Top-Down? Bottom Up?

A Survey of Hierarchical Design Methodologies

Trent McConaghy
@trentmc0

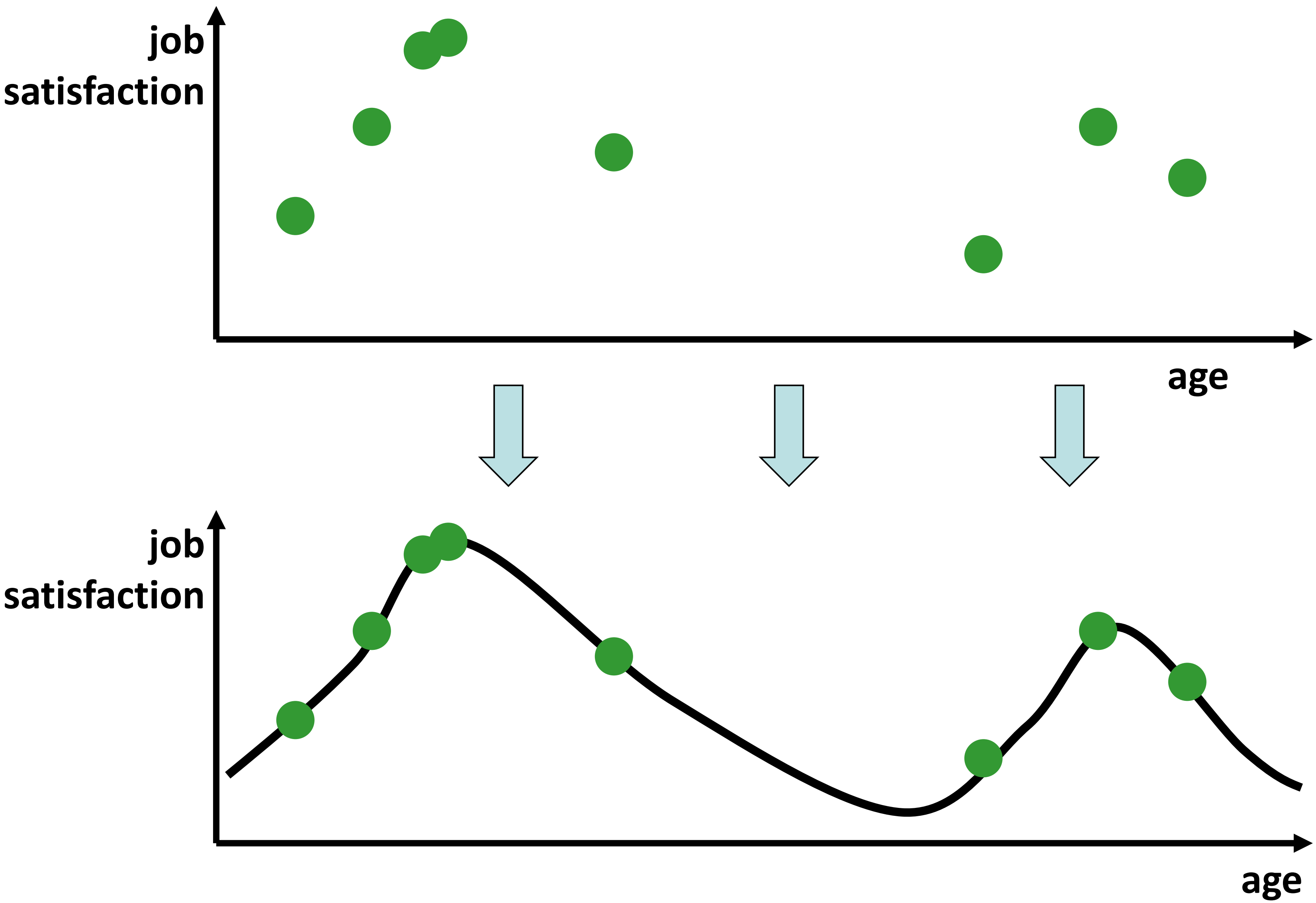
Pragmatic Ways to View AI, by Example

Classification, in 2D

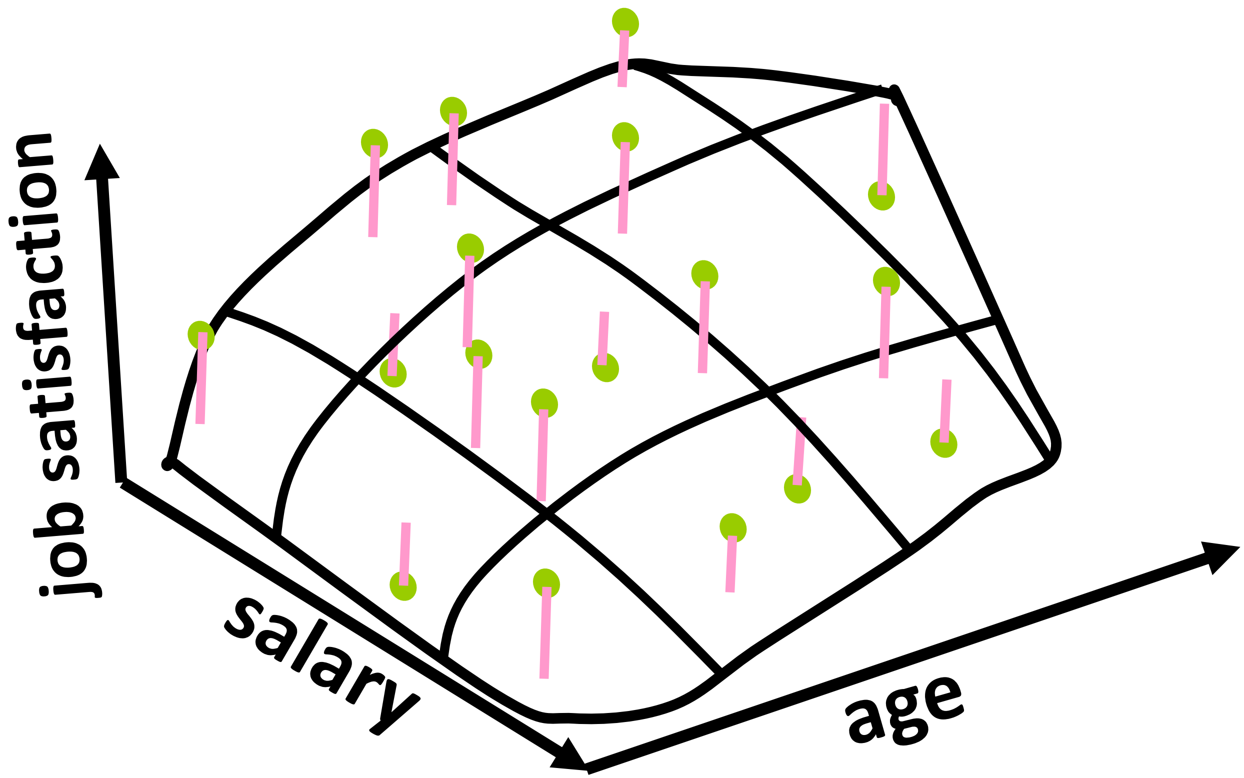
Credit profile:

- Paid bills
- Didn't pay

Regression, in 1D



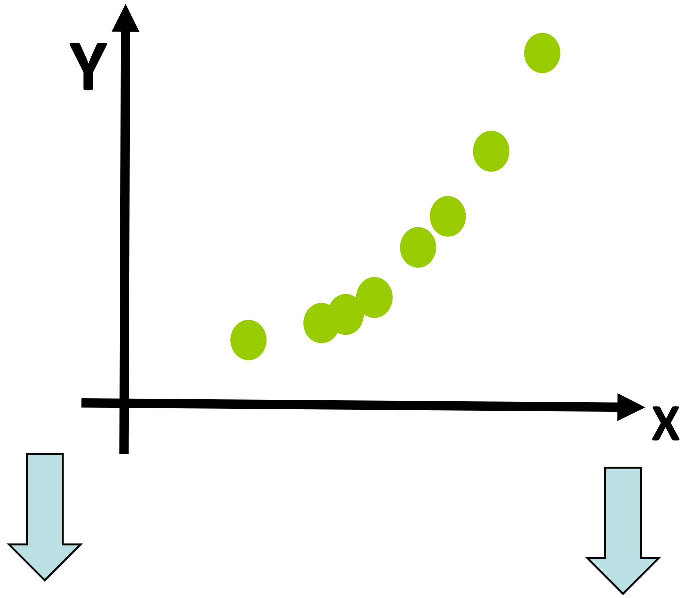
Regression, in 2D

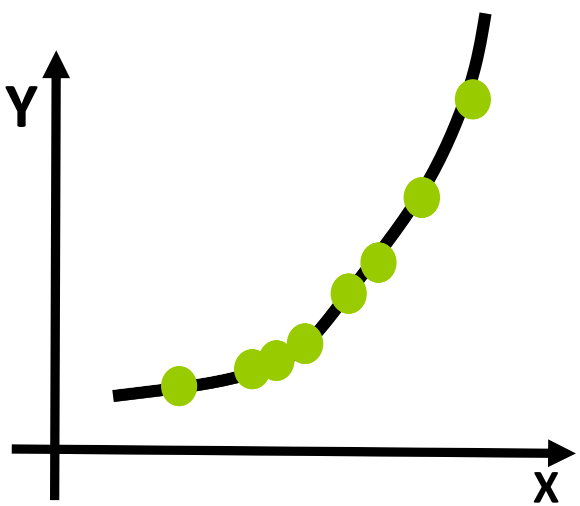


How: Polynomials, splines, neural networks, support vector machines, Gaussian process models, boosted trees, … [many refs]

Symbolic Regression (SR)

(Like regression, but output a symbolic model too)





<u>**Symbolic model:**</u>

$$y=50.2 + 9.1 \bullet x$$
$$+ 3.2 \bullet \max(0, x^2)$$

[e.g. McConaghy 2005;
McConaghy 2011]

AI Has a Toolbox of Ways to Solve…

- Classification – Fraud detection, spam filtering …
- Regression – Stock prediction, sensitivity analysis …
- Whitebox regression – Scientific discovery …
- Optimization – Airfoil design, circuit simulation …
- Structural synthesis – Analog synthesis, robotics …
- Pattern recognition – Face recognition, object recog …
- System identification – Scientific discovery …
- Ranking – Web search, ad serving, social discovery …
- Control – Auto-driving autos, spacecraft trajectories …
- …

AI Sub-fields

- machine learning
- neural networks
- evolutionary computation
- fuzzy logic
- data mining
- artificial general intelligence
- pattern recognition
- ..
- (nee) nonlinear programming
- (nee) databases
- ..

AI Sub-fields of sub fields

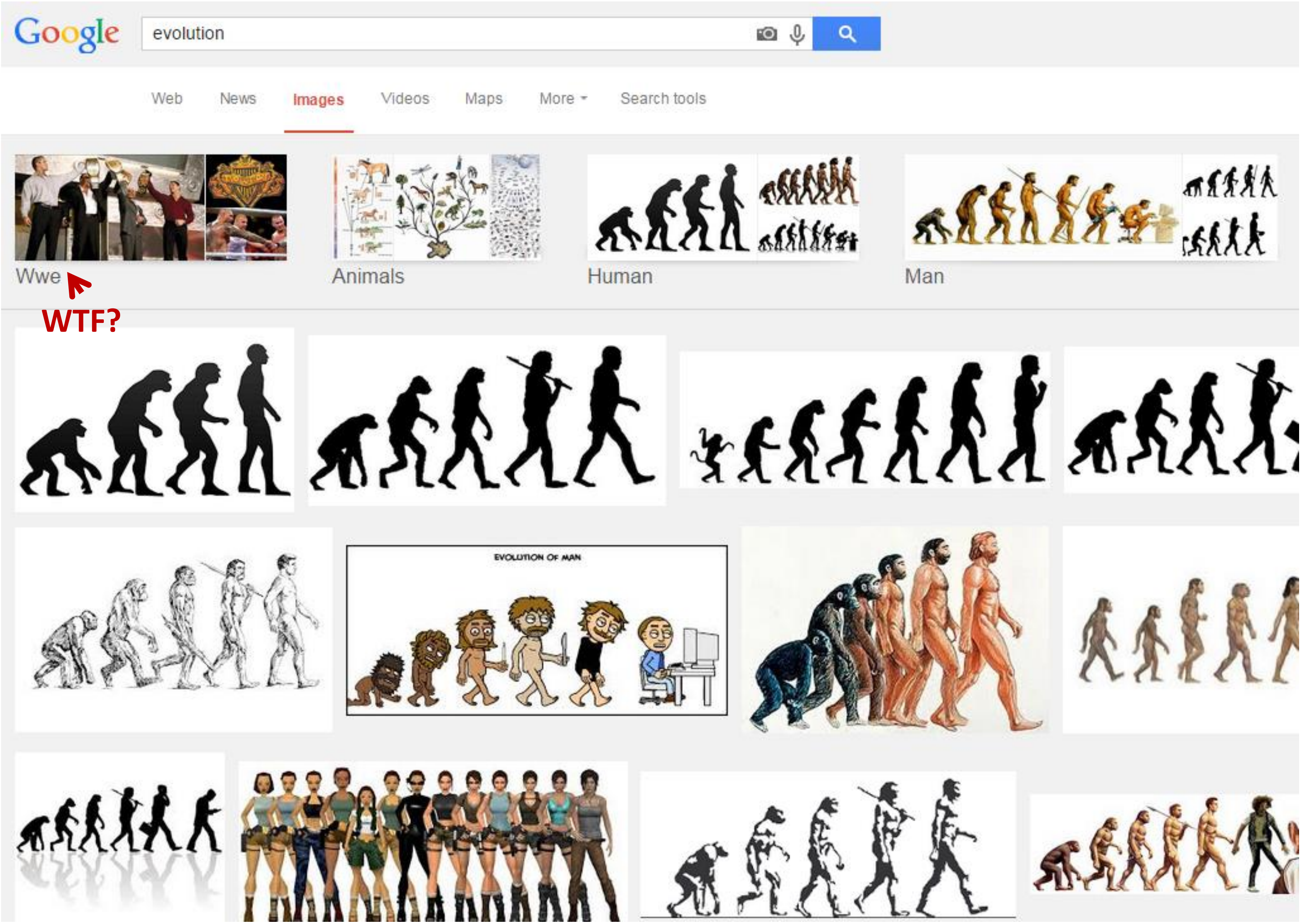
- machine learning + neural networks
  - recurrent neural networks
  - sparse linear regression
  - **deep learning**
  - ..
- evolutionary computation
  - evolutionary programming, evolution strategies
  - genetic algorithms
  - **genetic programming**
- ..

Genetic Programming (GP):

A branch of a branch of AI

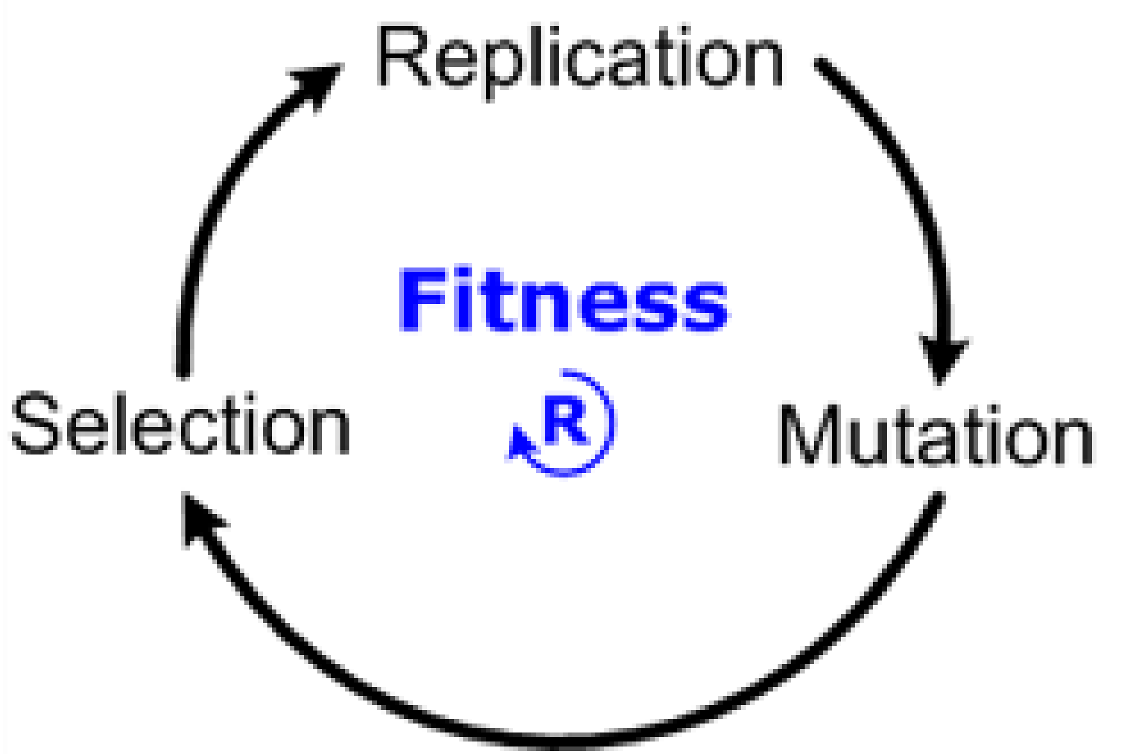
But a super-cool one..

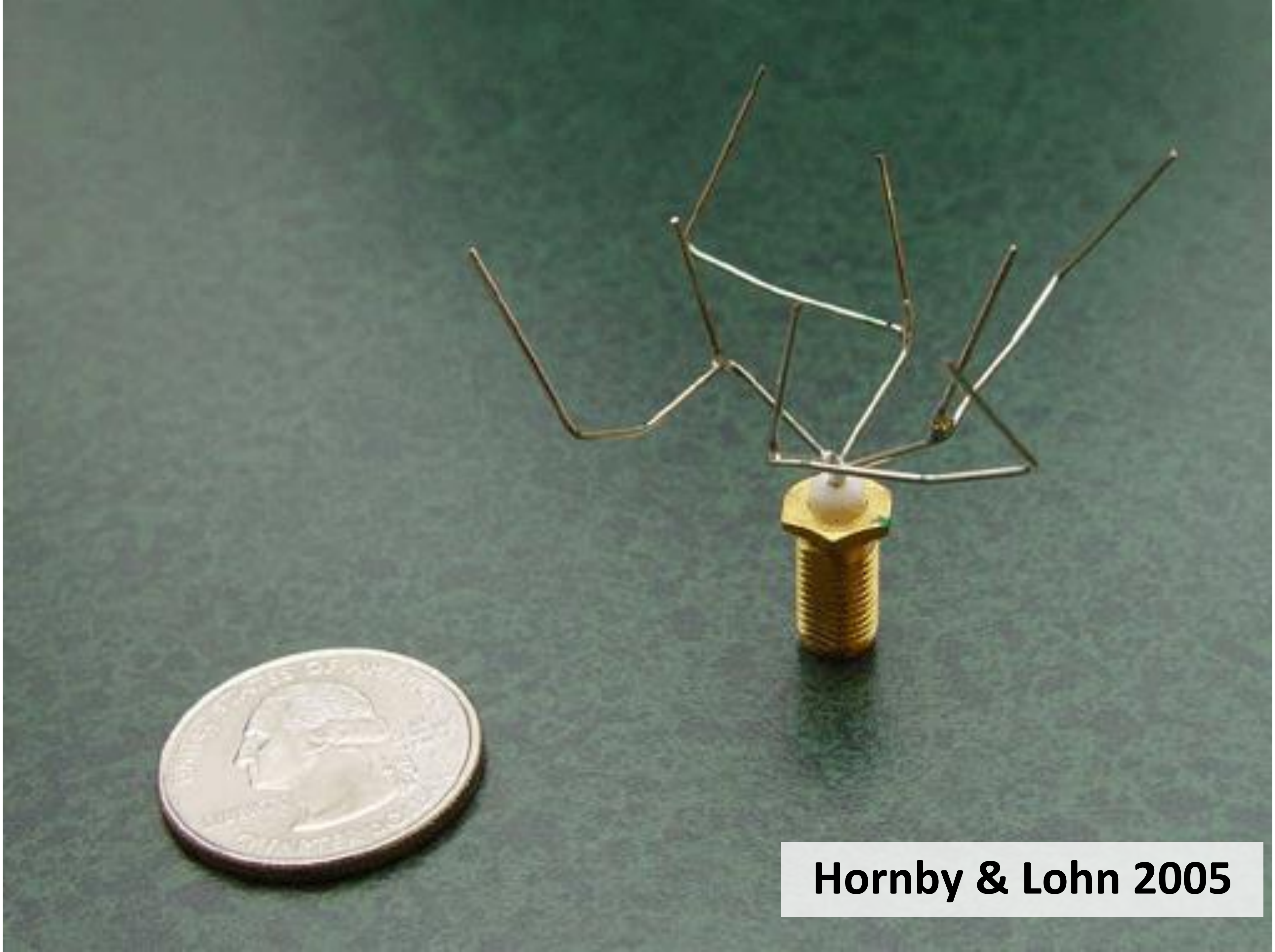
Evolution



evolution
Web
News
Images
Videos
Maps
More
Search tools
Wwe
Animals
Human
Man
WTF?
EVOLUTION OF MAN

The Cycle of Evolution





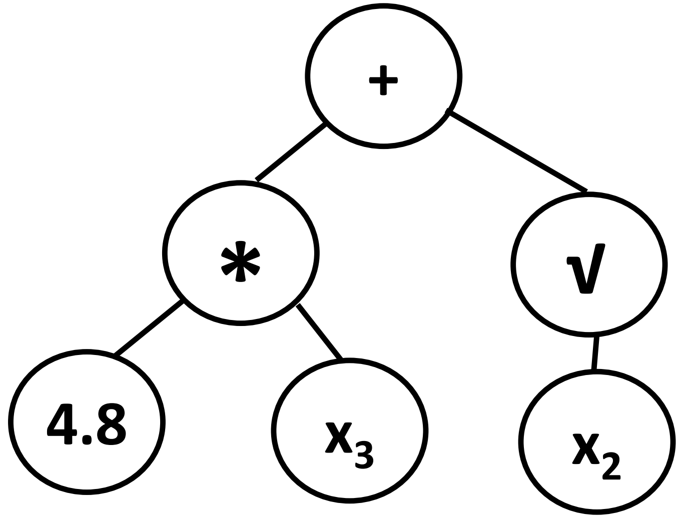
Hornby & Lohn 2005

Genetic Programming
GENETIC PROGRAMMING by JOHN KOZA
HOW CAN WE GET A COMPUTER TO KNOW WHAT TO DO
MACHINE INTELLIGENCE
HIGH RETURN
HUMAN COMPETITIVE
PATENTABLE
ROUTING
LOGICAL DISCONTINUITY
10:1 MOORES LAW JUMP
along!
That will never work!
A.I.
50's
Logic Based
Representations
SHIFT TO
FROM TOY PROBLEMS TO GP CREATING PROGRAMS AUTOMATICALLY
REPRESENTATION
FITNESS LOOPS
CREATING RANDOM PROGRAMS
SELECTION
CROSS OVER
REUSE
WORKS WITH A POPULATION (SIZE?)
SYMBOLIC REGRESSION
SECOND OFFSPRING IS SOLUTION TO THE PROBLEM
"DIVIDE + CONQUER"
PROBLEM INDEPENDENT
PROBABILISTIC
DECOMPOSE PROBLEMS INTO SUBPROBLEMS
SCALABILI
TREE REPRESENTAT OF CIRCUITS
truss design
Ant
G.P. AS A PATENT GENERATING MACHINE
EVOLUTION DRIVEN BY HIGH LEVEL SPECS + NOVELTY-DRIVEN
AMENABLE TO BEOWULF
15% of LEADING EDGE companies exploring +/or usi
—Brad Holtz
COFES

GP for SR

"A function is a *tree*"

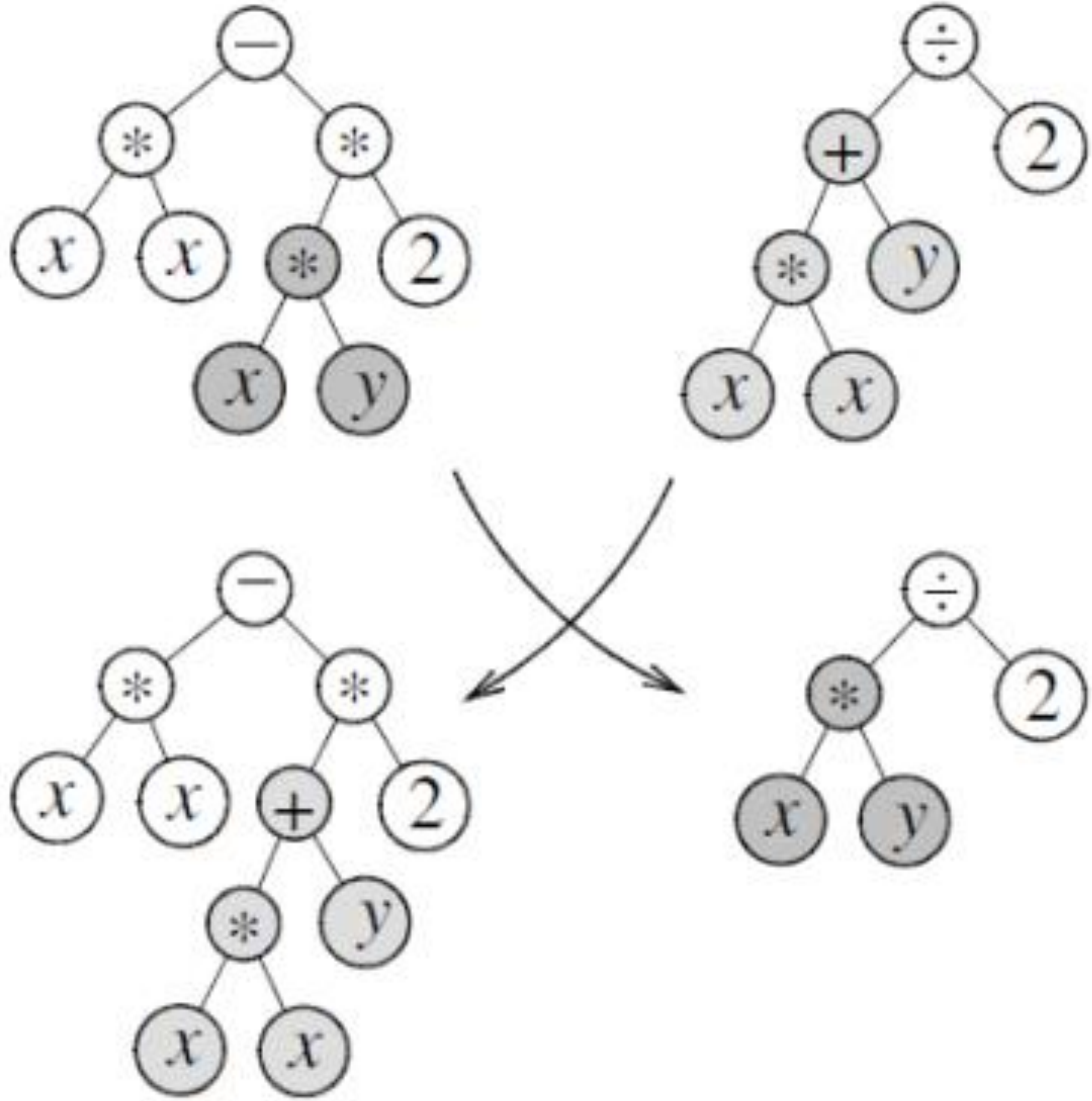
$f(x) = 4.8 * x_3 + \sqrt{x_2}$



Searches through the space of trees:

1. Initial random population; evaluate
2. Create children from parents via operators; evaluate
3. Select best; goto 2

GP for SR: Crossover Operator

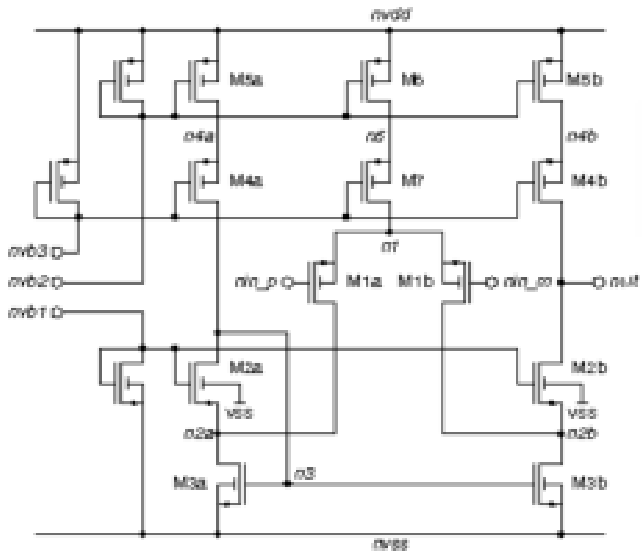


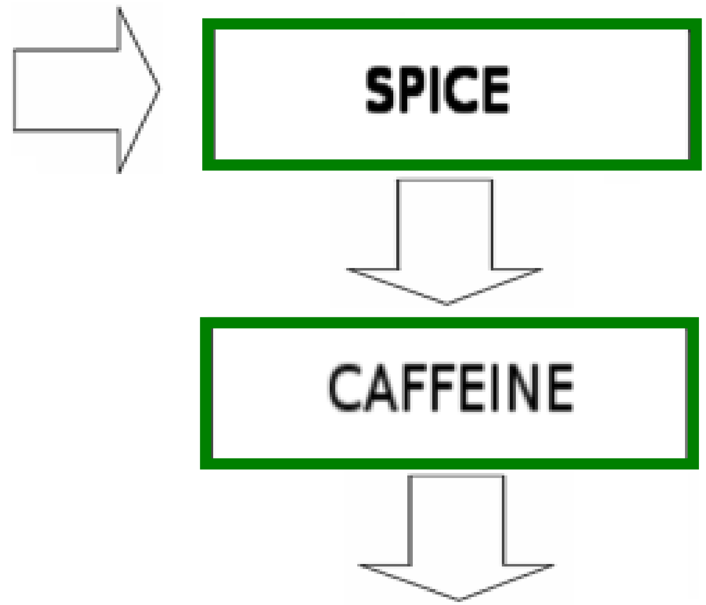
SR with Vanilla GP – Koza 1991

```
(+ (- (% (RLOG (COS X)) (* (RLOG 0.48800004)
                           (* (+ (- X X) (COS -0.8))
                              X)))
      (- (COS -0.8) (COS -0.8)))
   (* (COS (- (COS (COS (+ (RLOG X)
                           (RLOG (COS X)))))
              (RLOG X)))
      (* (COS (- (COS -0.8) (RLOG X)))
         (* (- (% (RLOG (COS X))
                  (* (RLOG 0.48800004)
                     (* (+ (- X X) (COS -0.8)) X)))
              (SIN X))
            (RLOG (COS (RLOG X))))))).
```

[Koza1992]

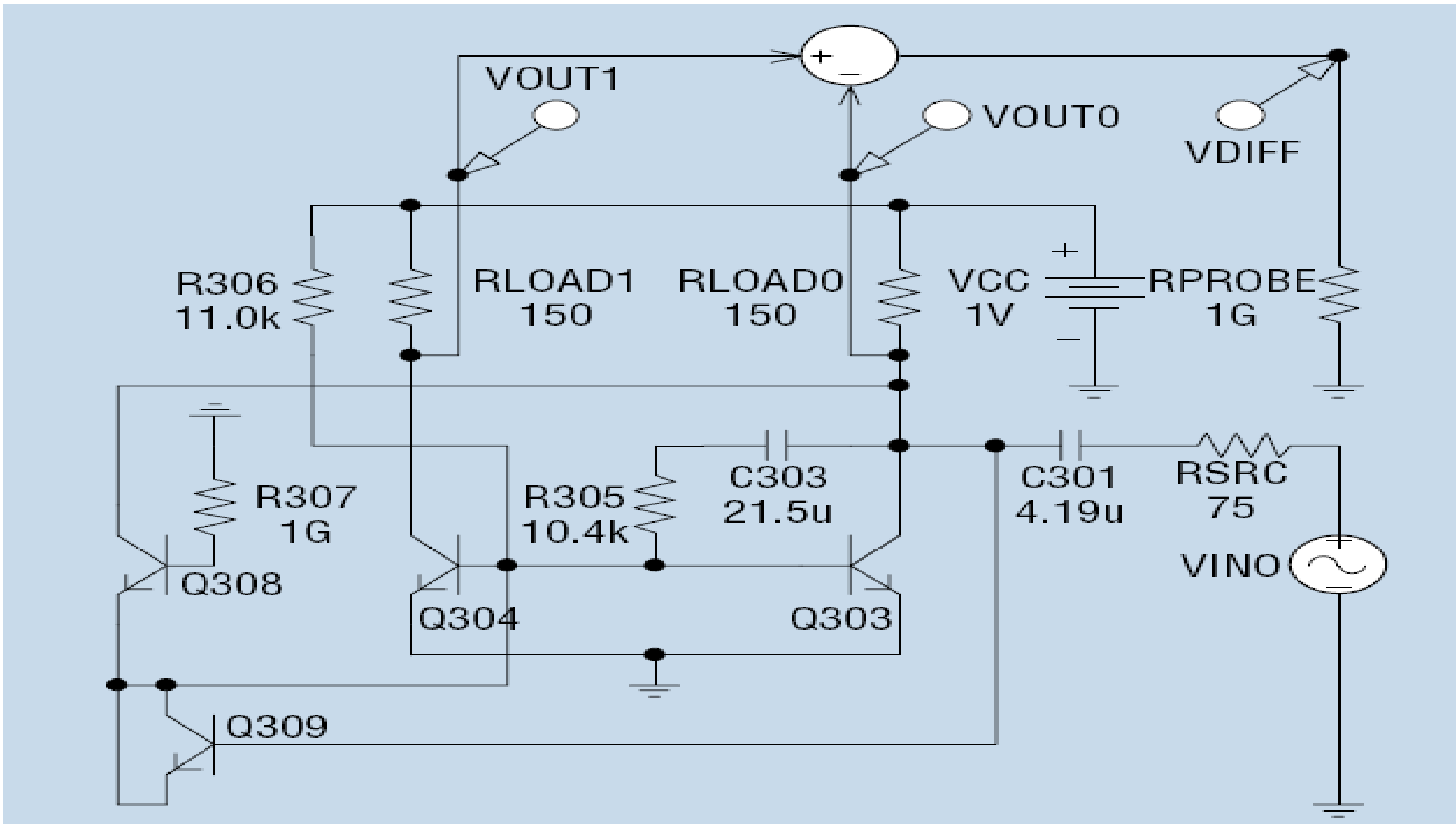
SR With Fancier GP – McConaghy 2005



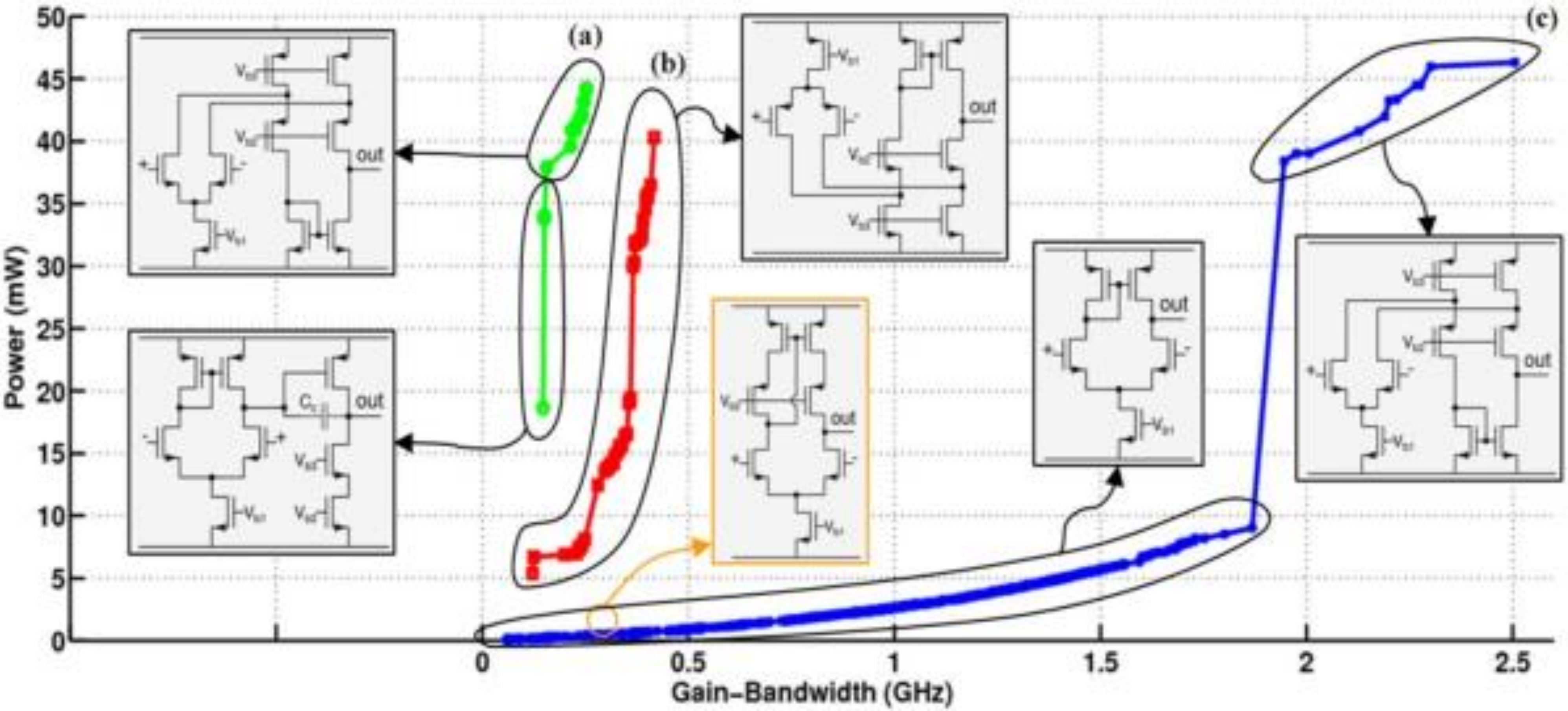


Perf.	Expression
A_{LF}	-10.3 + 7.08e-5 / id1 + 1.87 * ln(-1.95e+9 + 1.00e+10 / (vsg1*vsg3)+ 1.42e+9 *(vds2*vsd5) / (vsg1*vgs2*vsg5*id2))
f_u	10^(5.68 - 0.03 * vsg1 / vds2 - 55.43 * id1+ 5.63e-6 / id1)
PM	90.5 + 190.6 * id1 / vsg1 + 22.2 * id2 / vds2
V_{offset}	- 2.00e-3
SR_p	2.36e+7 + 1.95e+4 * id2 / id1 - 104.69 / id2 + 2.15e+9 * id2 + 4.63e+8 * id1
SR_n	- 5.72e+7 - 2.50e+11 * (id1*id2) / vgs2 + 5.53e+6 * vds2 / vgs2 + 109.72 / id1

Circuit synthesis with GP – Koza 2001



Circuit synthesis with GP – McConaghy 2005

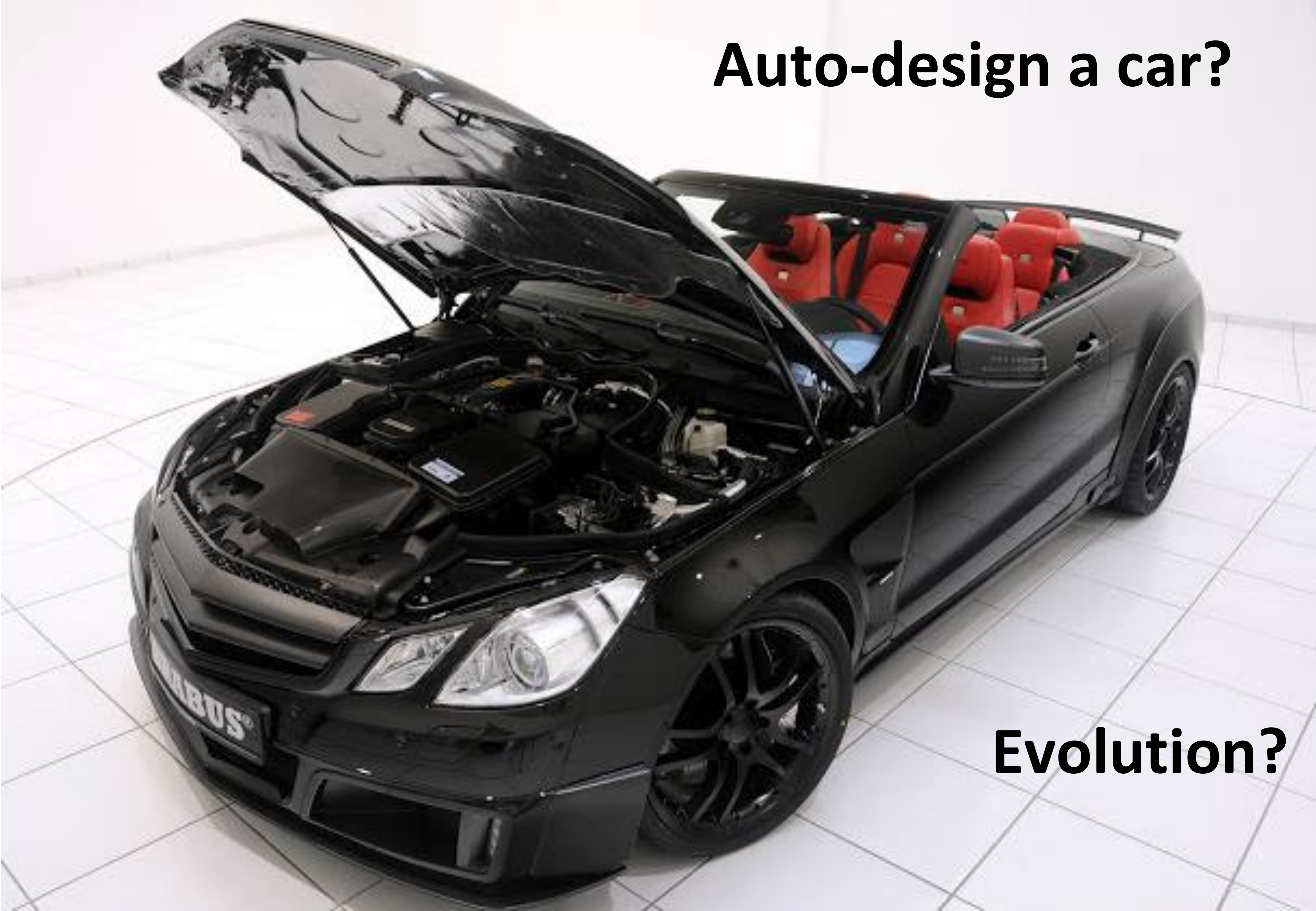


Synthesis of jewelry – Hornby 2011

Let’s Get Ambitious!
SCALE

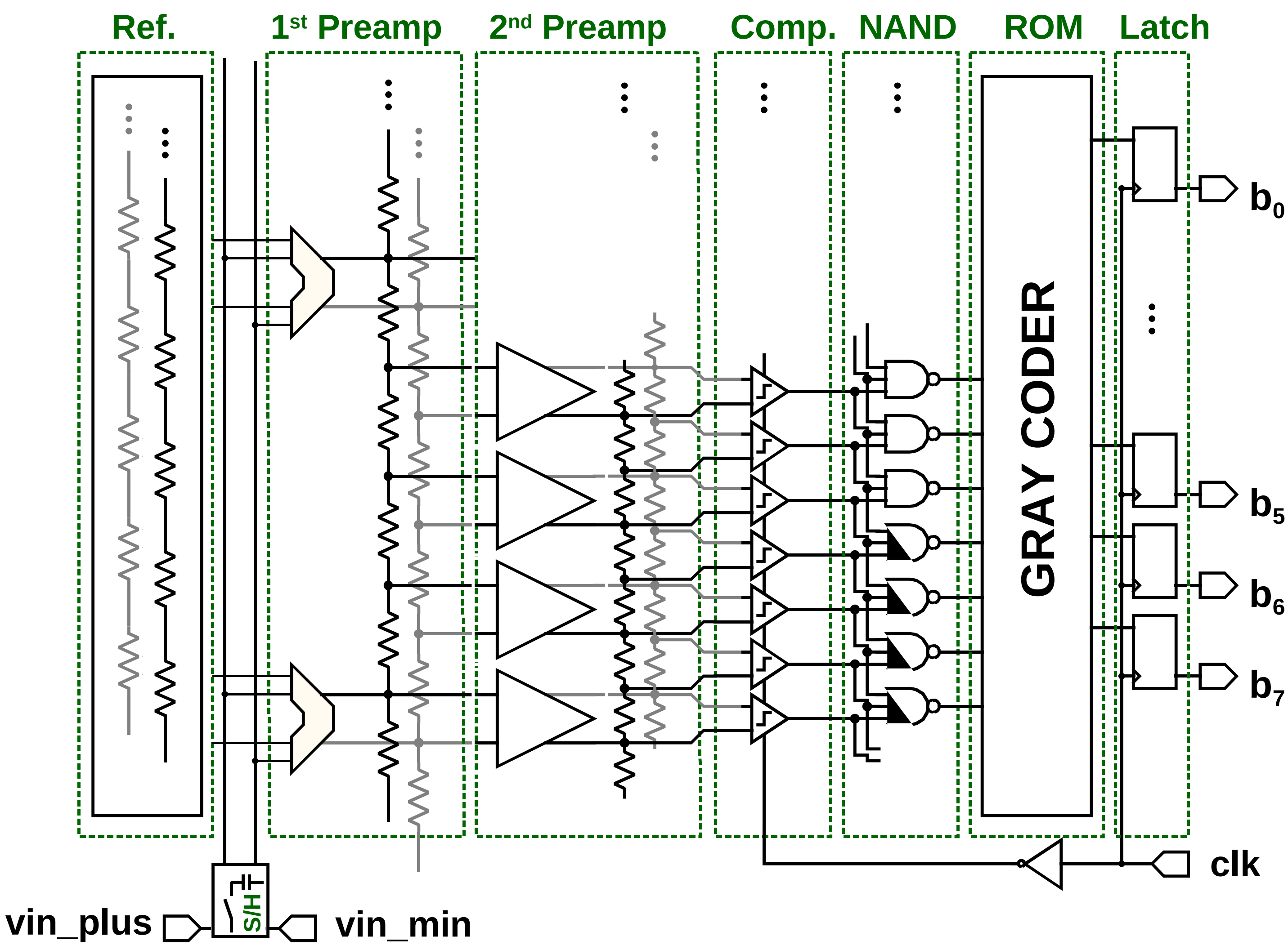
Auto-design a piston?

Evolution?


Auto-design a car?
Evolution?

How do you evolve a 5K-part car?

How do you evolve a 10G-part chip?

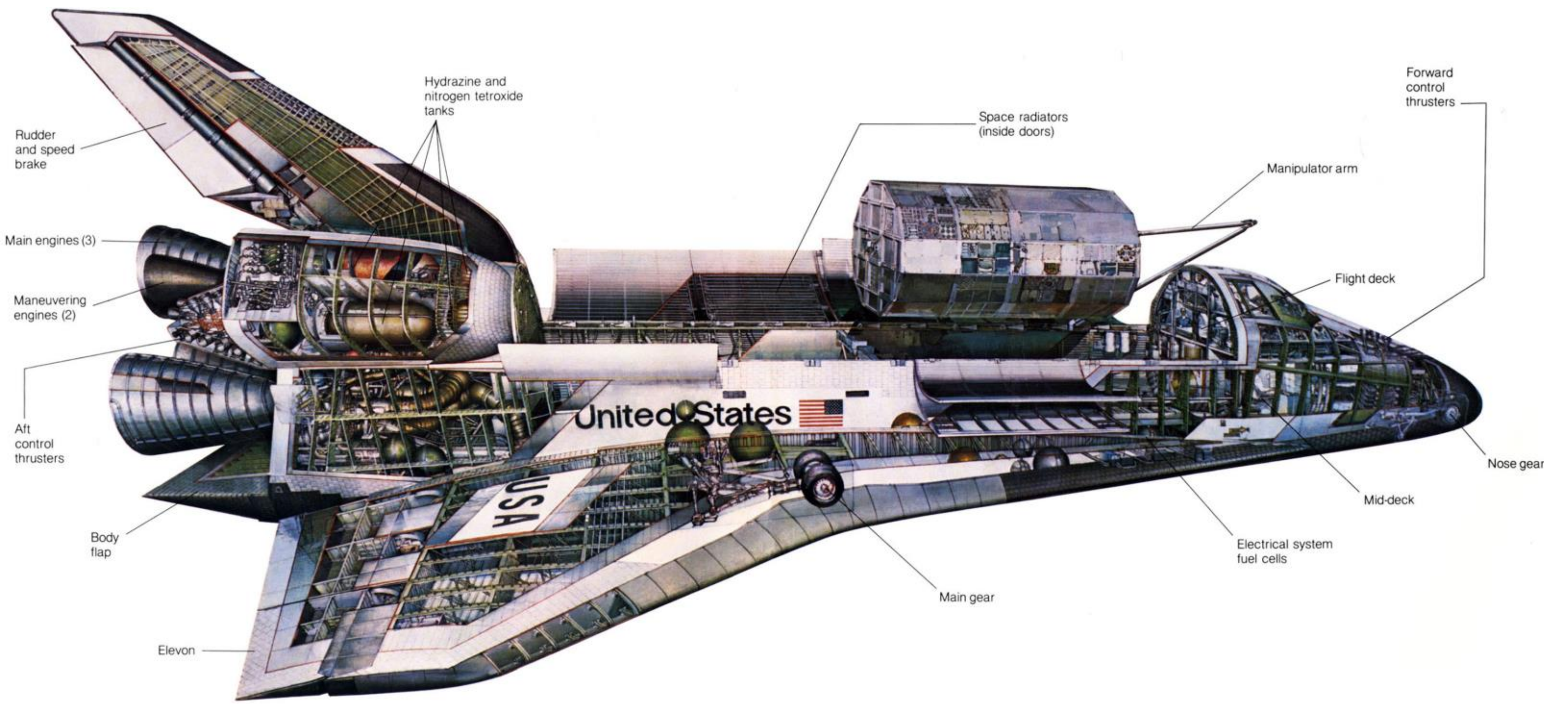


How do you evolve a 37T cell animal?

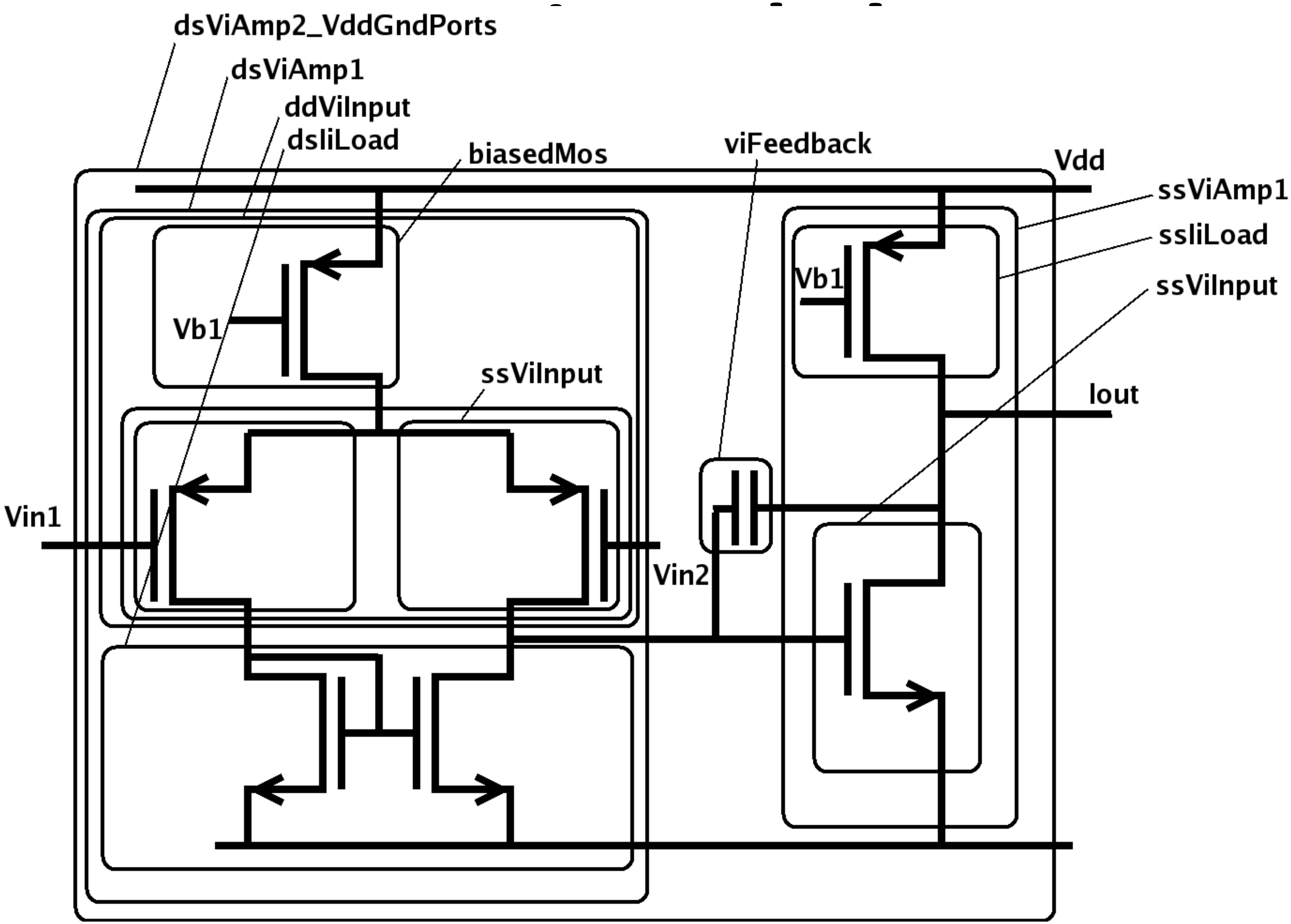
Approach the Design “Flat”?

- Simple
- What 99% of optimization does
- But, *doesn’t scale*
- Yes, massive compute helps a lot. Eg deep learning
- But even for that: what about a 10x larger net? 1000x?

Top-Down Design? How, Specifically?



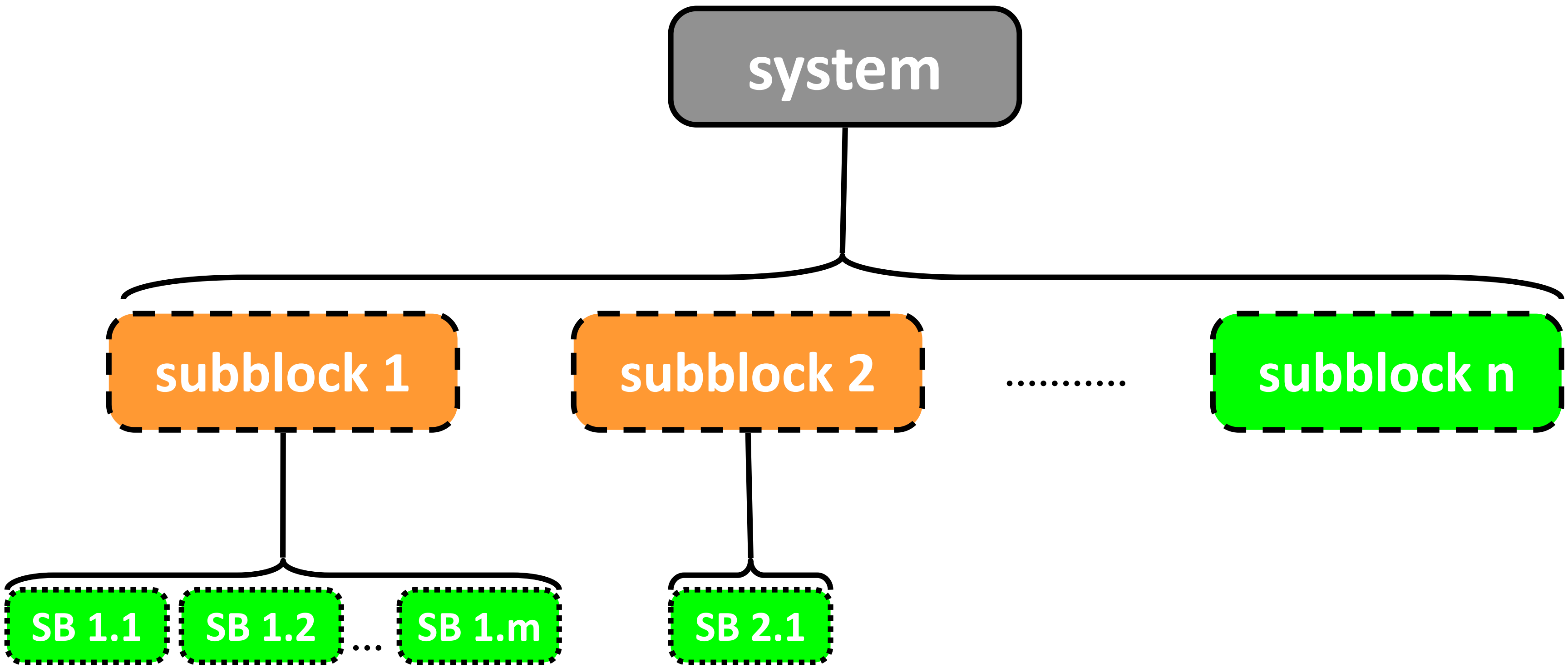
Hierarchy in a Circuit



Bottom-up Design? How, Specifically?

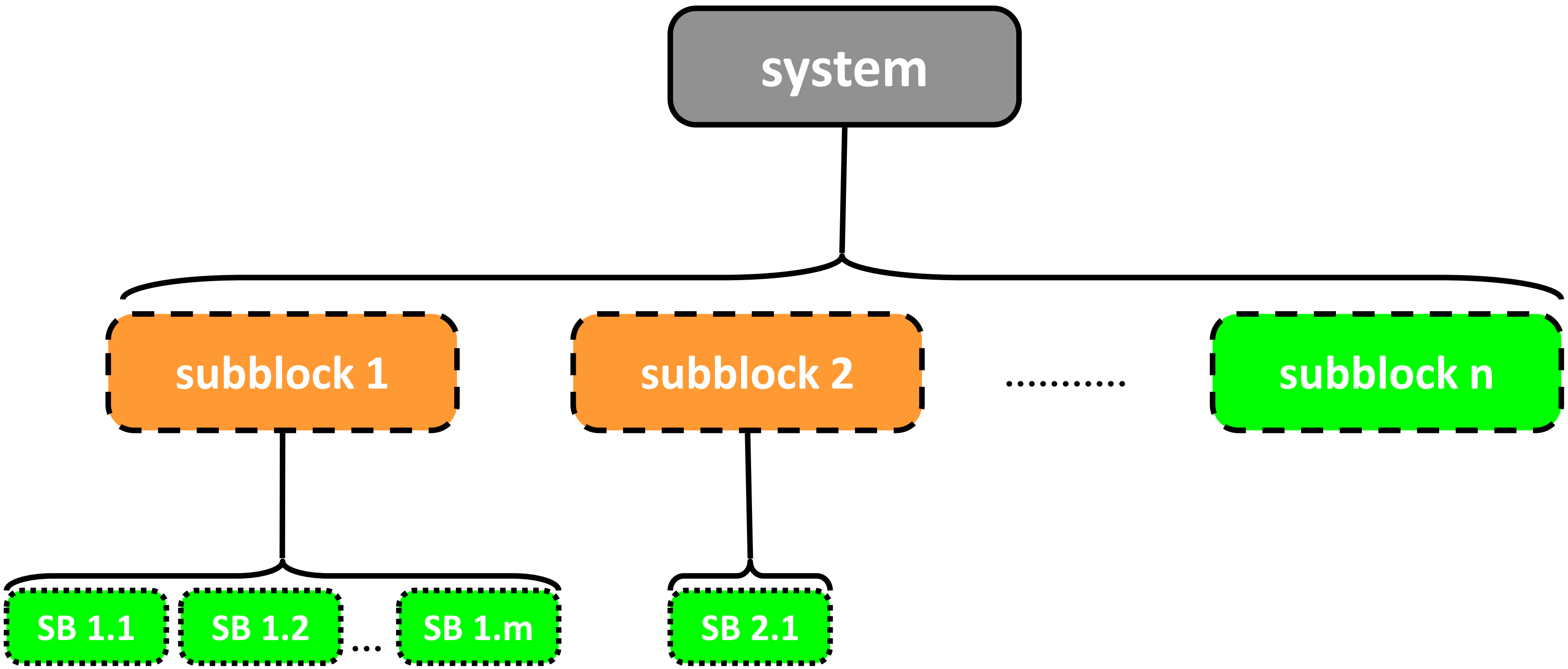
Hierarchy! Bringing Method to the Madness

- Divide, conquer, stitch it back together
- Difficulty(design) ≈ Difficulty(hardest block)



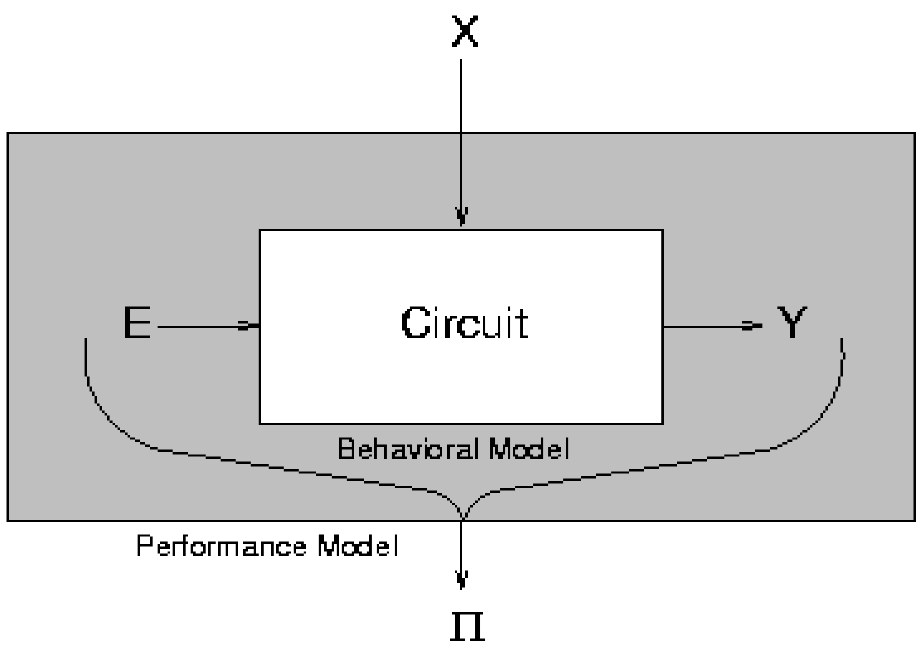
Hierarchical Design Methodologies

- Target behavior of each sub-block is well defined
- Design involves
  - A way to estimate performance of a candidate block
  - A way to search each sub-block
  - A way to stitch together the blocks
- Many possible ways!



A key requirement: we need a way to estimate performance of each sub-block

- Input: design parameters (e.g. device sizes)
- Output: performance metrics (e.g. power, …)
- Higher-level blocks are lower fidelity (to simulate faster)

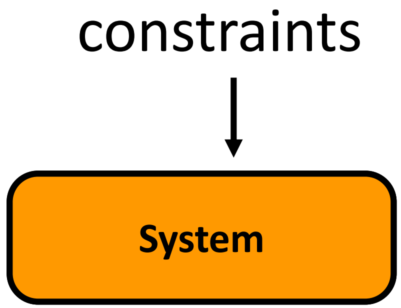


design space organization & traversal

flat

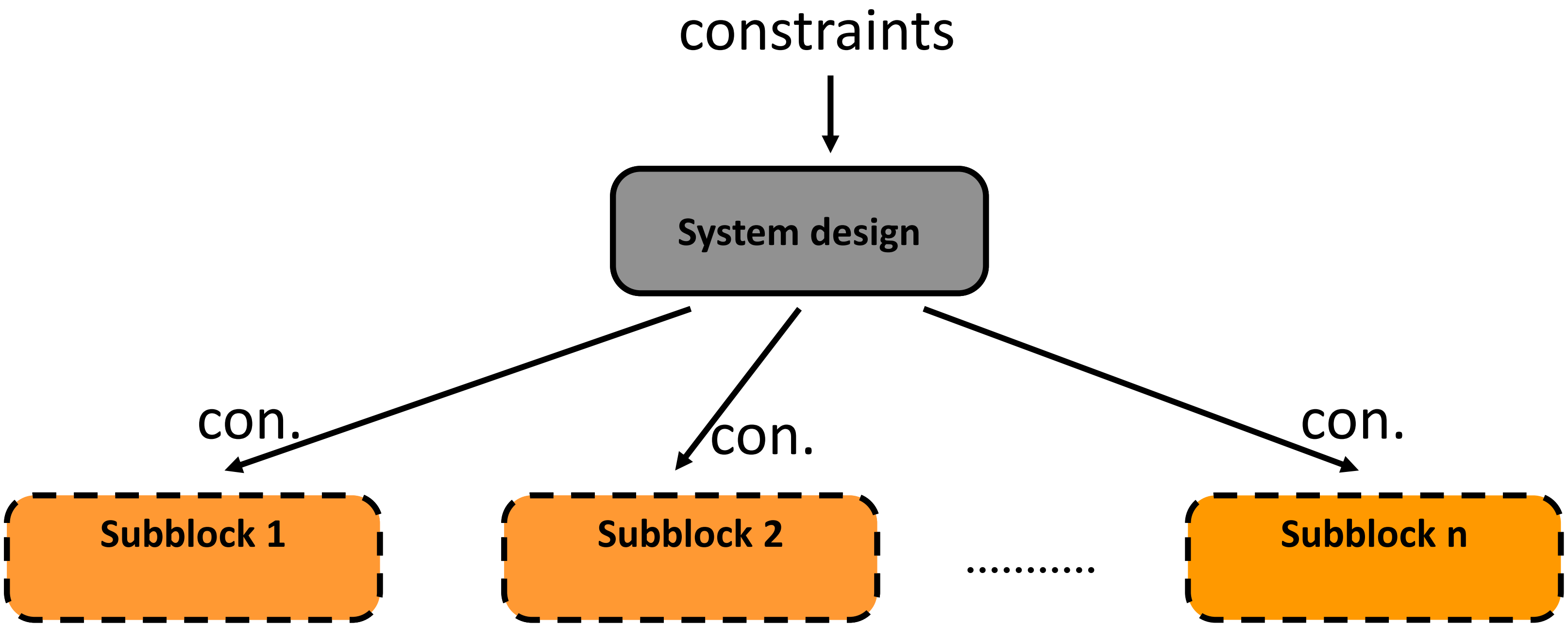
top-down
constraint-driven

Top-Down Constraint-Driven Design

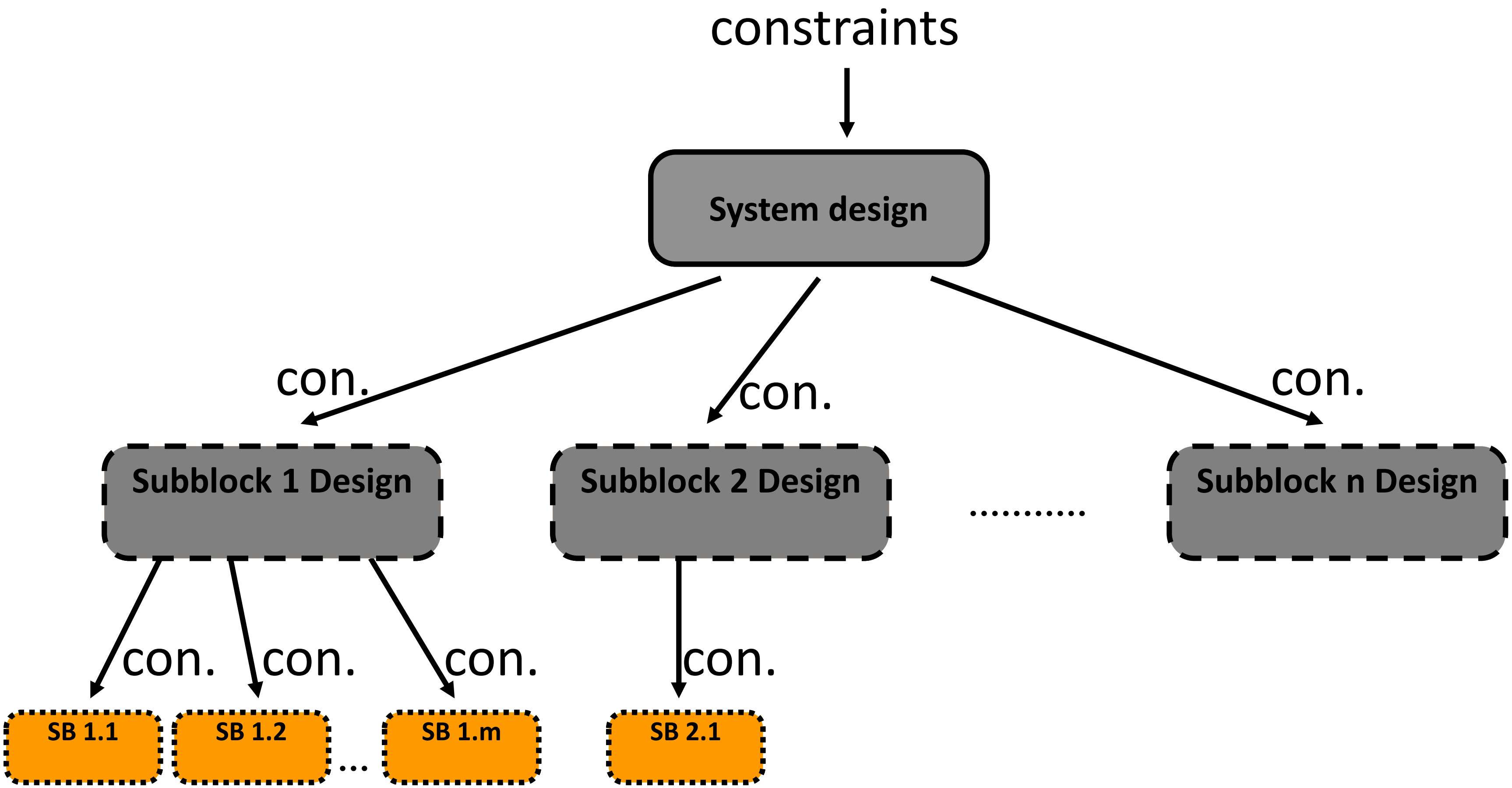


Top-Down Constraint-Driven Design

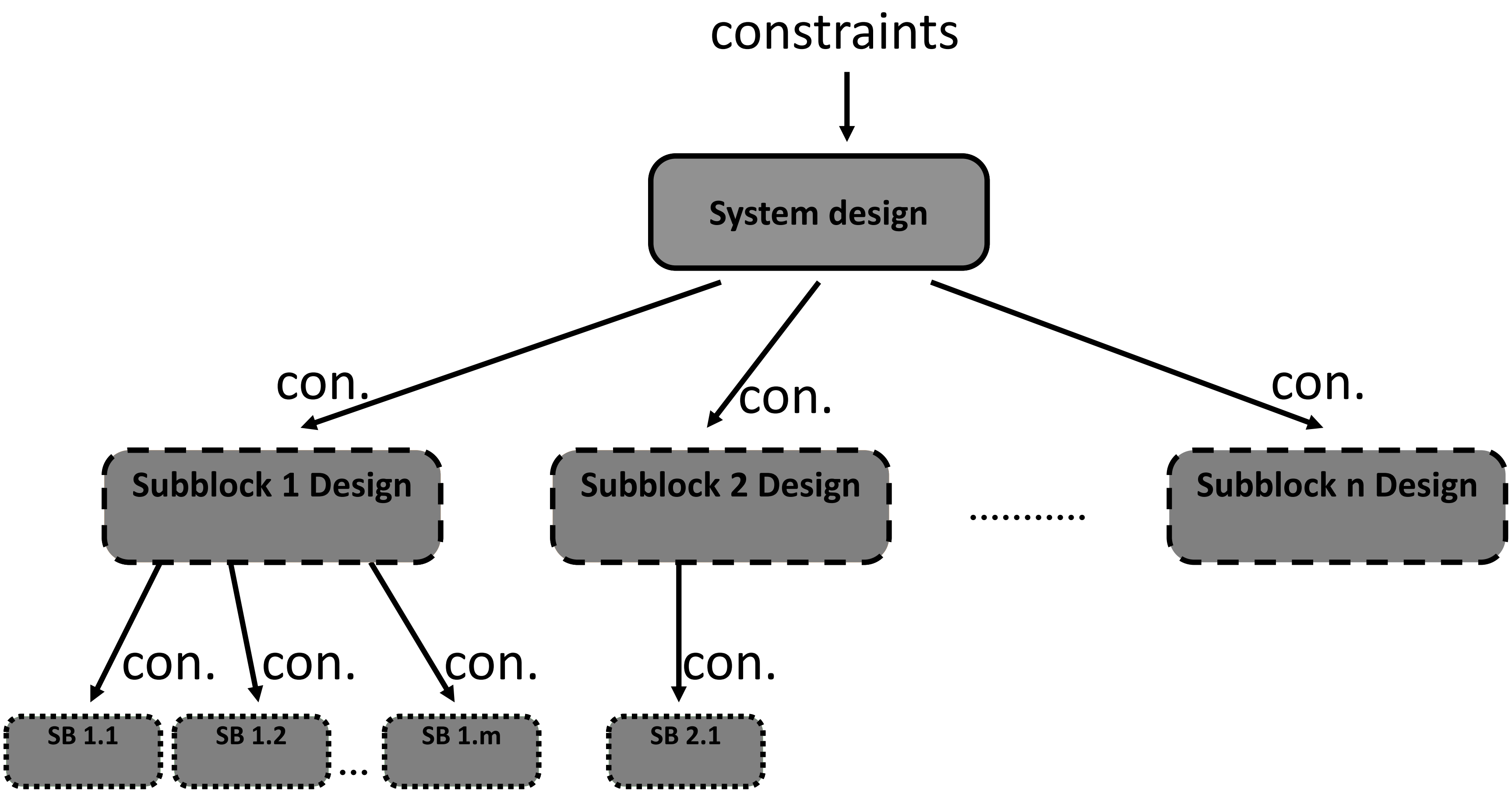
- **Step 1: map specifications down the hierarchy (via search)**



Top-Down Constraint-Driven Design

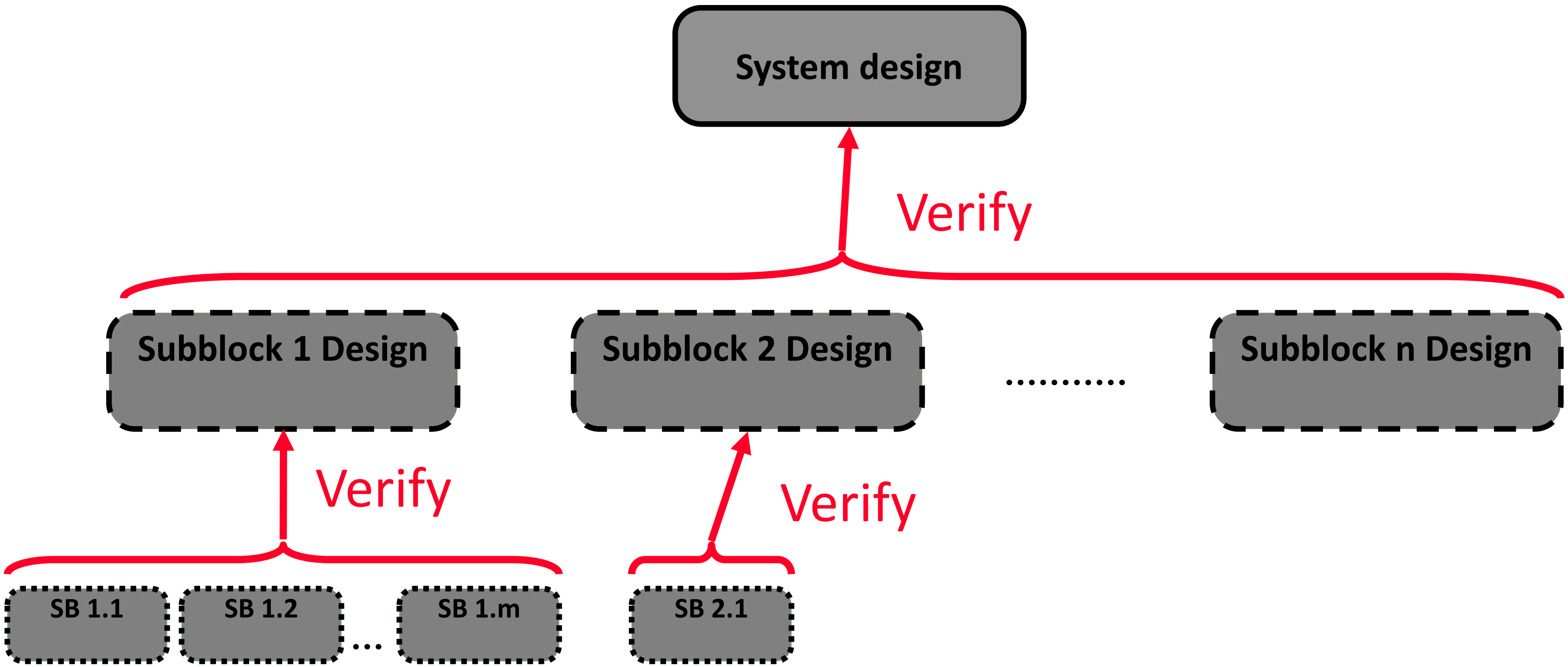


Top-Down Constraint-Driven Design



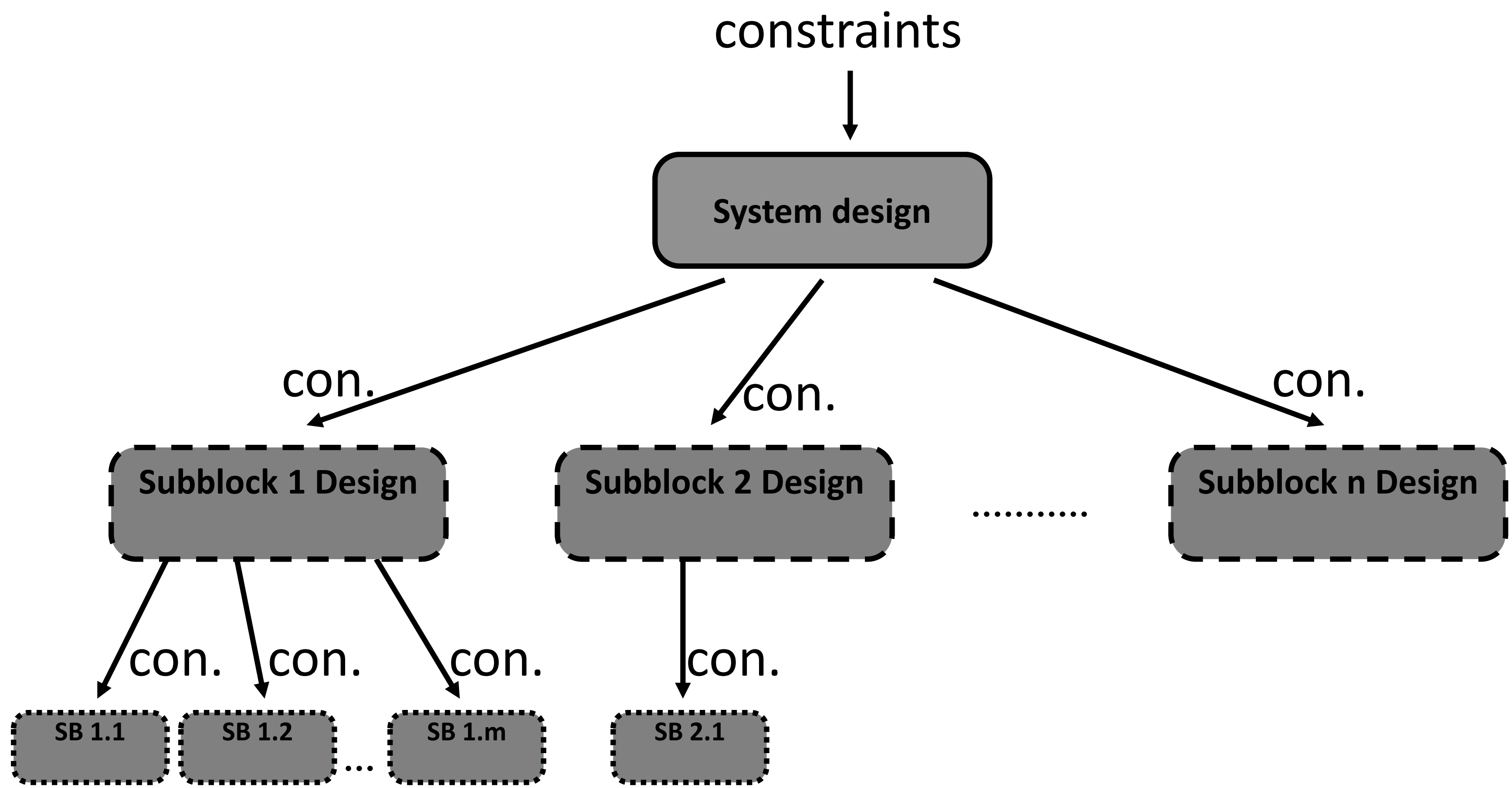
Top-Down Constraint-Driven Design

- bottom-up verification to verify performance against specifications



Top-Down Constraint-Driven Design

- What happens when constraints cannot be met?
- General heuristic: backtrack



design space organization & traversal

flat

top-down
constraint-driven

manual or
semi-auto
feasibility
modeling

design space organization & traversal

flat

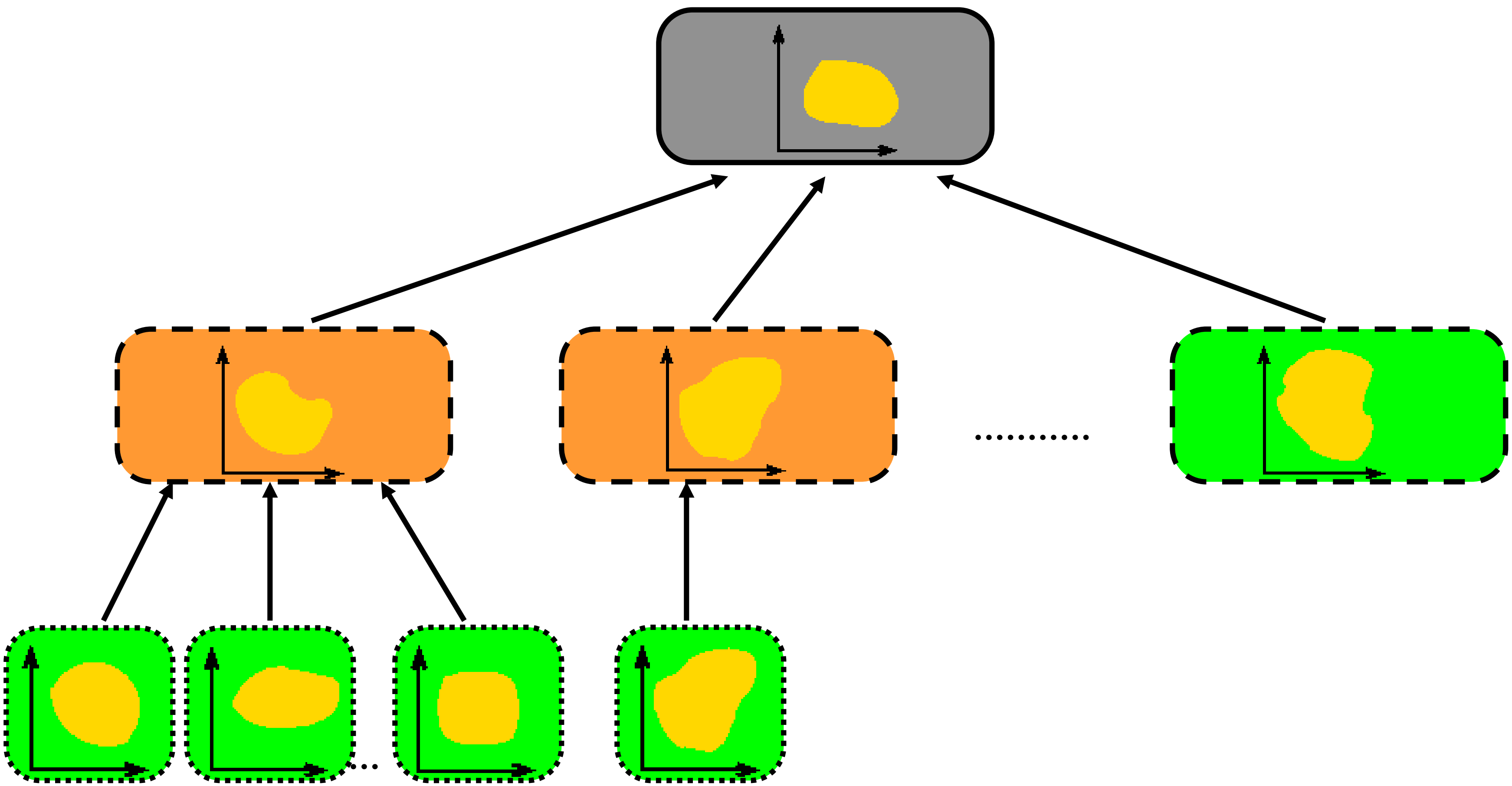
top-down
constraint-driven

manual or
semi-auto
feasibility
modeling

automatic
bottom-up
feasibility
modeling

Feasibility modeling bottom-up

- **Model *feasible performance space* of a design**
- For entire range of design parameters
- E.g. for a car: “can have 200mph top speed and 20 mpg separately but not together”. Represent this continuously.



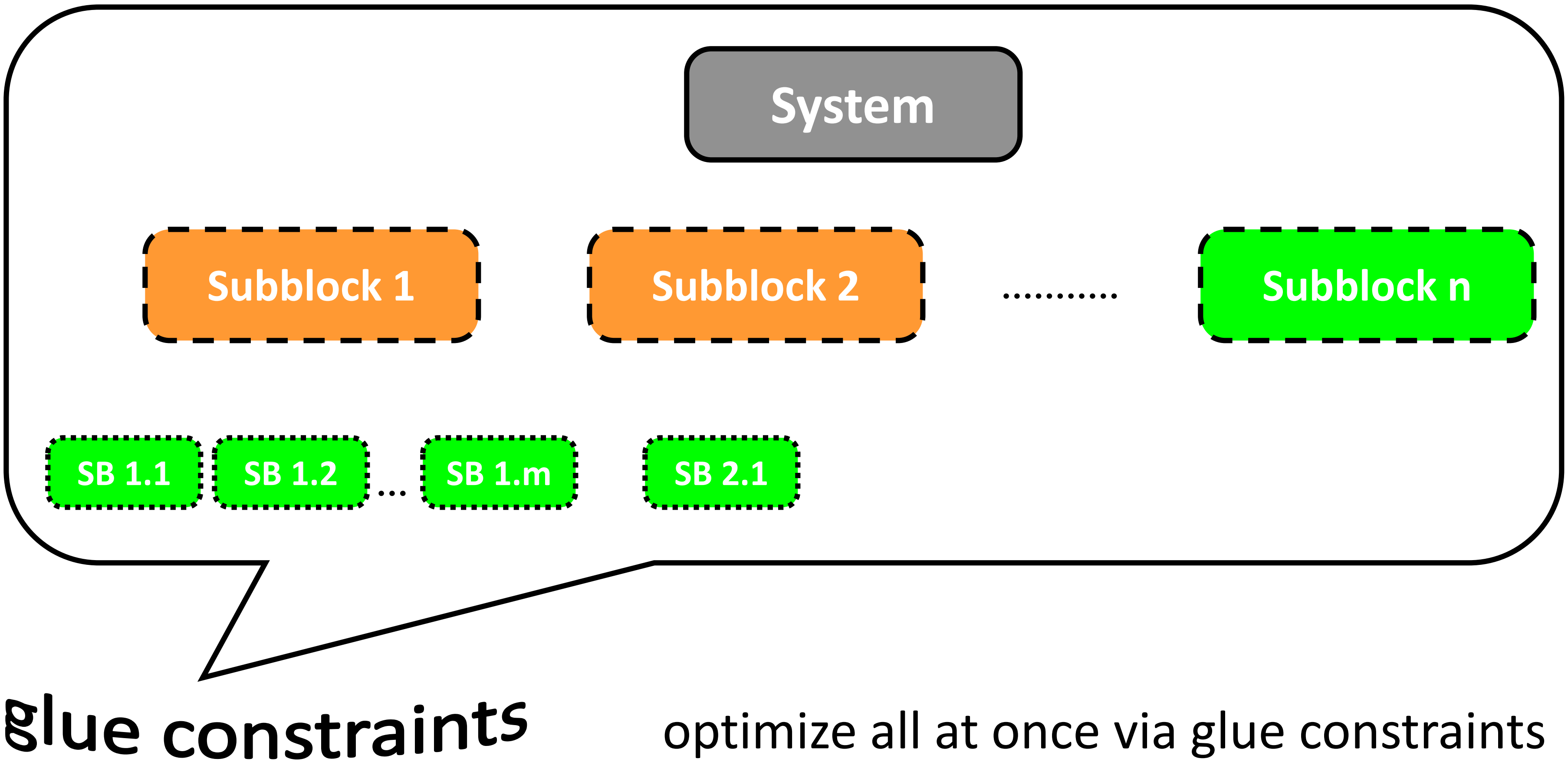
Top-down constraint-driven design

- **Pros:**
  - Scale! (Compared to flat)
  - Feasible runtime
  - Can re-use feasibility models of building blocks
  - “Natural choice” for advocates of top-down design
- **Cons:**
  - Need a way to know what performance combinations of sub-blocks are feasible
  - Once feasibility modeling is done, still need to do top-down steps (i.e. optimization at each node in hierarchy)

design space organization & traversal

- **flat**
- **top-down constraint-driven**
-
-
- **concurrent**

Concurrent hierarchical methodology



optimize all at once via glue constraints

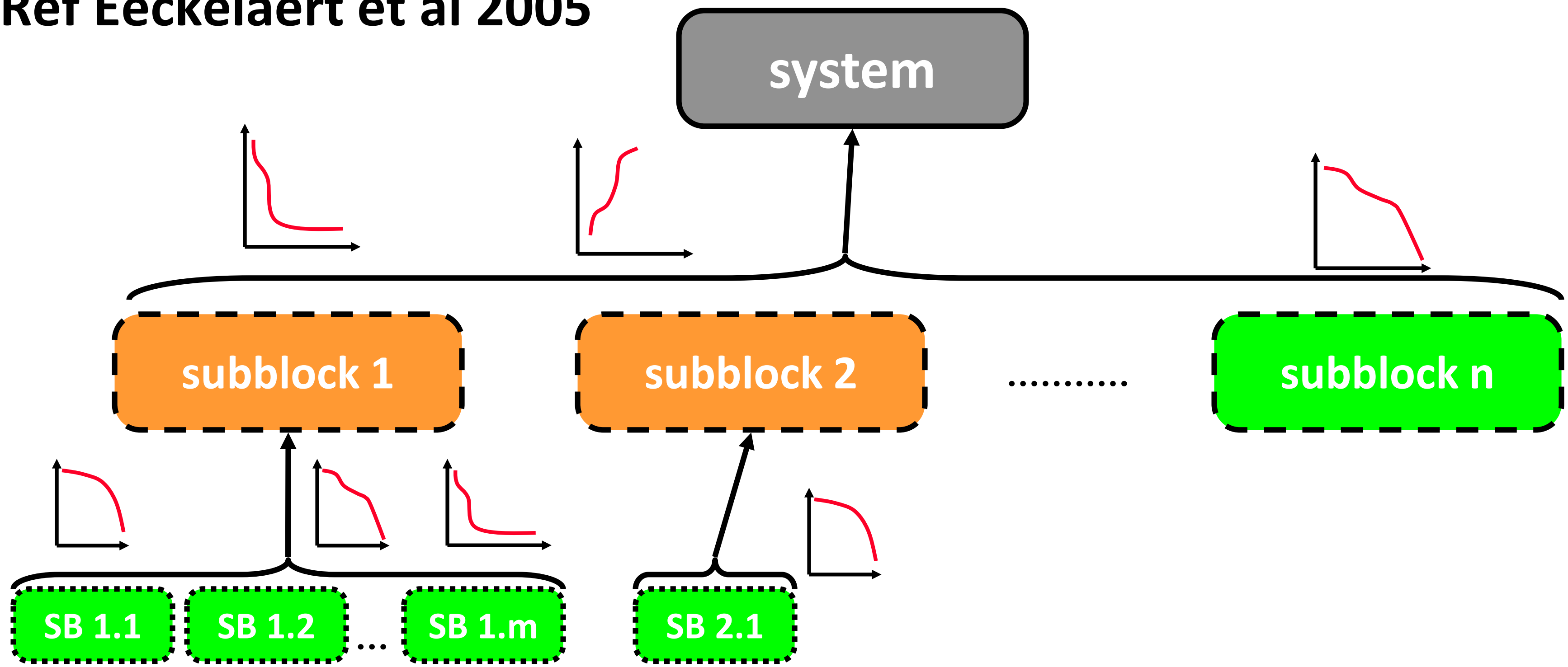
- **Pros:** an alternative to “flat”
- **Cons:** scales badly; more complex than “flat”

design space organization & traversal

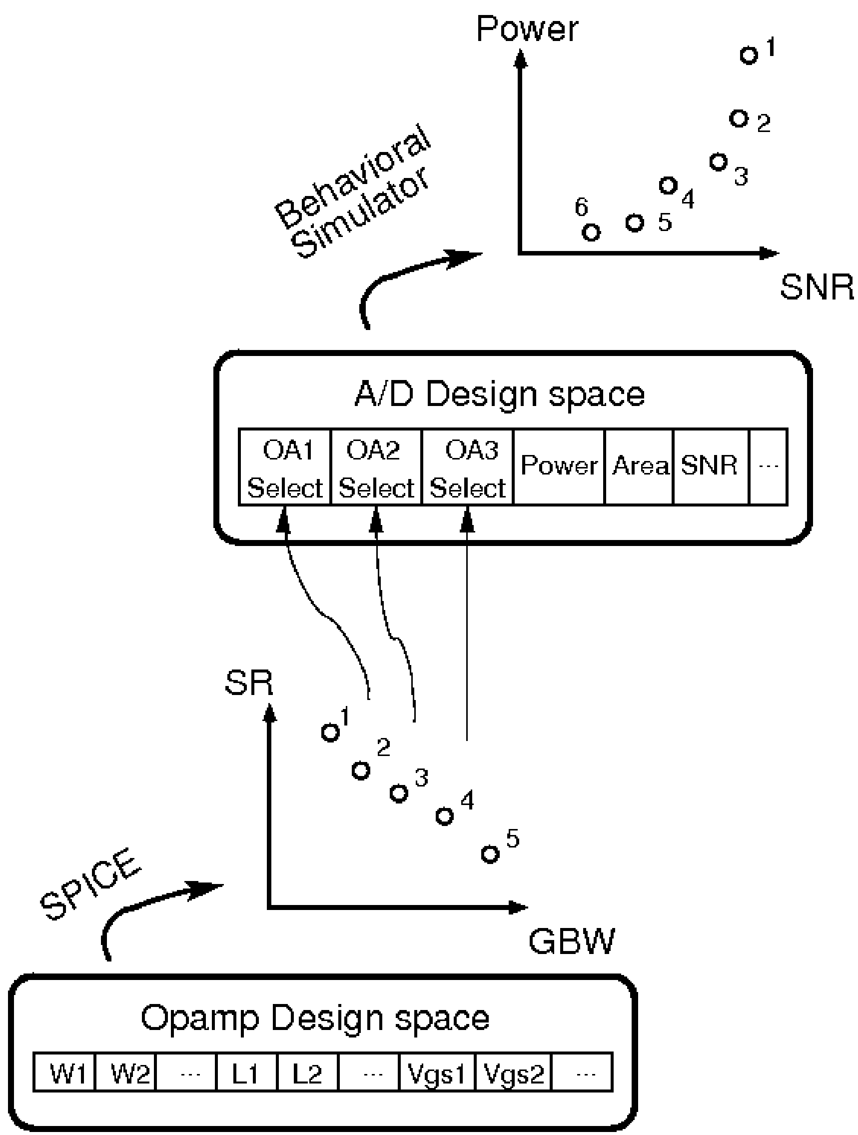
- flat
- top-down constraint-driven
- bottom-up multi-objective
- concurrent

Multi-objective bottom-up design (MOBU)

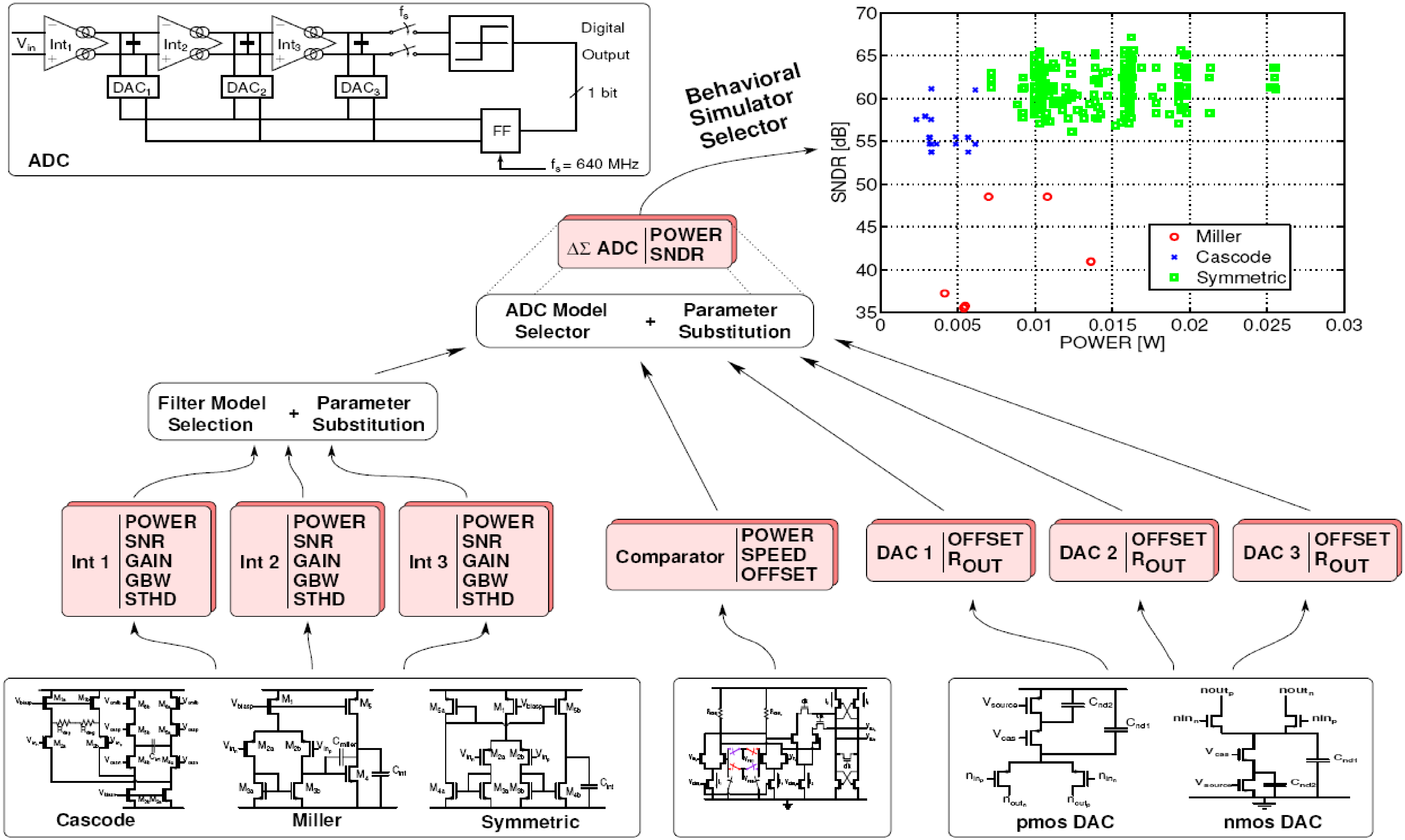
- Pareto front of a lower level block is “dimensions” in the design space of each layer that uses it
- Effectively “compresses” lower level design spaces into the meaningful part (i.e. into the tradeoffs)
- Ref Eeckelaert et al 2005



MOBU Example #1



MOBU Example #2



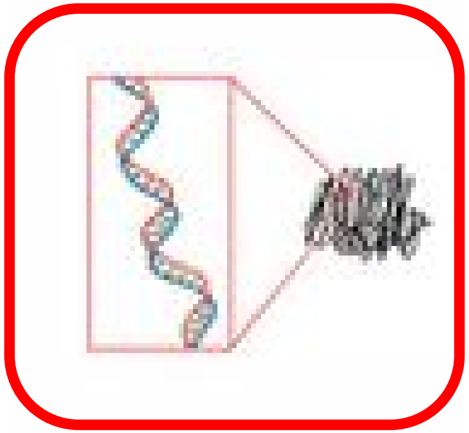
Multi-objective bottom-up design (MOBU)

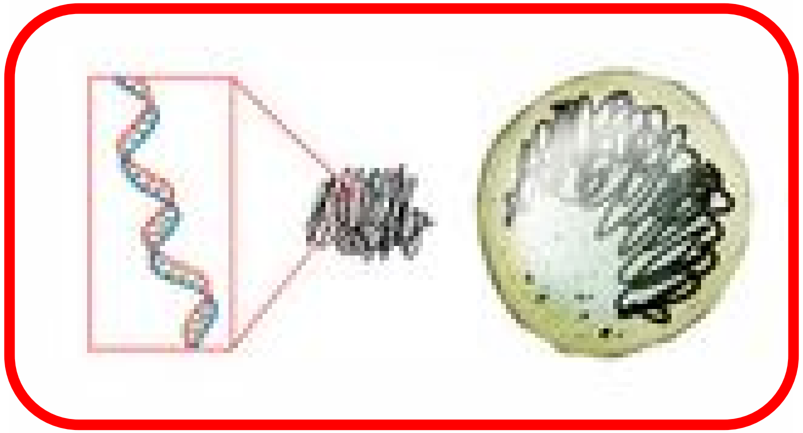
- **Pros:**
  - Feasible runtime
  - Provides system level tradeoffs
  - Can re-use multi-objective tradeoffs of building blocks
  - “Model” of tradeoffs is simply the discrete designs themselves
  - Once once top-level tradeoffs are known, the designs already exist too. No need for a subsequent top-down sizing
- **Cons:**
  - No full model of feasibility, just of tradeoffs
  - Cannot refine designs precisely, the way that top-down does

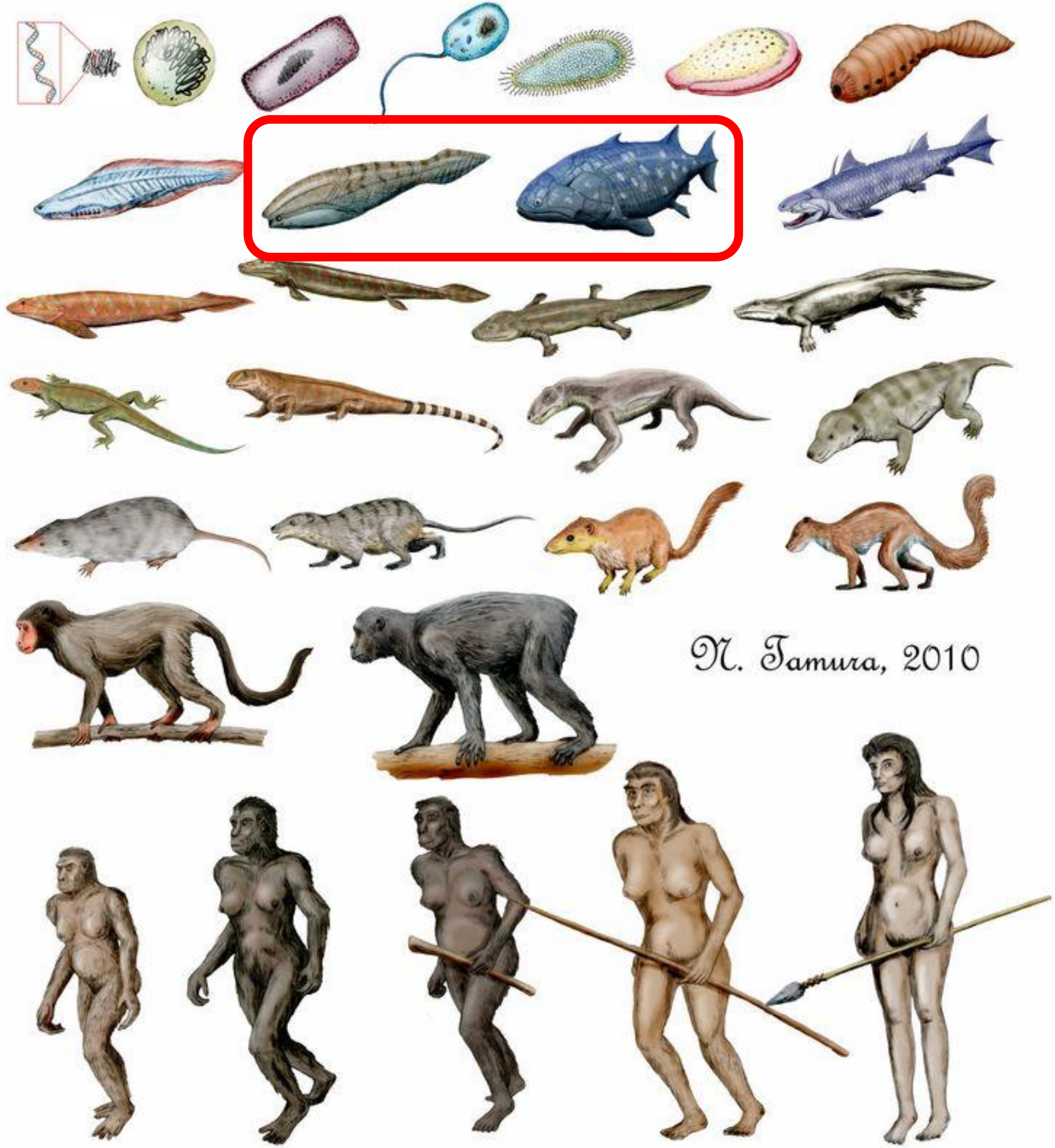
How do you evolve a 37T cell *animal*?

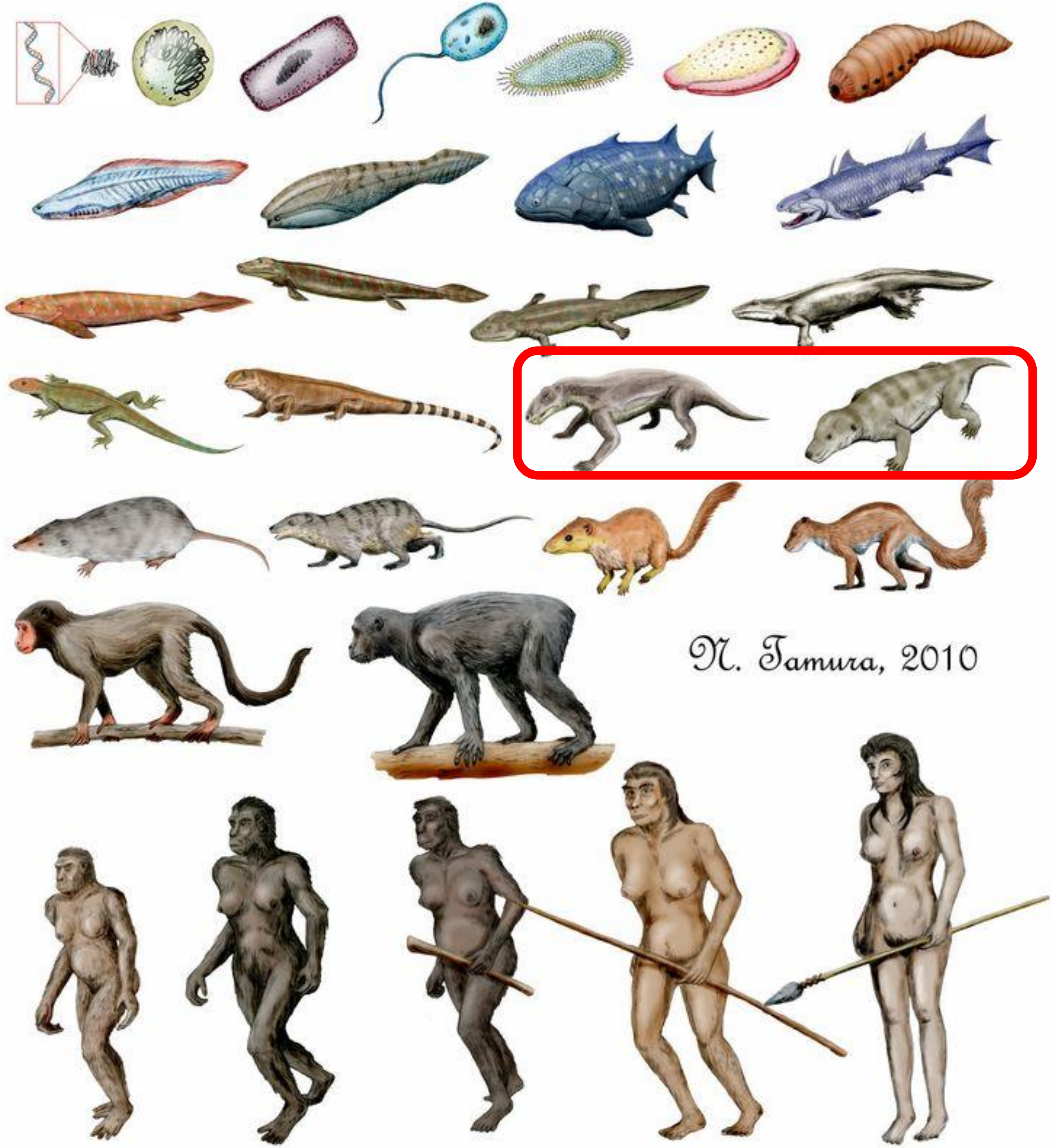
design space organization & traversal

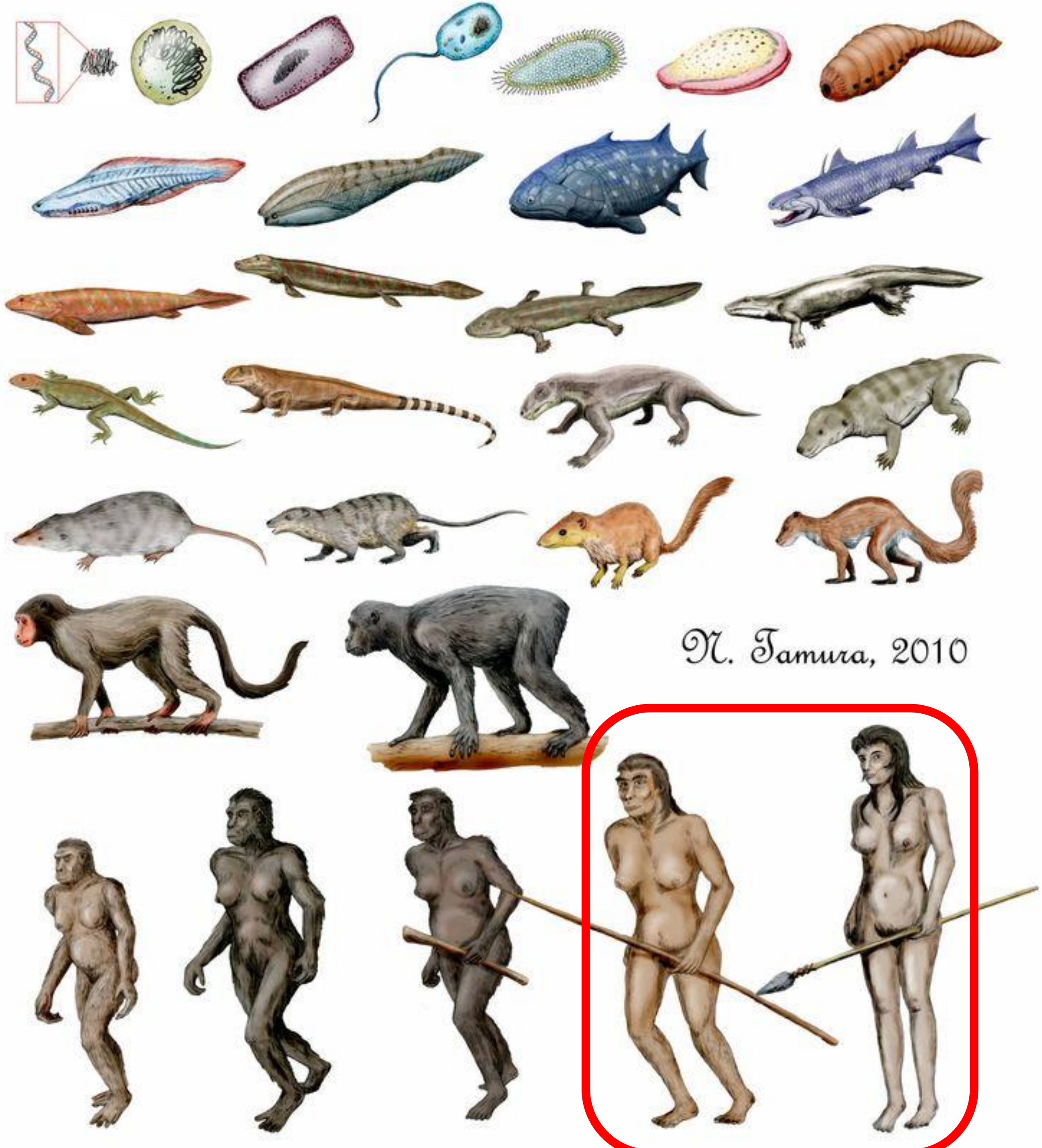
- flat
- top-down constraint-driven
- bottom-up ossification
- bottom-up multi-objective
- concurrent





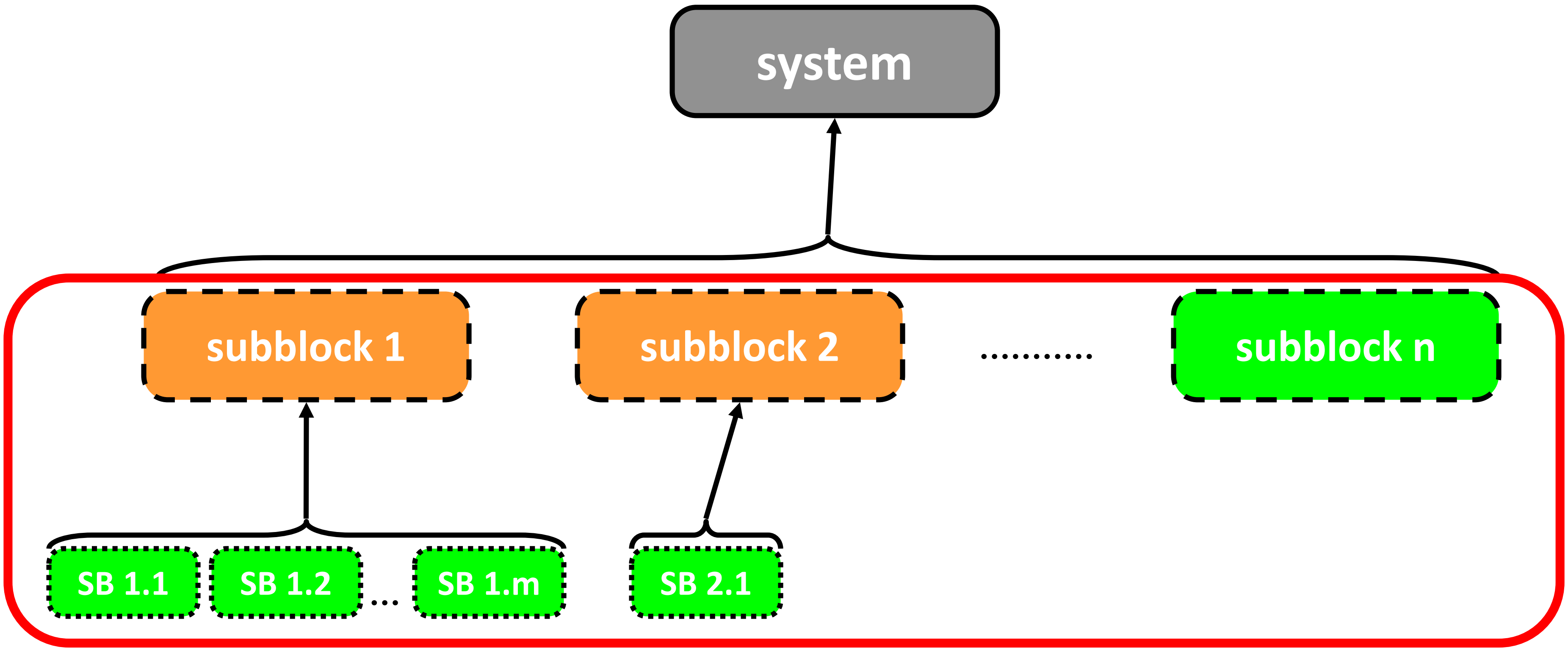

N. Tamura, 2010


N. Tamura, 2010


N. Tamura, 2010

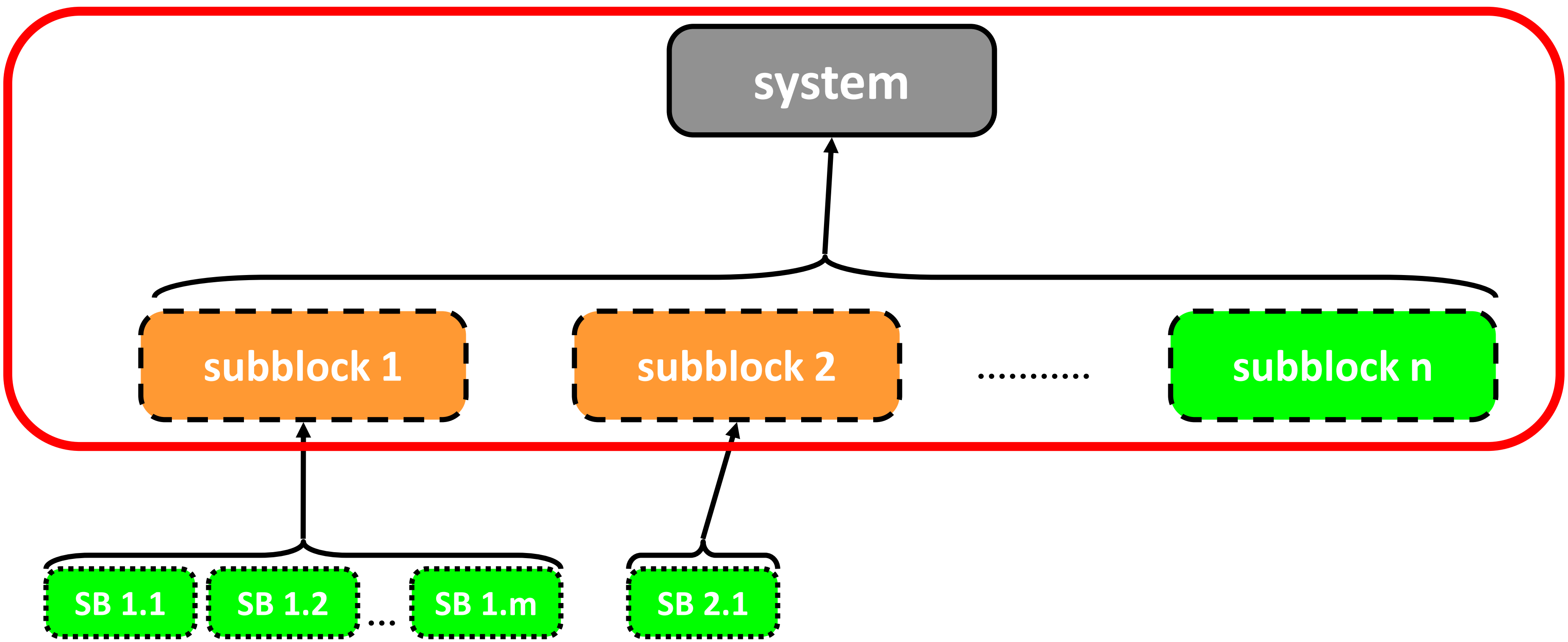
Bottom-up Ossification

- Two levels at a time are evolving
- Then lower level's mutation rate →0 (ossifies), and a new upper level emerges



Bottom-up Ossification

- Two levels at a time are evolving
- Then lower level's mutation rate →0 (ossifies), and a new upper level emerges



Conclusion

- Hierarchical design methodologies enable ridiculously complicated designs
  - In machine learning
  - And in nature!
- They are *structured*, not ad-hoc
  - Top-down constraint-driven
  - Multi-objective bottom up
  - And more
- Are you ready for 1000x larger nets?

Trent McConaghy
@trentmc0

Survey / further reading

- Georges Gielen, Trent McConaghy, and Tom Eeckelaert, “Performance Space Modeling for Hierarchical Synthesis of Analog Integrated Circuits”, Proc. Design Automation Conference, 2005. http://trent.st/content/2005_DAC_hierarchy.pdf
- For further references, see references of that work