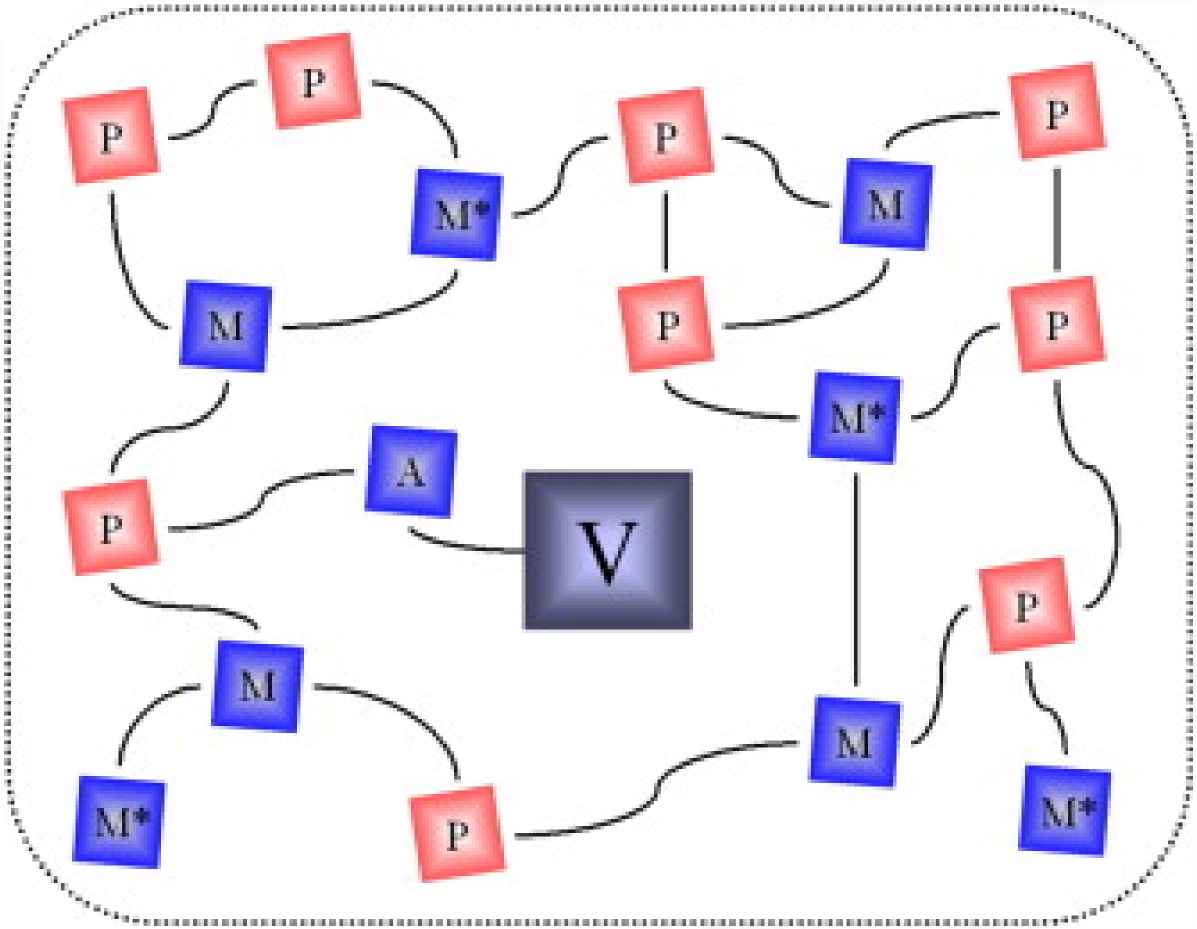
# Challenges

- **Small-scall control**: nodes with limited space, communication, coordination
- **Large-scale randomness**: node placement, orientation, connectivity
- **High defect rates**

# Architectural Implications

- Partition functionalities in order to exploit multiple small nodes
- Execution model (appropriate instruction set)
- Memory system (distributed accross nodes)
- Routing: limited space for complex dynamic routing, no guarantees on node placement and connectivity to use static routing
- Interfacing to microscale

# System Model




Nodes:

- processing (P),
- memory (M),
- memory ports(M*),
- Anchor via to microscale (A-V)

# NANA system features

- Each node generates its own clock (e.g. 10GHz, still pessimistic looking at CNFETs)
- Accumulator based ISA to minimize coordination among nodes
- Packets contain operations and operands in the appropriate order
- A processing node performs the operation and removes operands
- http://www.cs.duke.edu/~alvy/papers/nana.pdf

# Packet format

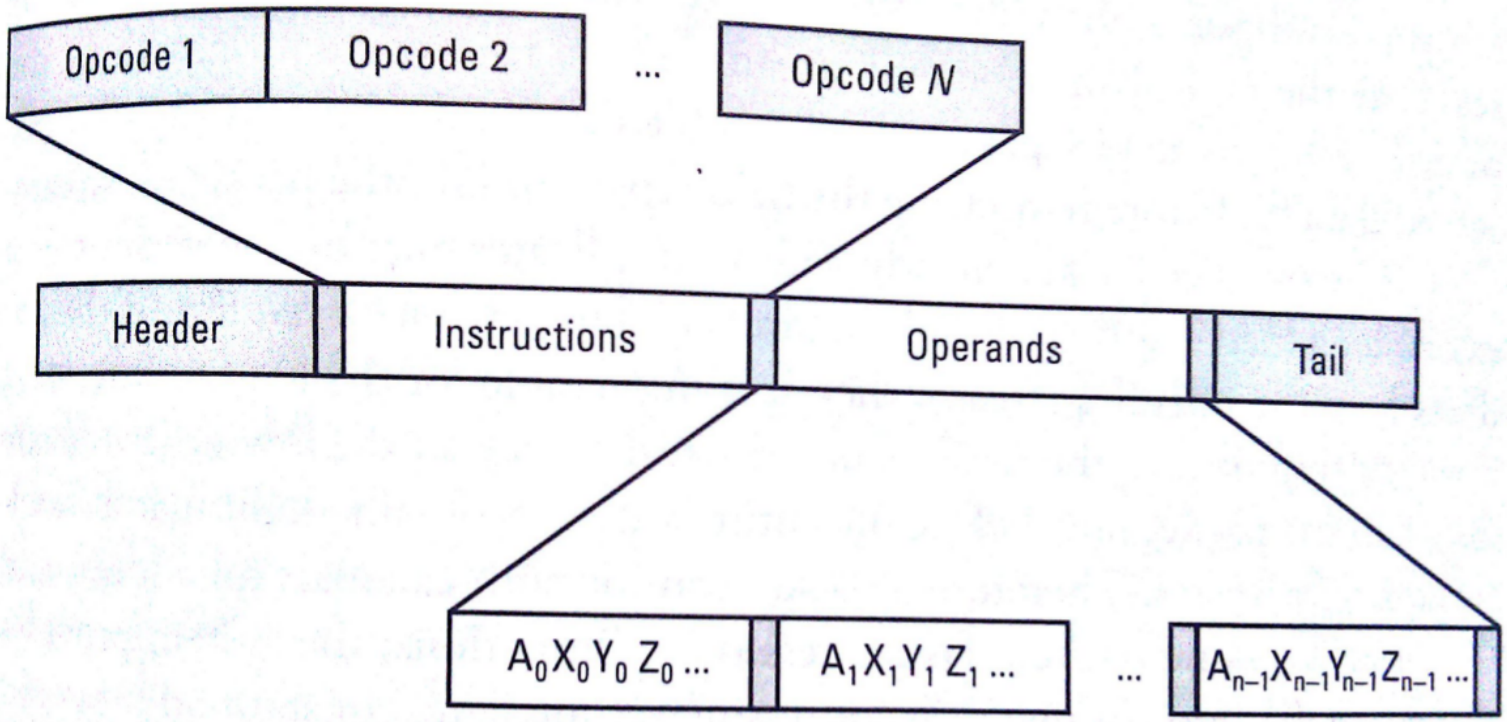


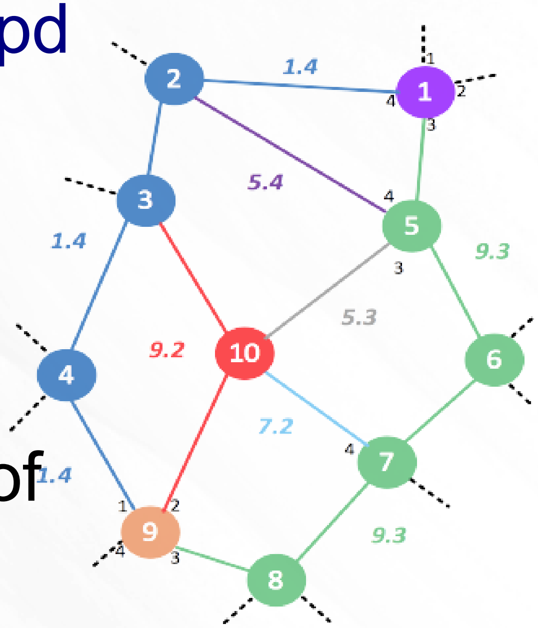

**Figure 7.4** Packet format.

# Finding resources for execution

- Packets travel in the network and must be able find the appropriate node type:
  - without deadlocking
  - In a irregular topology of a randomly interconnected *sea of nodes*
  - Limited node size → no large buffers or complex circuitry
- Must exclude hardware hungry solutions: *virtual channels, resource redundancy, dynamic recovery*

# Segment-based Routing

http://www.disca.upv.es/jflich/papers/ipdps06.pdf




- Partitions the topology into different disjoint paths called “**segments**”
- Each segment connects two other segments (i.e. starts/ends into nodes of another segment)
- Deadlock freedom from turn-prohibition: prohibition of a turn for each segment
- Topology agnostic, but **…require topology graph as input!**

# Nanoxim: Distributed Segment-based Routing (DiSR)

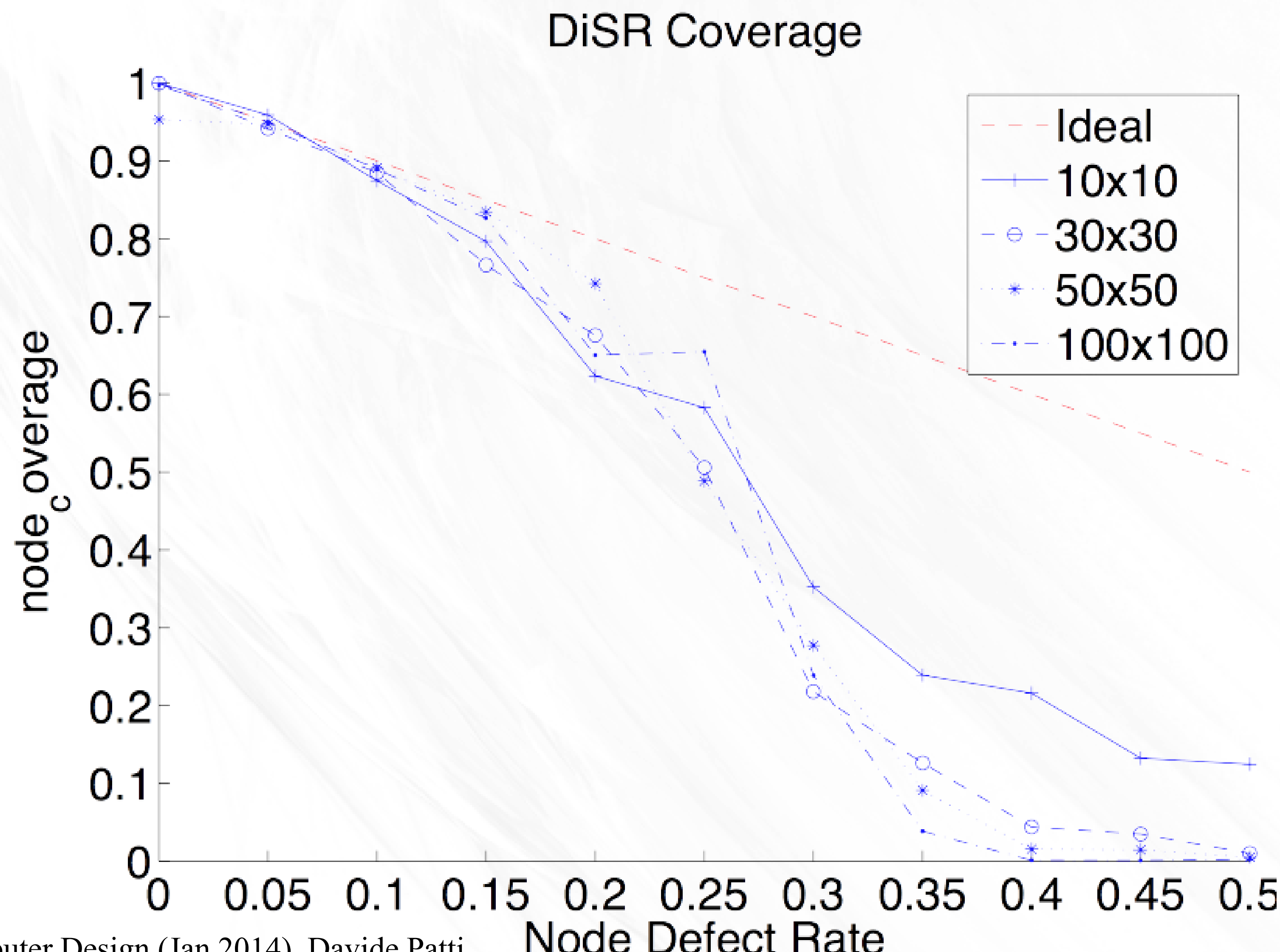
- DiSR: same properties as SR
- <u>No topology graph required</u>
- Each node separately contributes to a distributed process that estabilishes segments
- Special packets to discover the network topology and impose segments structures
- SystemC opensource platform:

http://code.google.com/p/nanoxim/

# Proposed Activities

- Comparison against simple Up*/Down* spanning tree based
- Quality measures: % of coverage at different node defect rates
- Logic required at each node
- Topology singularities
- Parallel Applications

# Node Coverage



# Setup Latency