

Legal Notices

- McMaster University and the authors make no claims of fitness and accept no liability for the statements in this talk
- Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc., in the United States, other countries, or both.
- IBM is a registered trademark of International Business Machines Corporation in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others.

Coconut: COde CONstructing User Tool

Christopher Kumar Anand



Software Quality Research Laboratory

McMaster University



<http://ocalgorithms.com>



Advanced Optimization Laboratory
McMaster University

Thanks

Wolfram Kahl (McMaster)

Robert Enenkel (IBM)

Stephen Adams

Kevin Browne

Ding Cong

Shiqi Cao

Nathan Cumpson

Andrew Curtis

Saeed Jahed

Damith Karunaratne

Clayton Goes

Gabriel Grant

William Hua

Fletcher Johnson

Wei Li

Rakshit Kumar

Nick Mansfield

Mehrdad Mozafari

Paul Polak

Adam Schulz

Anuroop Sharma

Sanvesh Srivastava

Wolfgang Thaller

Gordon Uszkay

Christopher Venantius

Paul Vrbik

IBM Centre for Advanced Studies, CFI, OIT, NSERC, Apple Canada.

Coconut

1. experiment

- DSLs embedded in Haskell
- principled graph transformations

2. production compiler/code generator

- produces IBM MASS library on several platforms

Ideas

- orthogonal aspects into DSLs
 - greater control
 - one aspect at a time
- principled graph transformations
 - *still working on embedding into a textual language*

Roadmap

- SIMD Parallelism

-  extensible DSL captures patterns

-  verification via graph transformation

-  generated library shipping (Cell BE SDK 3.0)

- Multi-Core Parallelism

-  model on ILP

-  generation via graph transformation

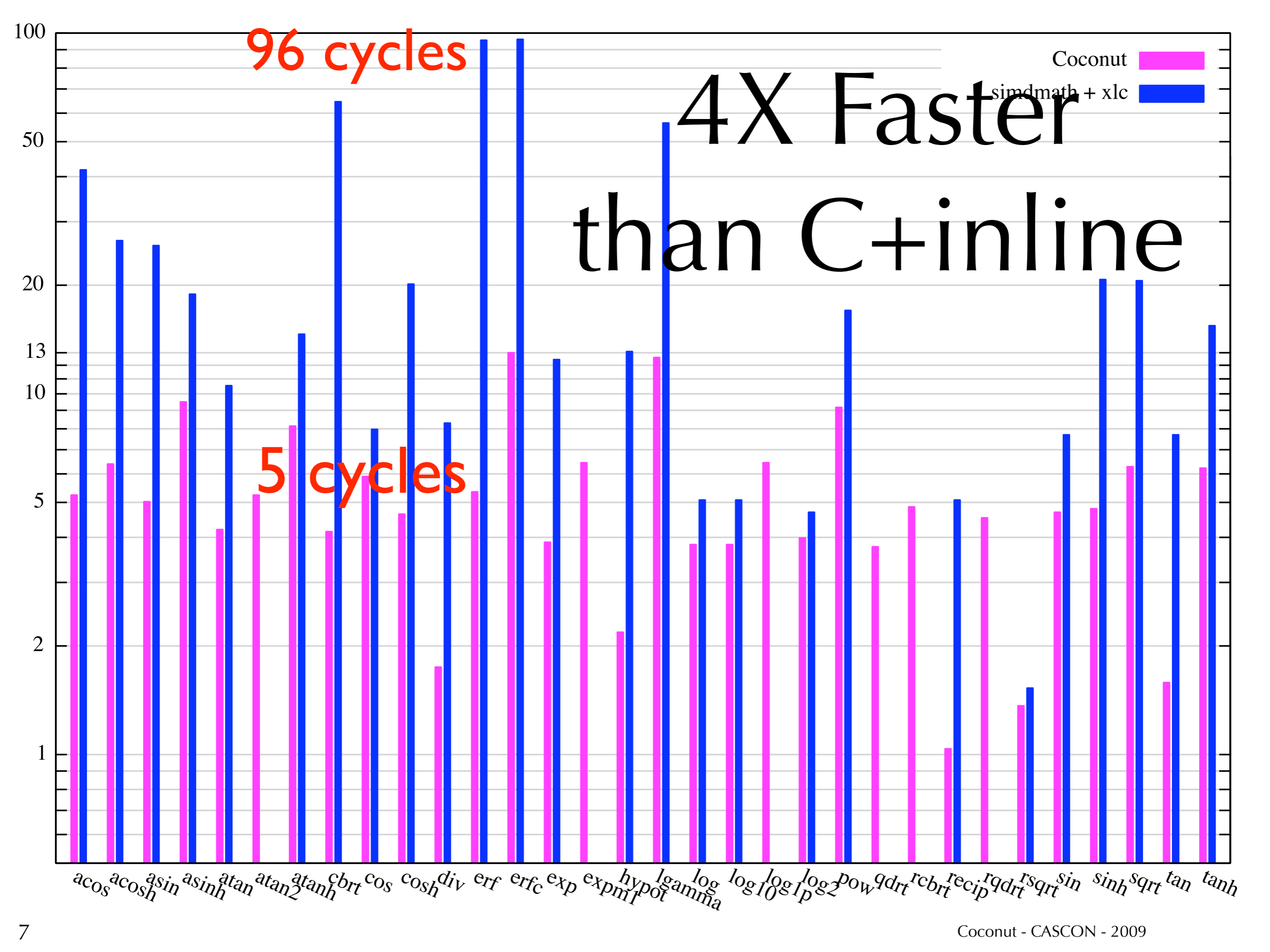
-  linear-time verification

-  run time

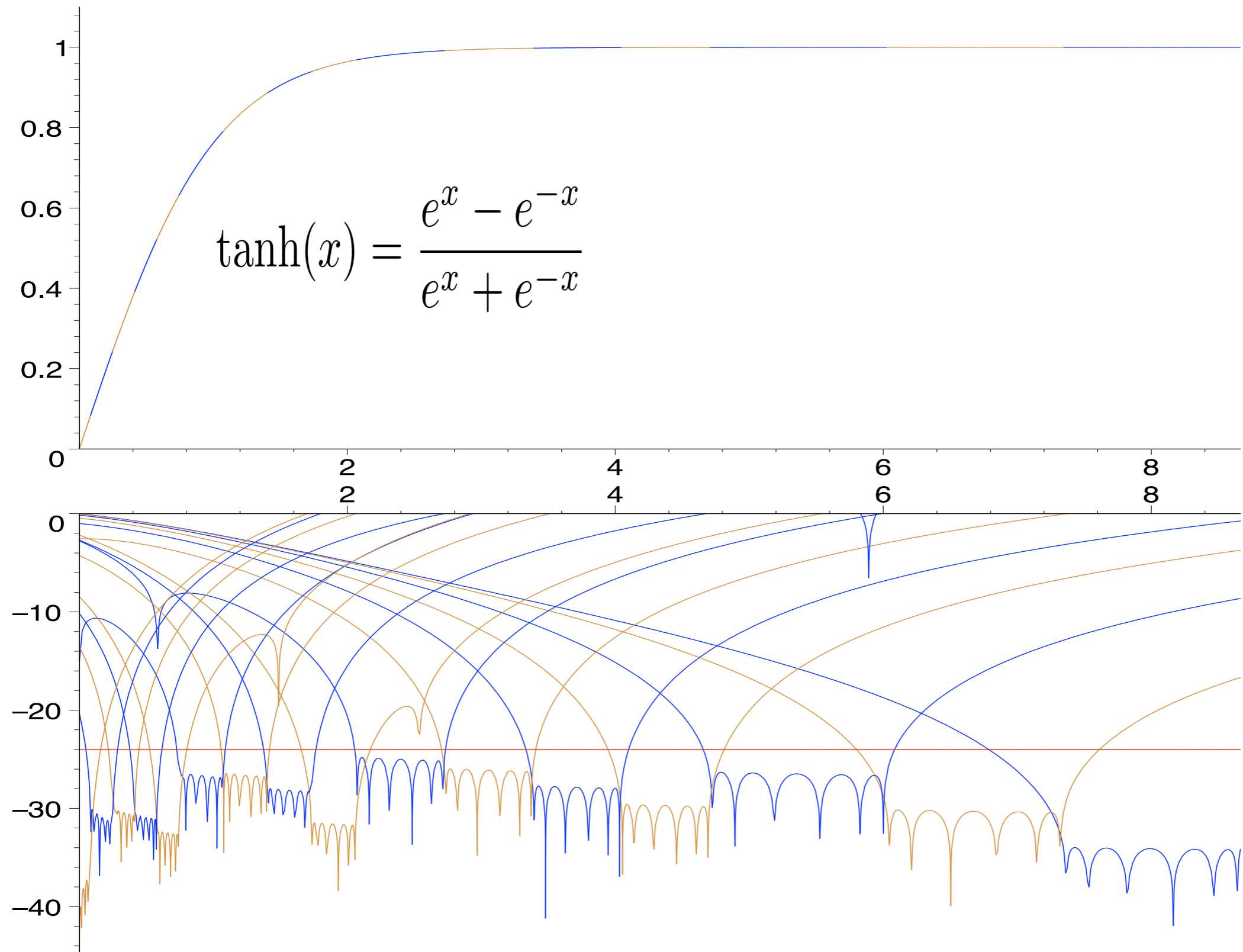
- Distant Parallelism

-  verification via model checking

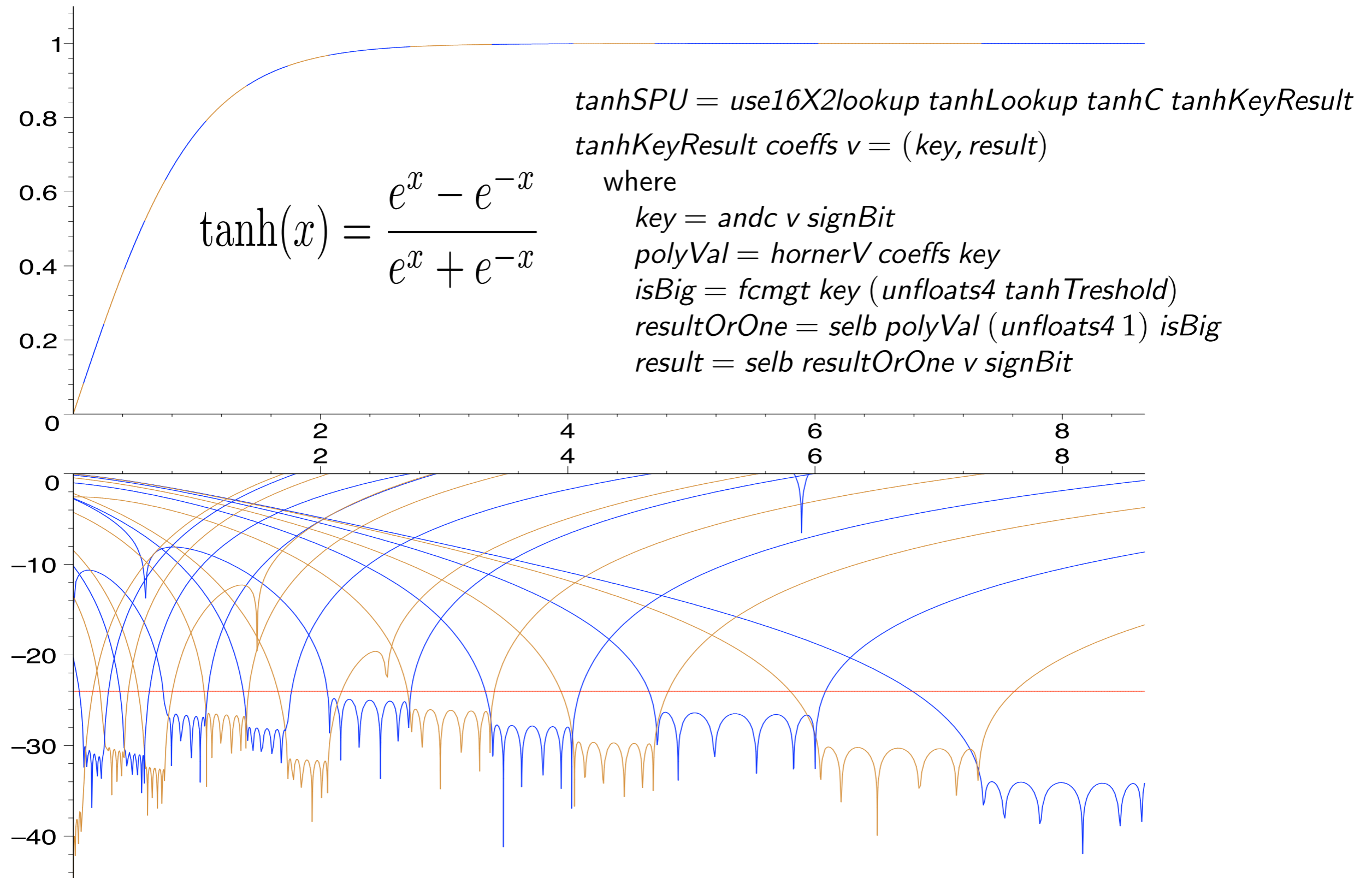
 Scheduling: ExSP



Compact Code



Compact Code



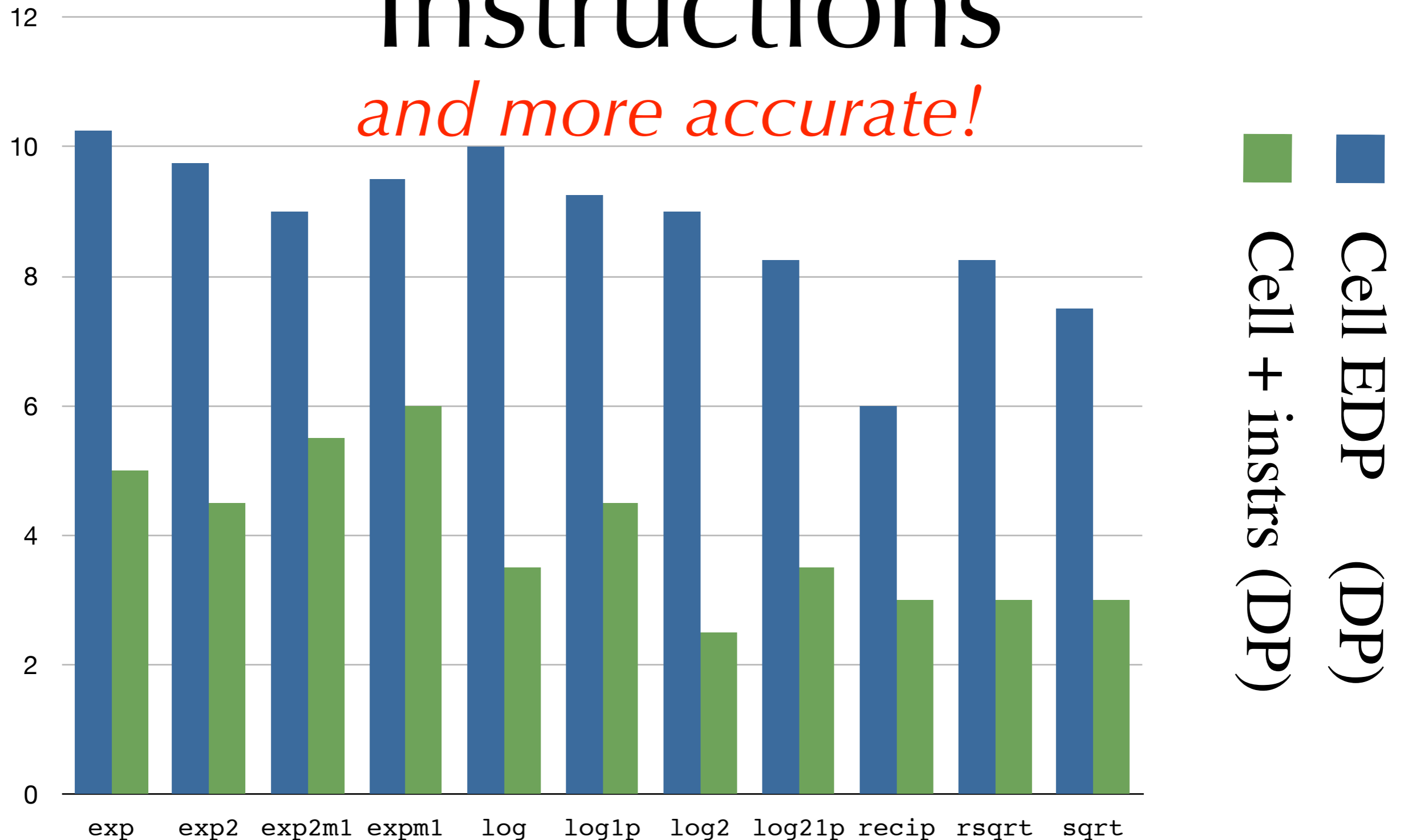
Rapid Prototyping

- 5X improvement not magic, it is
 - new algorithm ideas
 - pattern capture
- requires
 - rapid prototyping
 - refactoring

Another 2X from *New*

Instructions

and more accurate!



Roadmap

- SIMD Parallelism

- ✓ extensible DSL captures patterns
- 1/2 verification via graph transformation
- ✓ generated library shipping (Cell BE SDK 3.0)

- Multi-Core Parallelism

- ✓ model on ILP
- generation via graph transformation
- ✓ linear-time verification
- run time

- Distant Parallelism

- ∞ verification via model checking

Where is
Control Flow?

✓ Scheduling: ExSSP

Goal

- expose code graph transformation on the user level

Implementation

- explicit graph transformations
- capture correctness using type safety

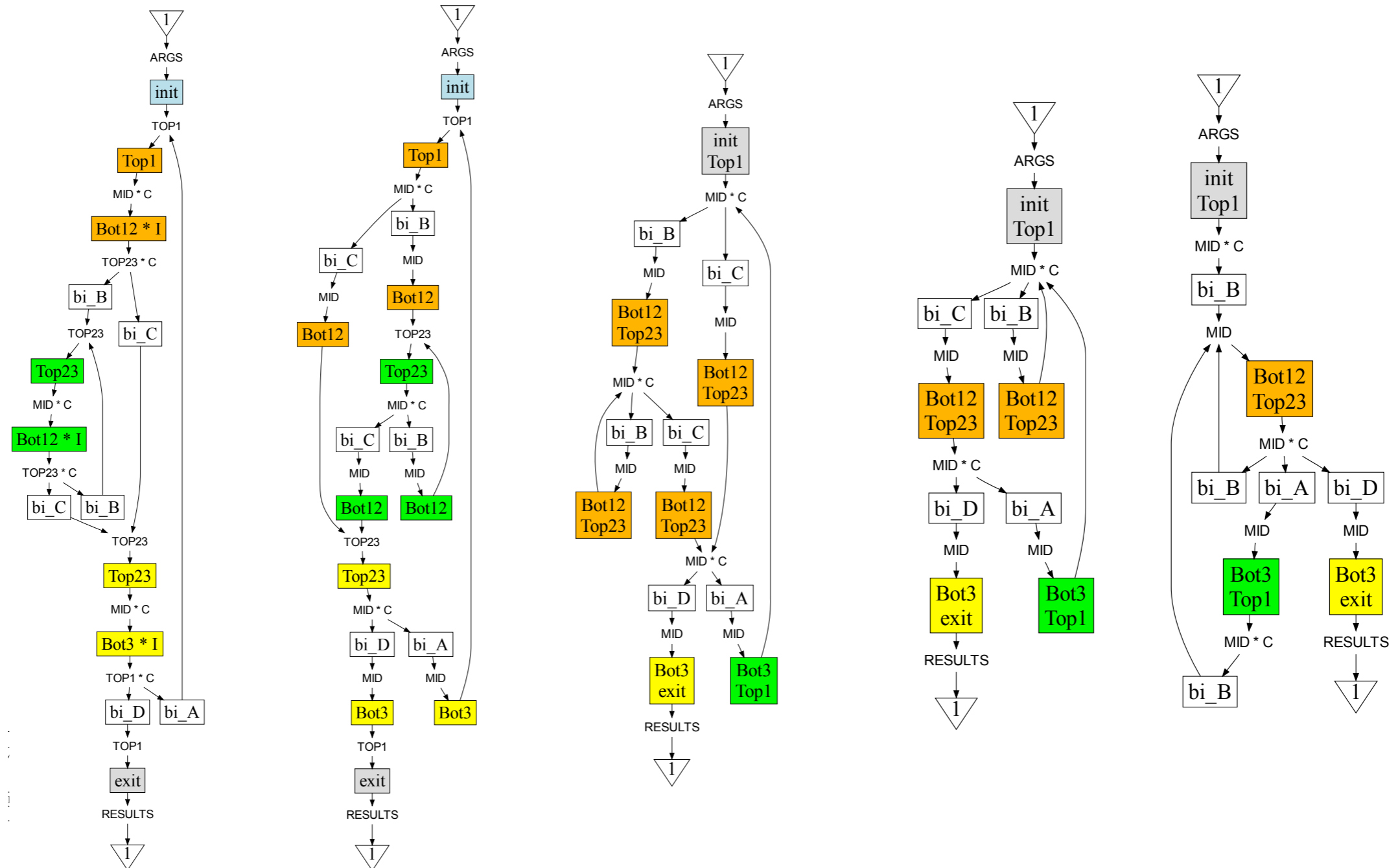
Plan

- orthogonal aspects into DSLs
- principled graph transformations

Reality

- control flow graphs on another plane of existence

Control Flow Rearrangements



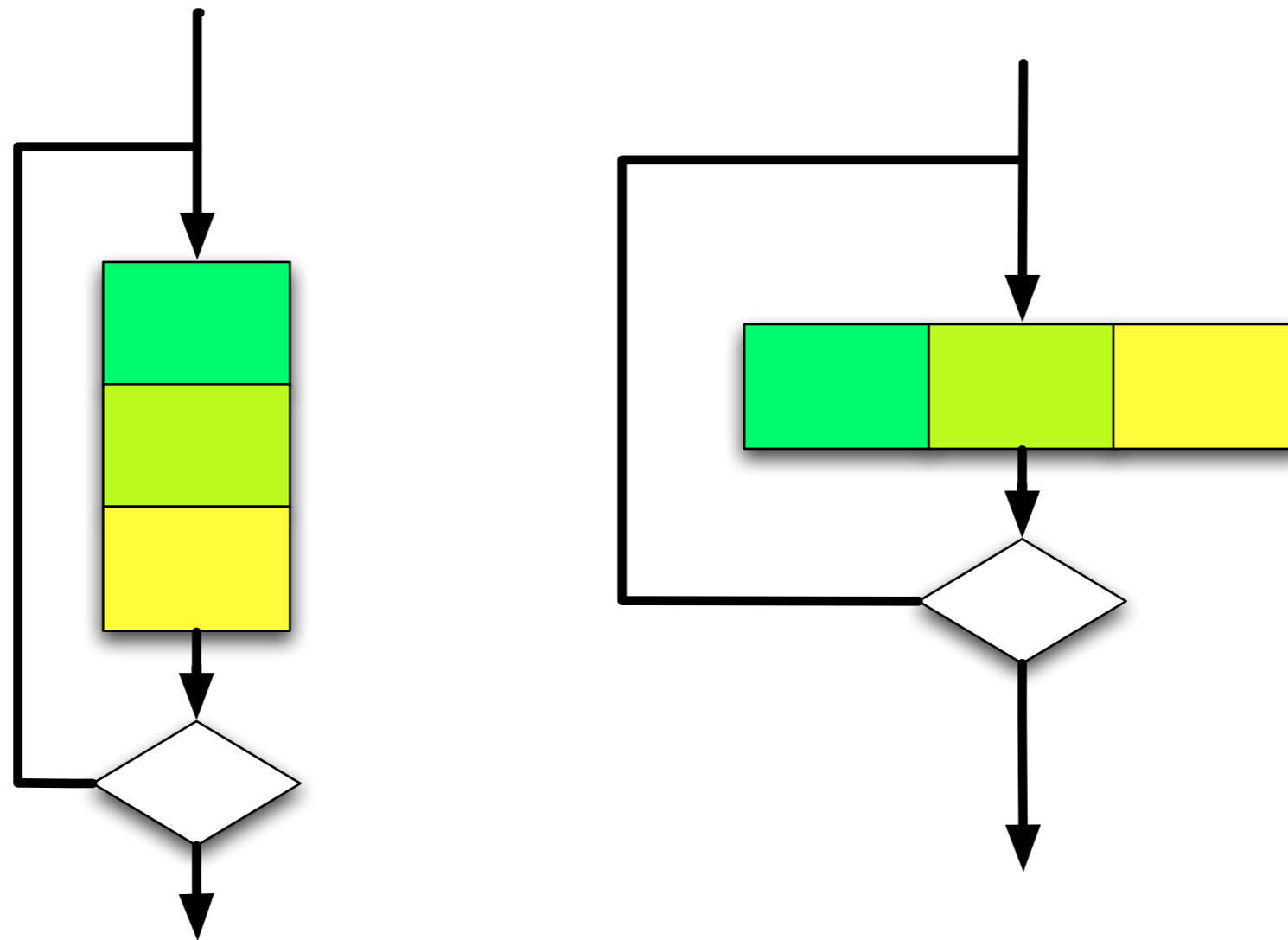
Problems

- need to program using graph language
 - hard to program
 - cannot see control flow and computation at the same time
 - lost interpretation
 - 10-page type errors
- only student who understood it also won the ICFP

Split approach

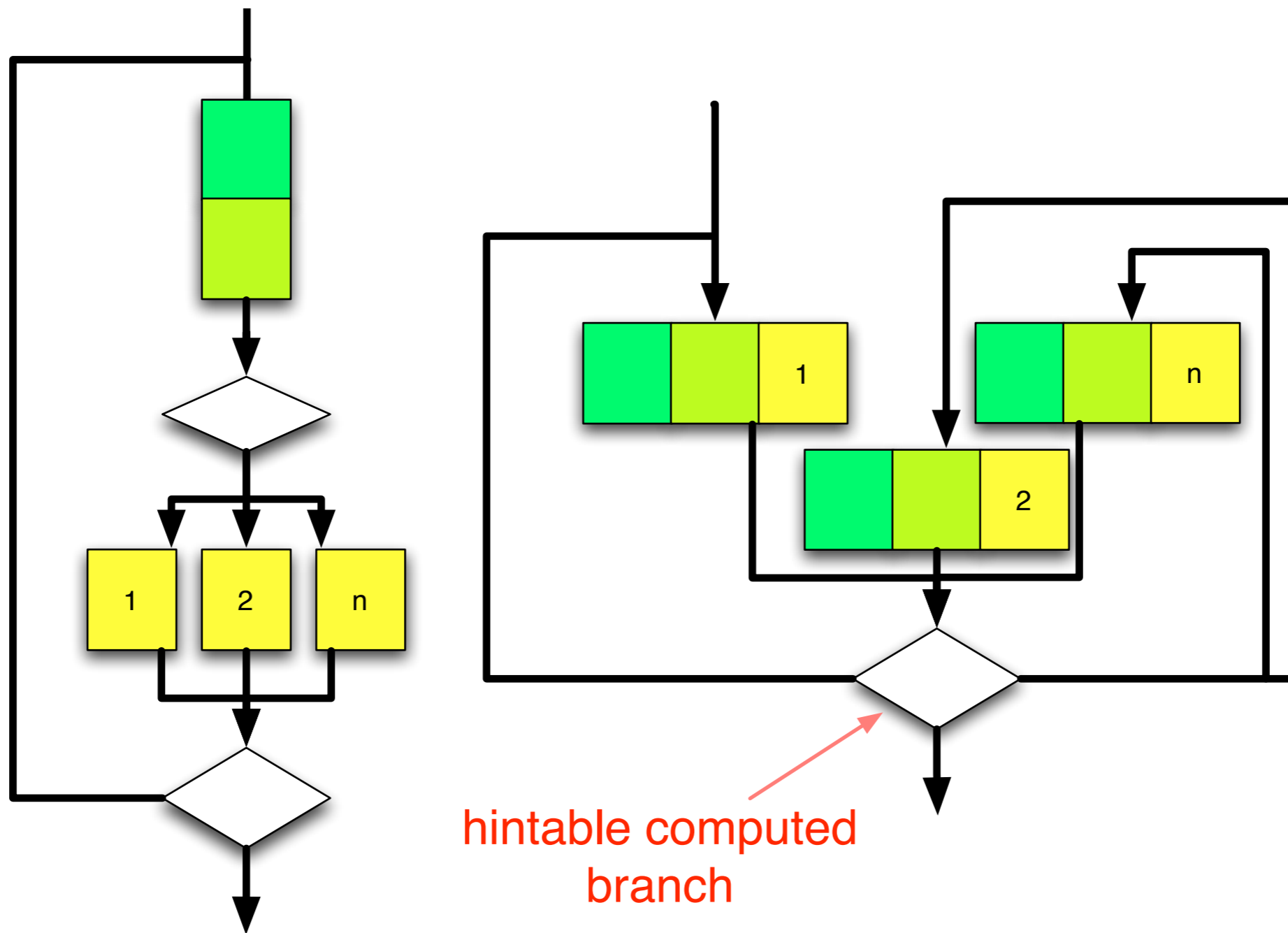
1. Control-Flow Rearrangements
 - Let user specify functionally
2. **E**xplicitly **S**taged **S**oftware **P**ipelining
 - Min-Cut to Chop into Stages
 - Principled Graph Transformation

Software Pipelining

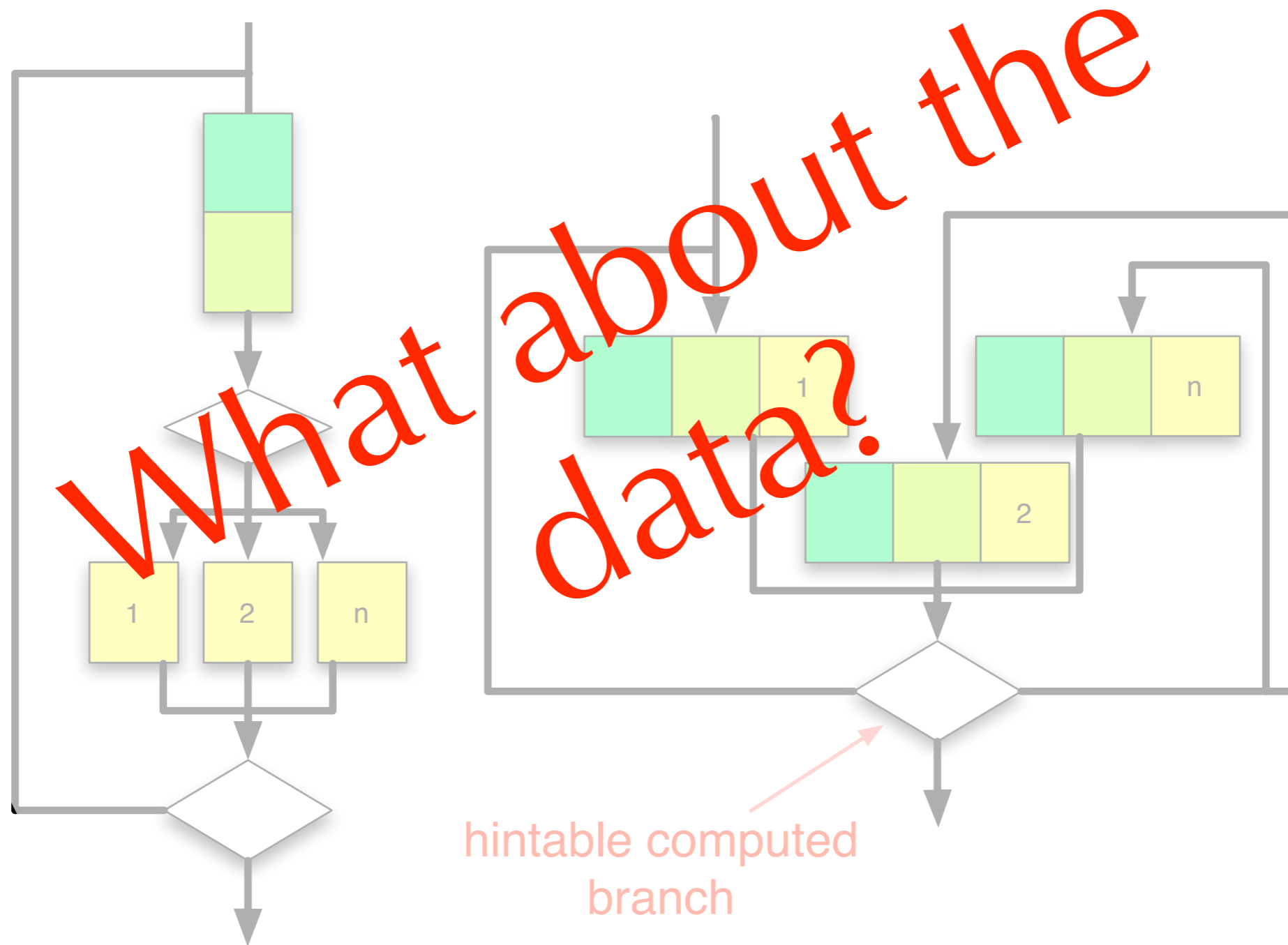


- hide latency
- same length loop body

MultiLoop



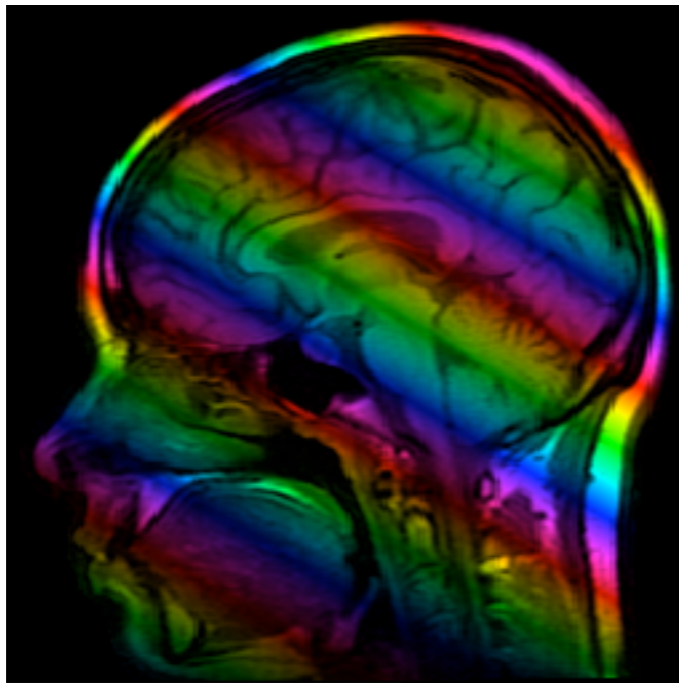
MultiLoop



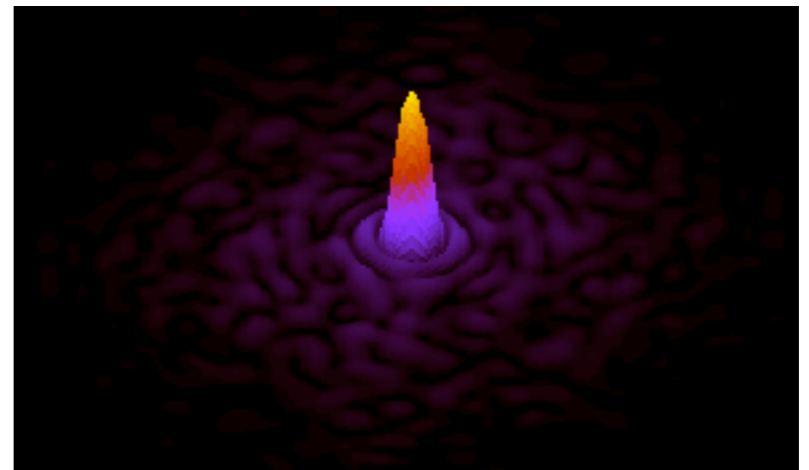
Example: Fast MRI

$$s(t) = \int_{\mathbb{R}^3} e^{i\langle x, k(t) \rangle} \rho(x) dx,$$

head



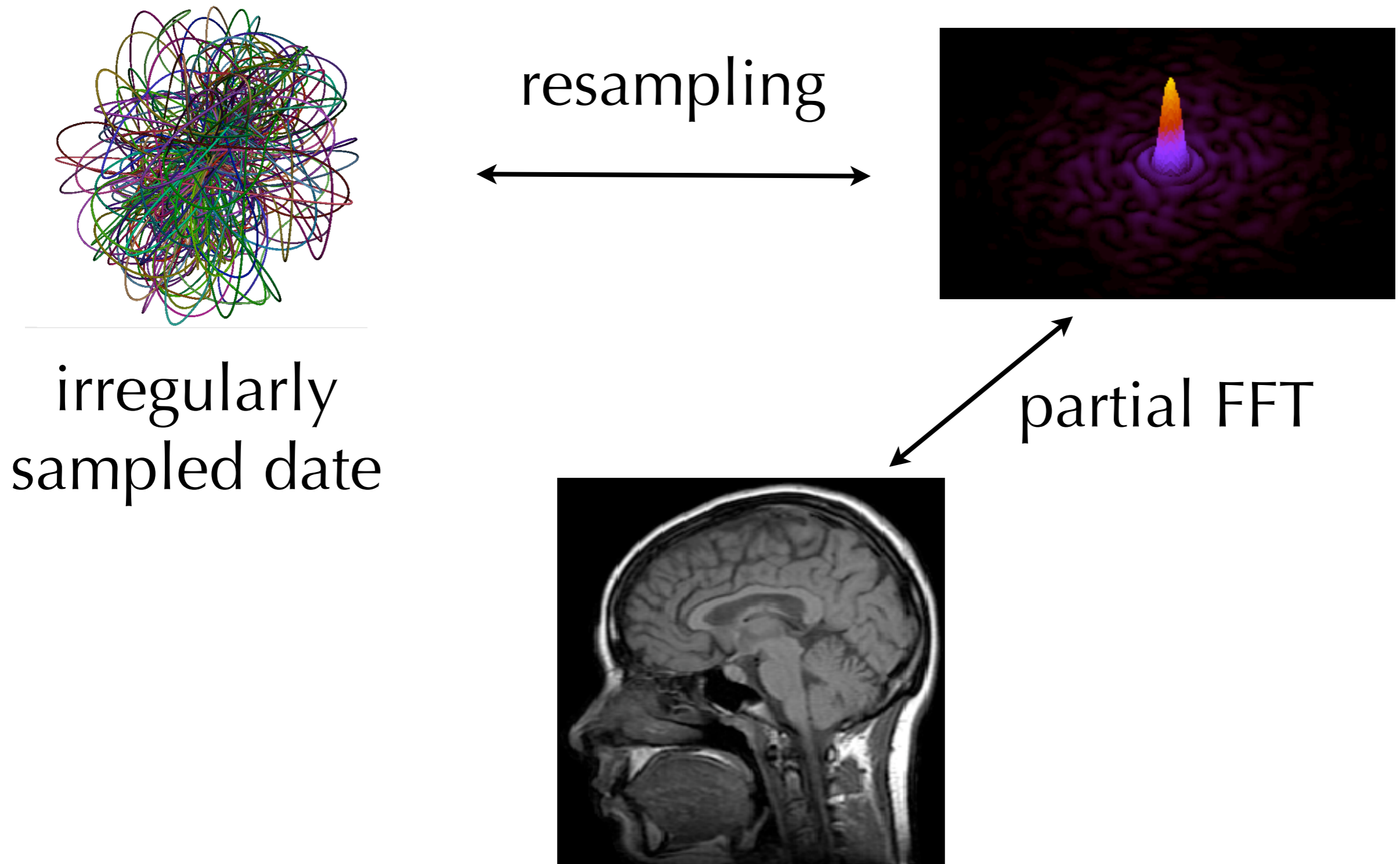
data



Fourier
Transform

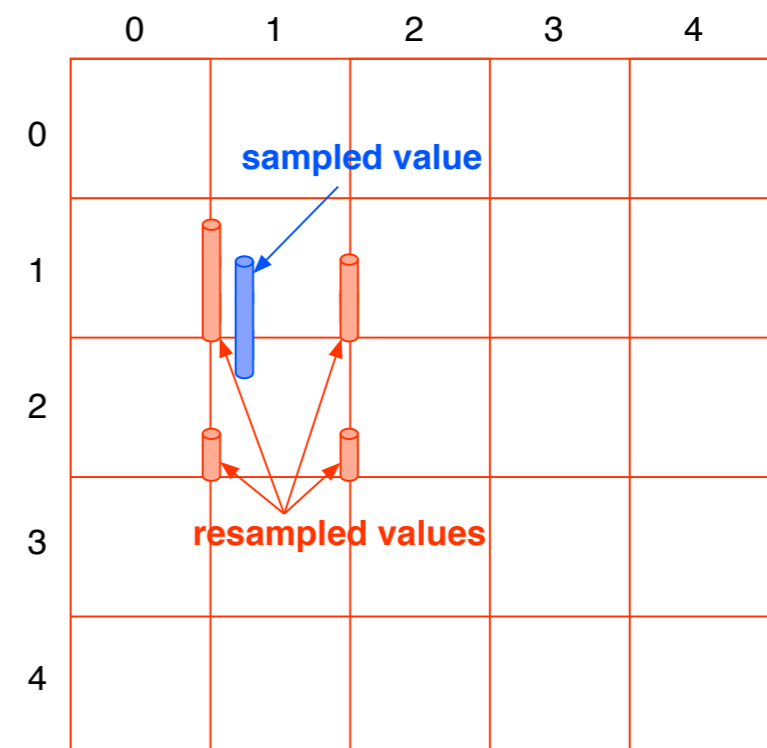


Fast Experiment \neq FFT



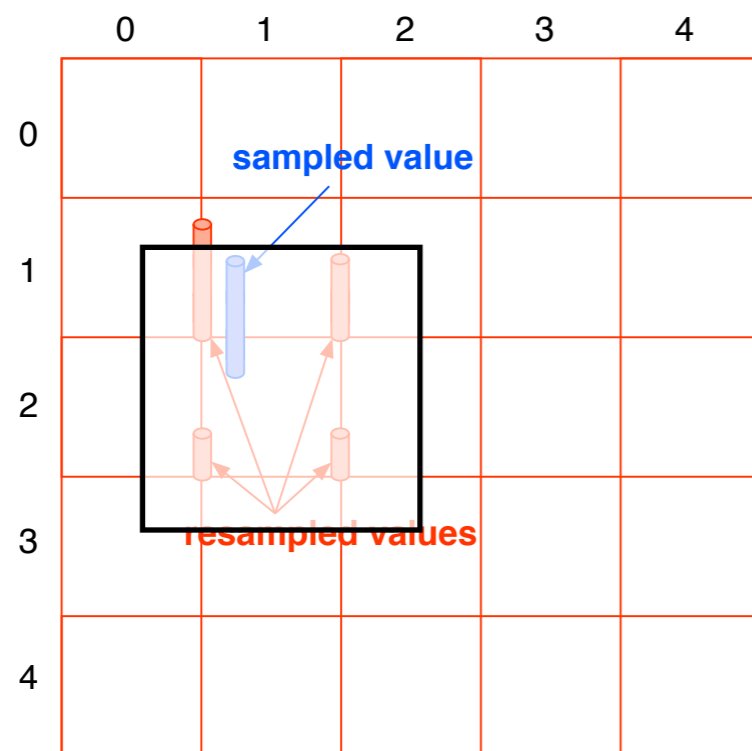
Resampling

- convolution with compact function
- outer product of vectors
- sin/cos evaluation
- accumulation in array
 - keep array in registers



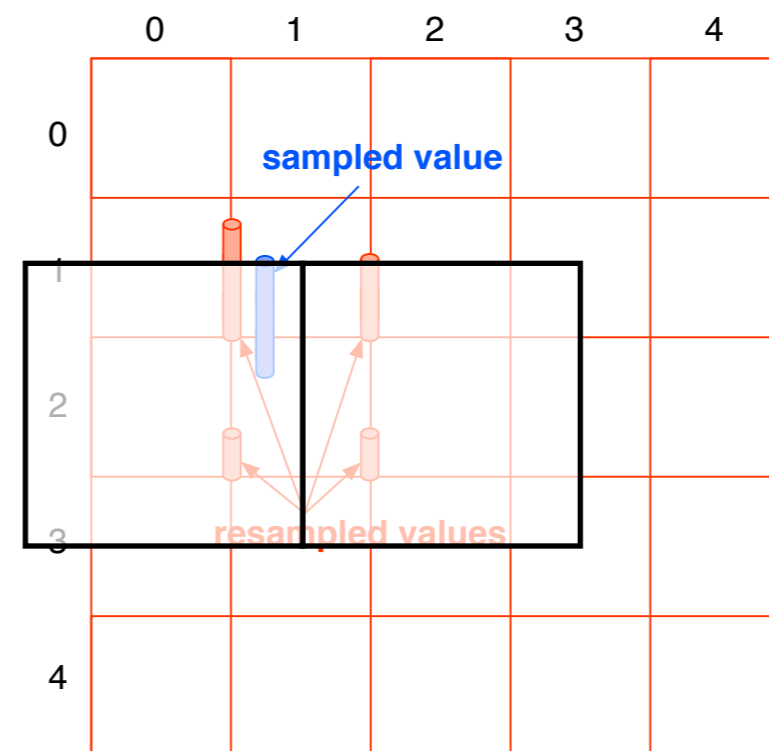
Two Cases

aligned



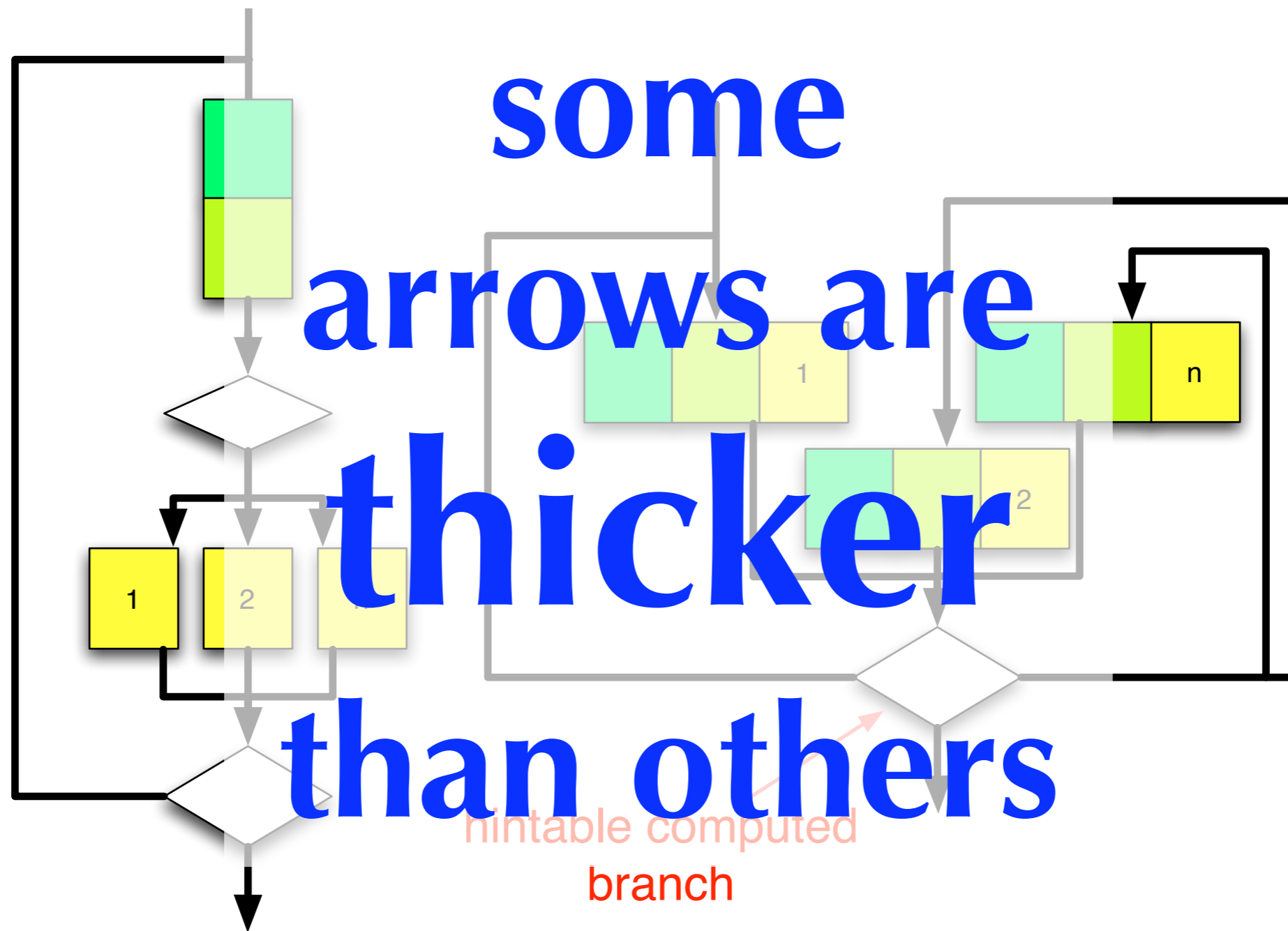
2 vector registers

unaligned



4 vector registers

MultiLoop



(multi)iterate

```
multiLoop8 :: (Integer,Integer)
```

case index range

```
-> MSCases8 swCom sw0 sw1 sw2 sw3 sw4 sw5 sw6 sw7 swExit
```

initial data

```
-> ((Int, MSCases8 swCom sw0 sw1 sw2 sw3 sw4 sw5 sw6 sw7 swExit)
    -> ([ (Int,key) ]
        , Maybe (MSCases8 swCom sw0 sw1 sw2 sw3 sw4 sw5 sw6 sw7 swExit
                      , [(String,key)]))
    )
```

function to iterate (lazily evaluates data to key)

```
-> (swExit, [(String,key)])
```

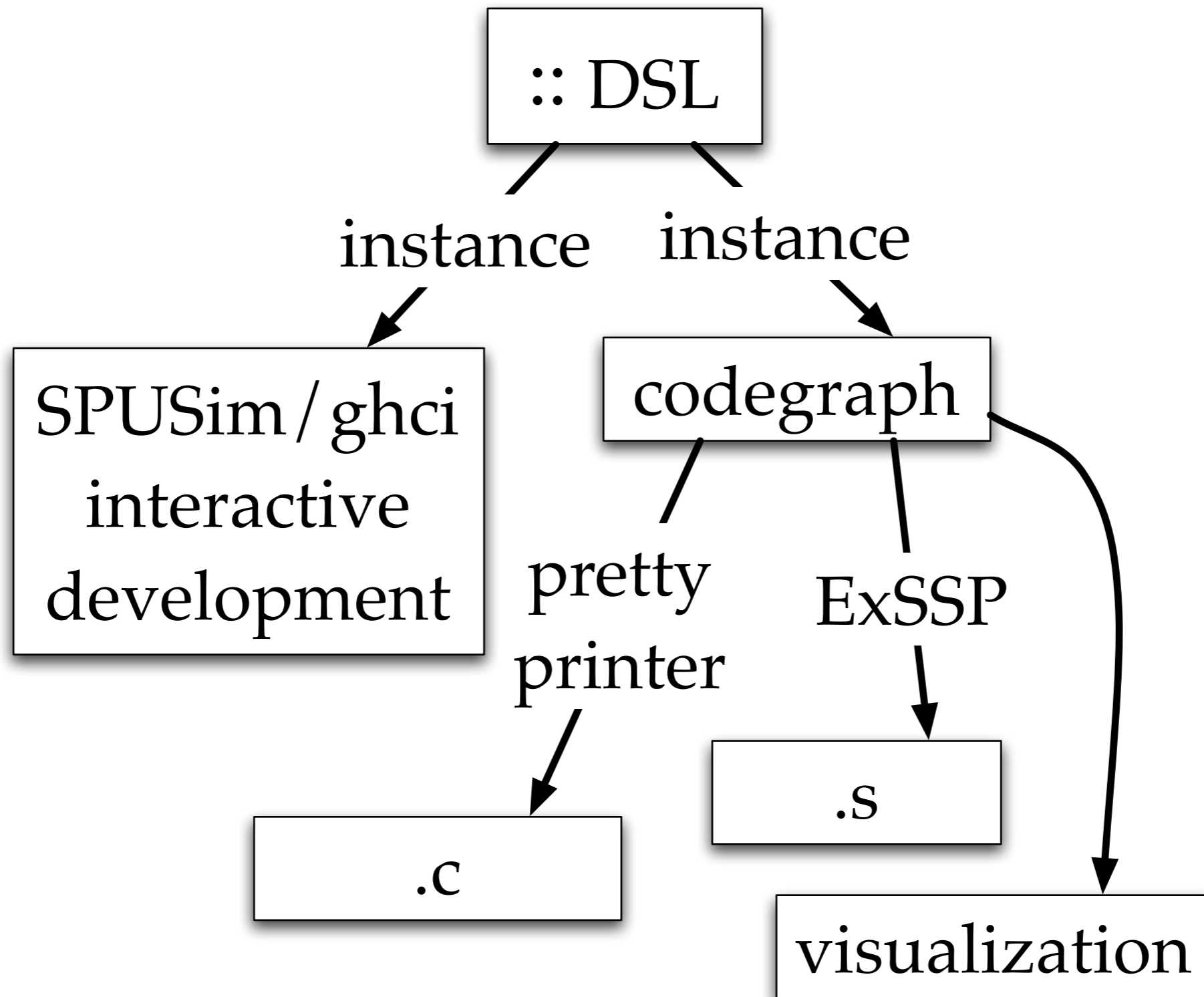
Example: PFT3d

(24to16)

```
pft3d memRegion
  = multiLoop8 "PFT3.body" (0,3) (startLoop mr) body

body (codeIdx, MS8C0 (ticker, (vsIn, mr)) ())
  = ([ (2,bKey)], Just (next, dbg++msDbg) )
  where
    ((ticker',bKey,key),dbg) = inductionPFFT24 ticker
    ldStAd = head ticker
    ldStAd' = head ticker'
    vsOut = quadRowFT vsIn
    (next,msDbg) = if codeIdx == 3
      then (MS8CExit $ meet "stores should commute"
            $ storeXRowsZ (xySize 2) (stAddr ldStAddr) vsOut mr,[])
      else let (vsIn',ldMRs)= loadYCol (xSize codeIdx) (ldAd ldStAd') mr
        vsIn'' = case codeIdx of
          0 -> riririri2rrrrriiii vsIn'
          _ -> vsIn'
        stMRs = storeXRowsZ (xySize codeIdx) (stAddr ldStAddr) vsOut mr
      in  (MS8C0 (ticker',(vsIn'', meet "load/stores should commute"
                                (ldMRs++stMRs)))
          (),[])
```

Assembly embedded in Haskell



Interpreter Semantics

- **Just Haskell Data Types**
- folds lazily
- data-determined case and returned as input
 - different cases constructed using Haskell control flow
- specify (limited) re-orderable load/store
 - interpreter **verifies correctness**

Codegraph Semantics

- $\{\text{indices}\} \times \{\text{data cases}\}$ loop bodies
- iterate through and generate valid cases
- cut common codegraph
- restrict case-specific code to last stage
- restrict case calculation to previous stage

Partial FFT

- example of a separable transform
- SIMD doesn't like row transform
- do column transform + transpose

$$xyz \longrightarrow Yzx \longrightarrow ZxY \longrightarrow XYZ$$

- sequence of 3 triple-nested loops
- one MultiLoop
- pointer/counter overhead = **1 mult-add**
 - and lots of shuffles and bit rotates

Coconut Roadmap

- SIMD Parallelism

still great!

- Multi-Core Parallelism

another talk

- Itra-Core Control Flow

major bump in the road

finally back to simple model with compact text

stay tuned for performance numbers after code generation is rejiggered :)