

TASBE: A Tool-Chain for Accelerating Synthetic Biology Engineering

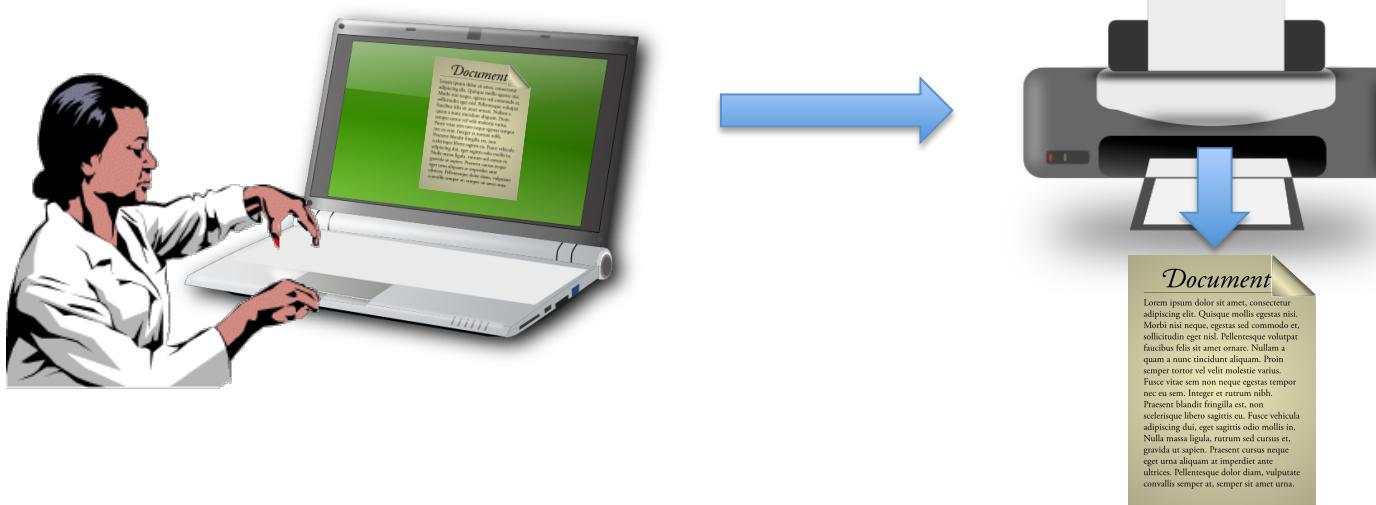
Jacob Beal, Ron Weiss, Douglas Densmore, Aaron Adler,
Jonathan Babb, Swapnil Bhatia, Noah Davidsohn, Traci
Haddock, Fusun Yaman, Richard Schantz, Joseph Loyall

3rd Int'l Workshop on
Bio-Design Automation
June, 2011



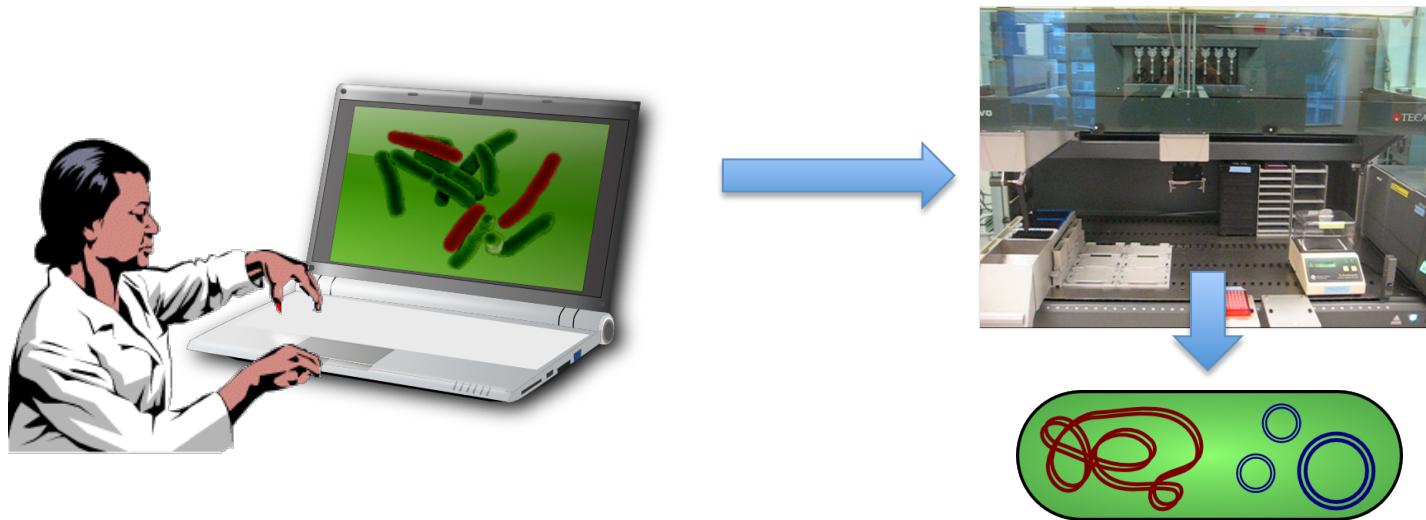
Vision: WYSIWYG Synthetic Biology

Bioengineering should be like document preparation:



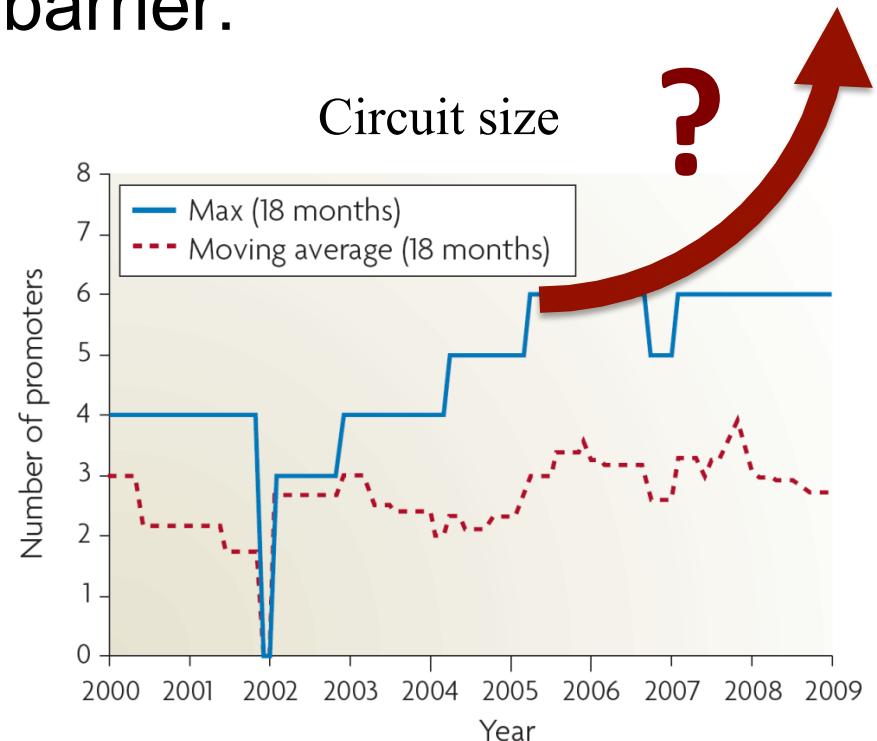
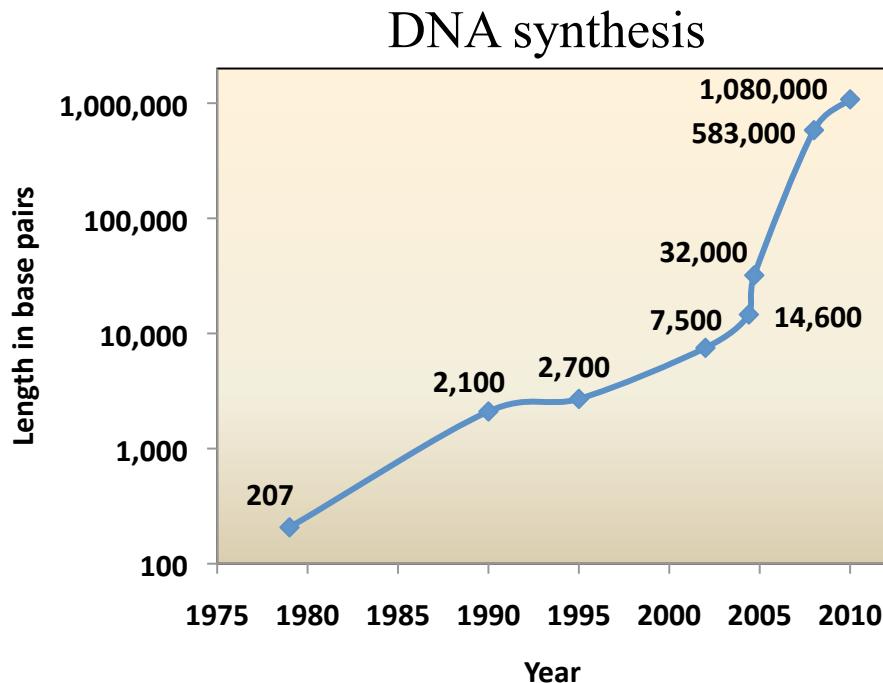
Vision: WYSIWYG Synthetic Biology

Bioengineering should be like document preparation:



Why is this important?

- Breaking the complexity barrier:



- Multiplication of research impact
- Reduction of barriers to entry

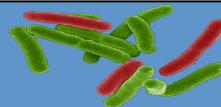
Why a tool-chain?

Organism Level Description

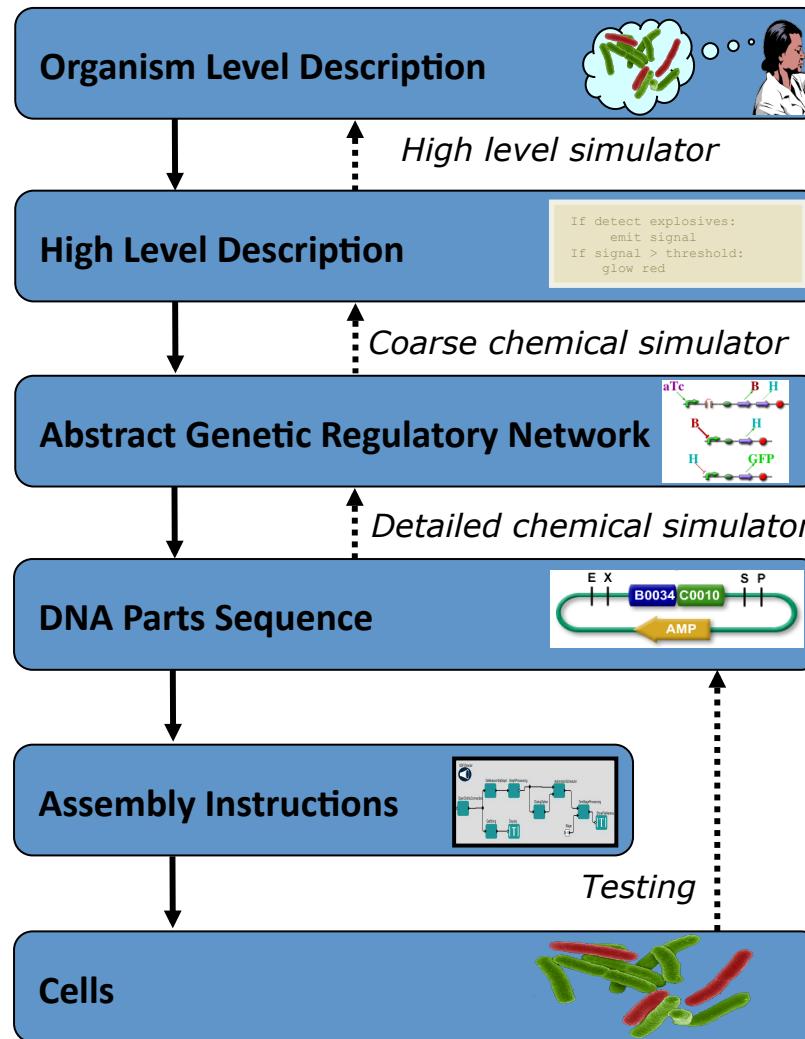


*This gap is too big
to cross with a
single method!*

Cells



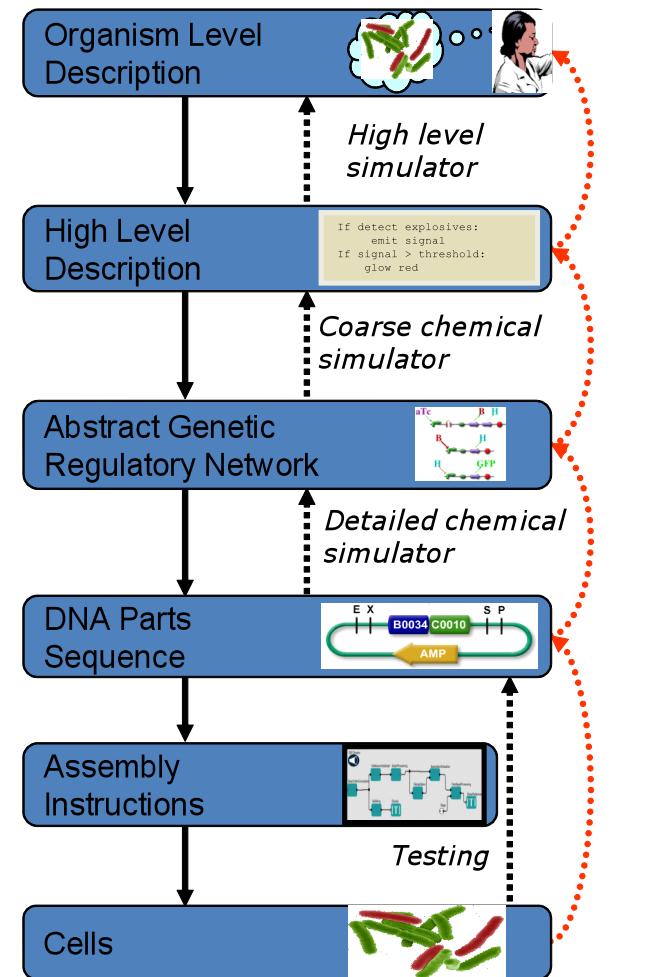
The TASBE architecture:



*Modular architecture
also open for flexible
choice of organisms,
protocols, methods, ...*

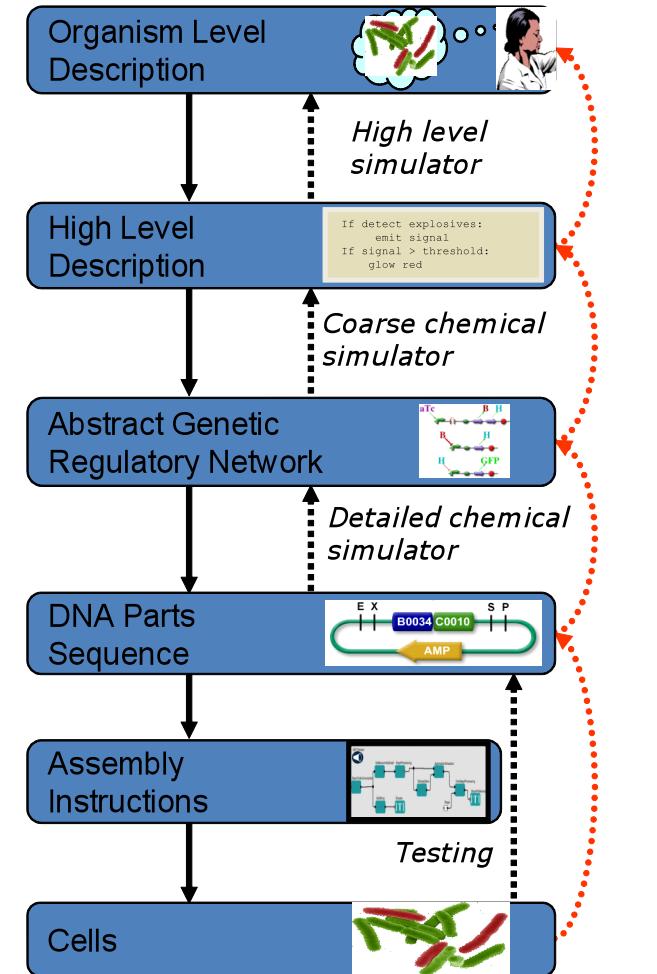
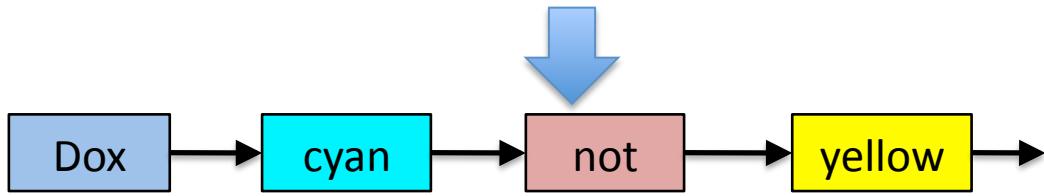
A Tool-Chain Example

(yellow (not (cyan (Dox))))



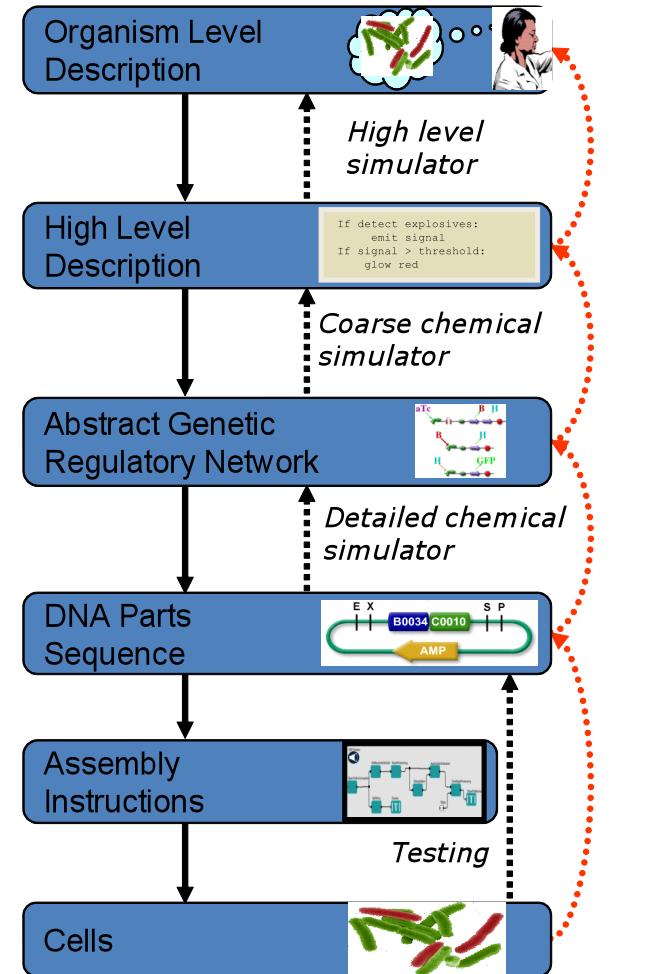
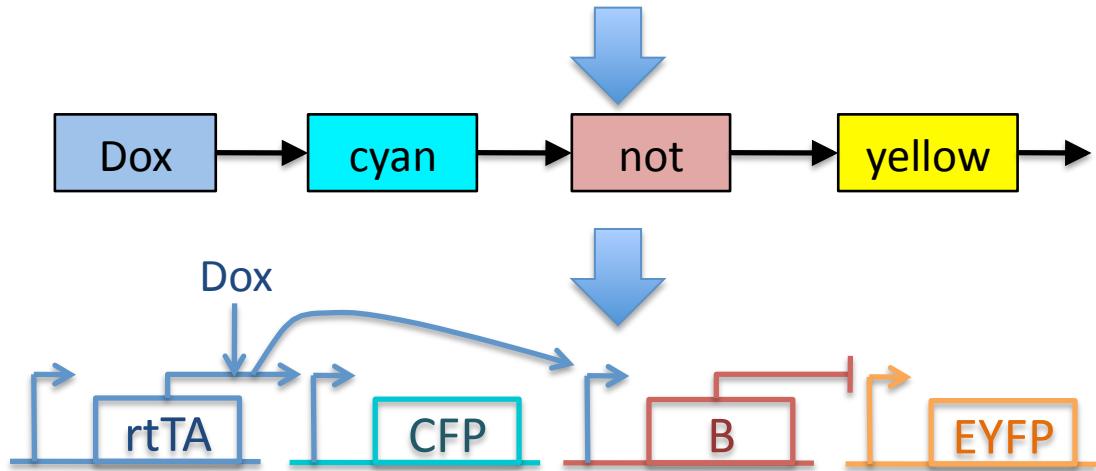
A Tool-Chain Example

(yellow (not (cyan (Dox))))



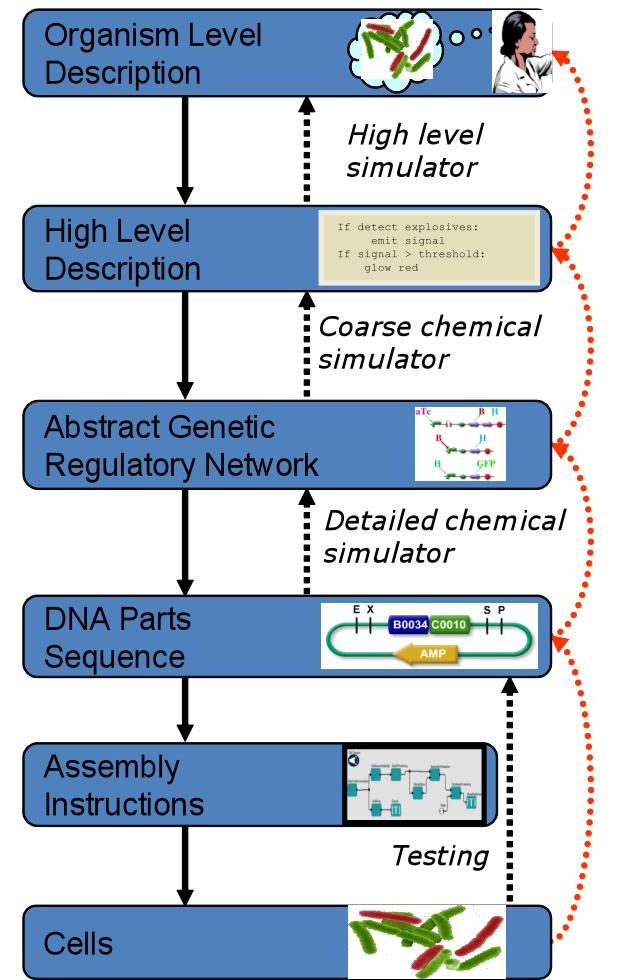
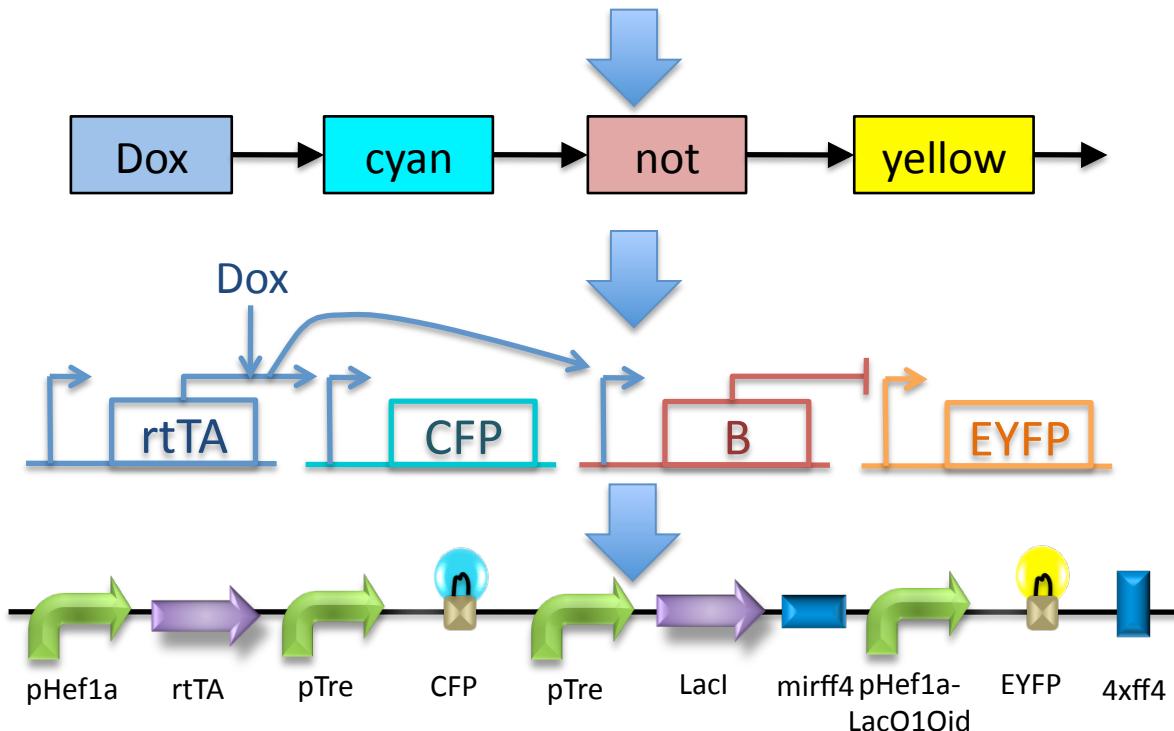
A Tool-Chain Example

(yellow (not (cyan (Dox))))



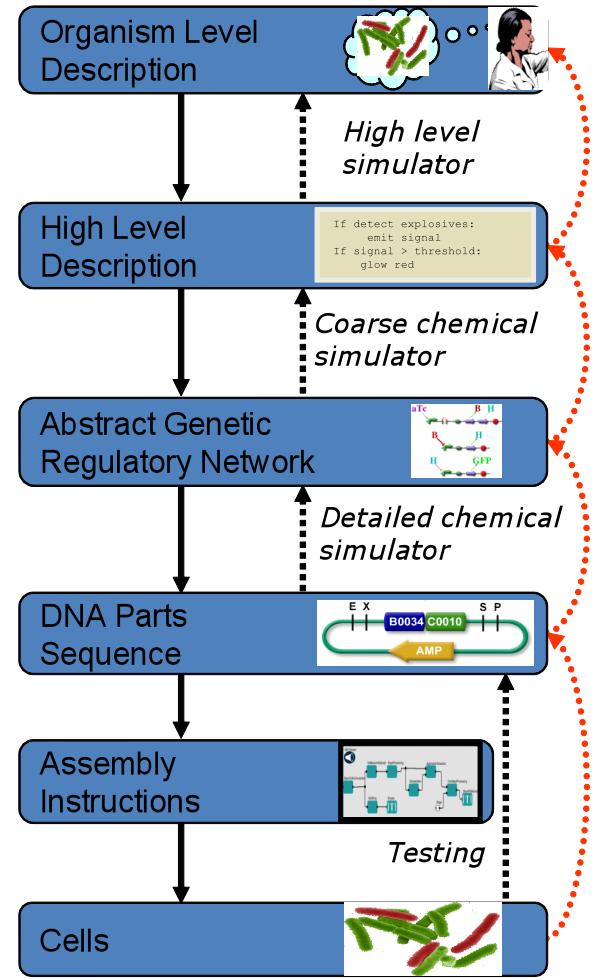
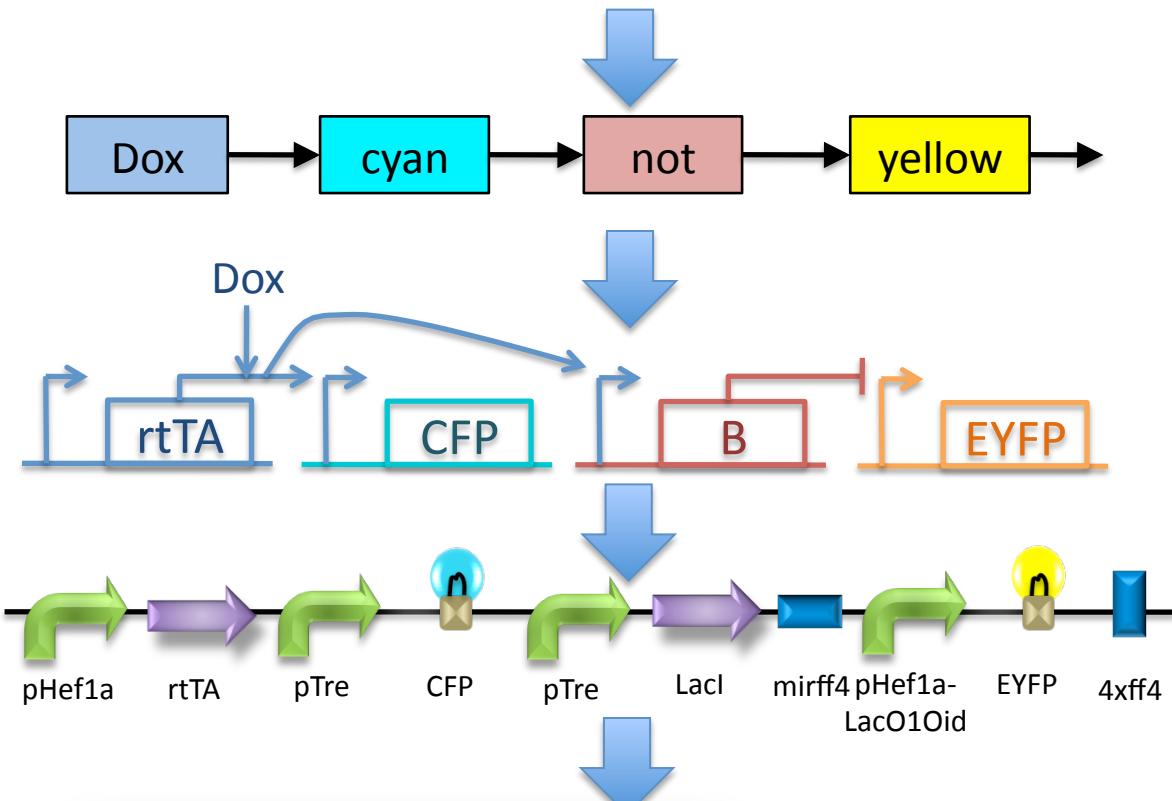
A Tool-Chain Example

(yellow (not (cyan (Dox))))



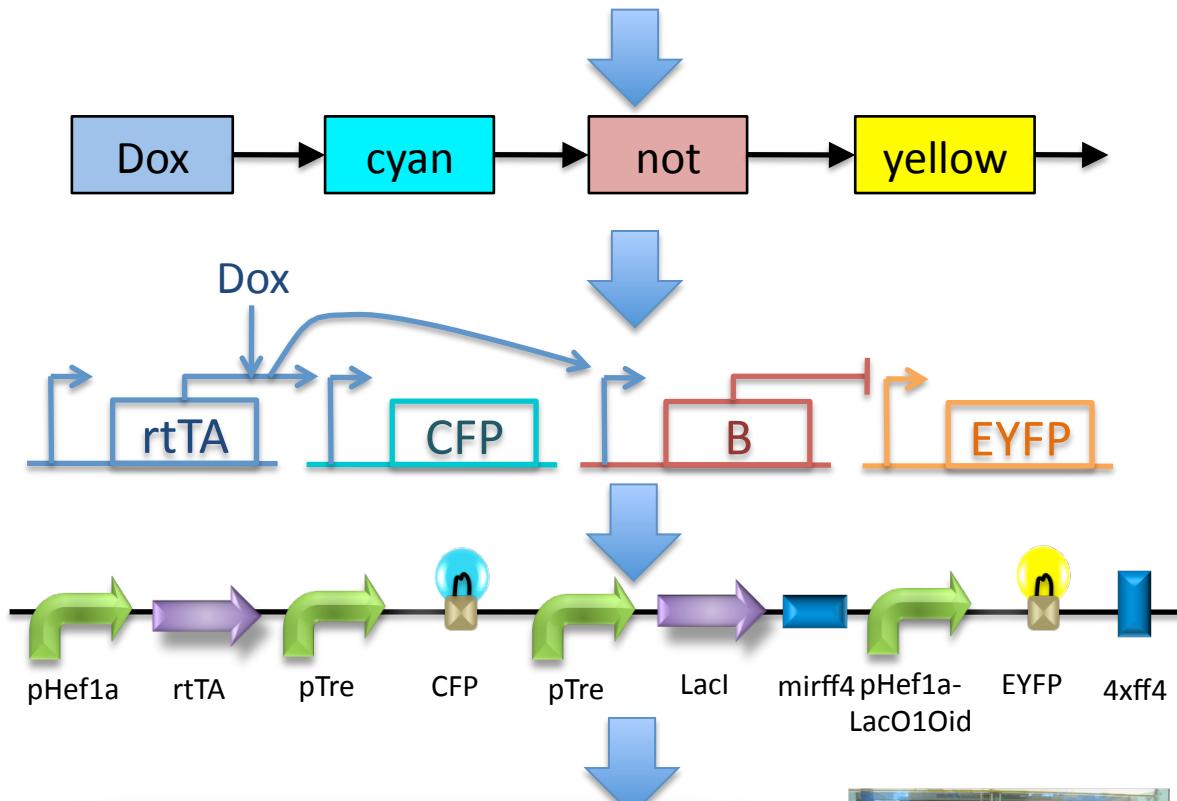
A Tool-Chain Example

(yellow (not (cyan (Dox))))

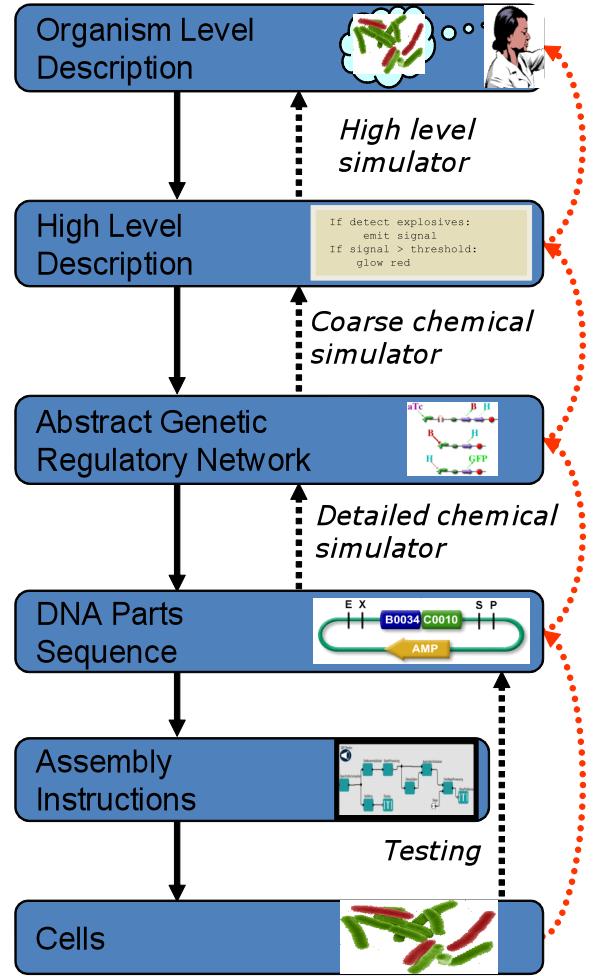


A Tool-Chain Example

(yellow (not (cyan (Dox)))))



44	 Aspirate		20 µl	training "Work" (Col. 1, Row 2)
45	 Dispense		20 µl	Water free dispense "Work" (Col. 1, Row 3)
46	 Aspirate		20 µl	training "Work" (Col. 2, Row 2)
47	 Dispense		20 µl	Water free dispense "Work" (Col. 1, Row 3)



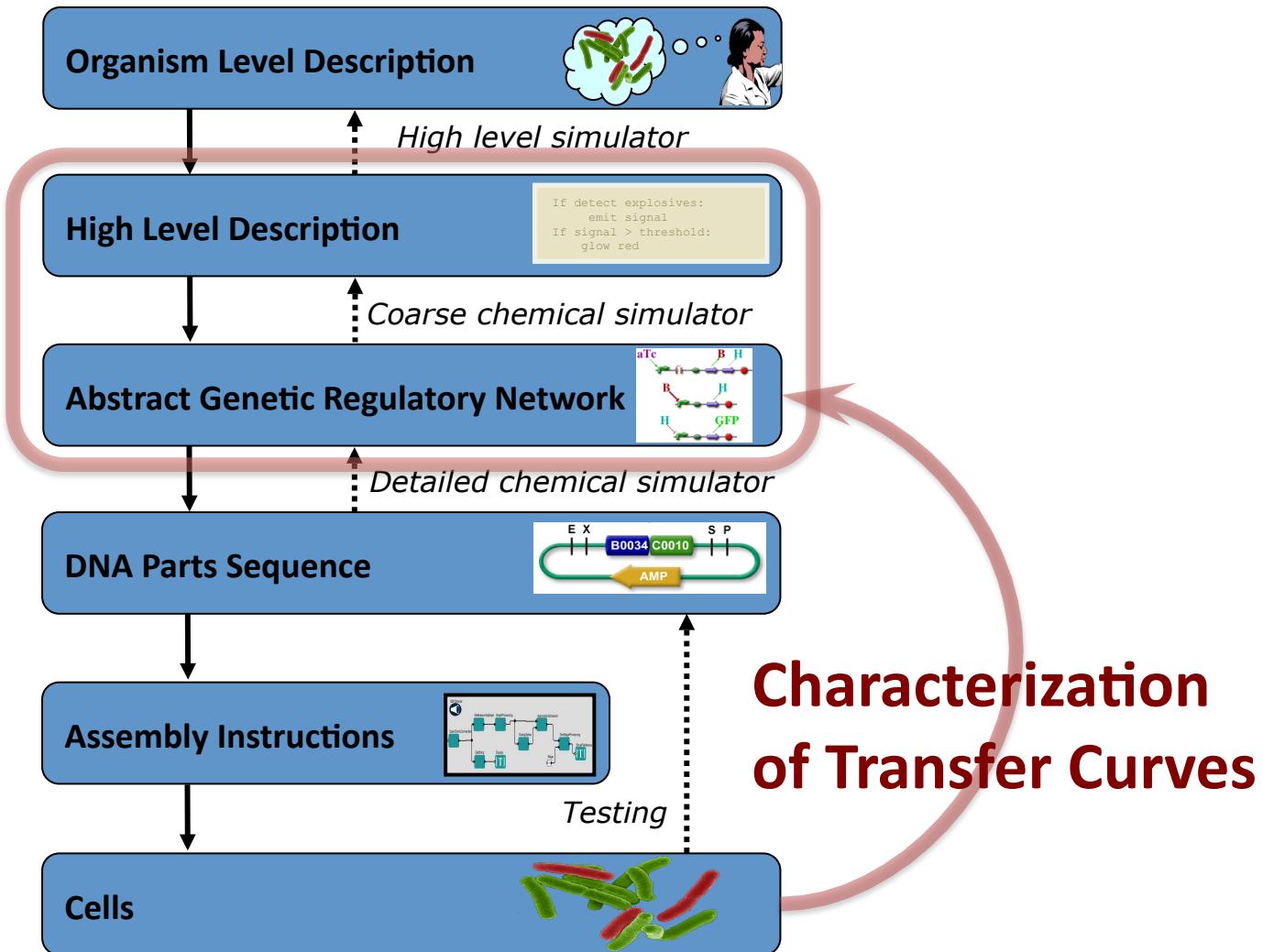
Current state of the tool-chain:

- End-to-end software integration
- Automated designs match hand-generated systems verified *in vivo*
- Some protocols automated

Next: verification of automated design & assembly

Advances on Two Key Problems:

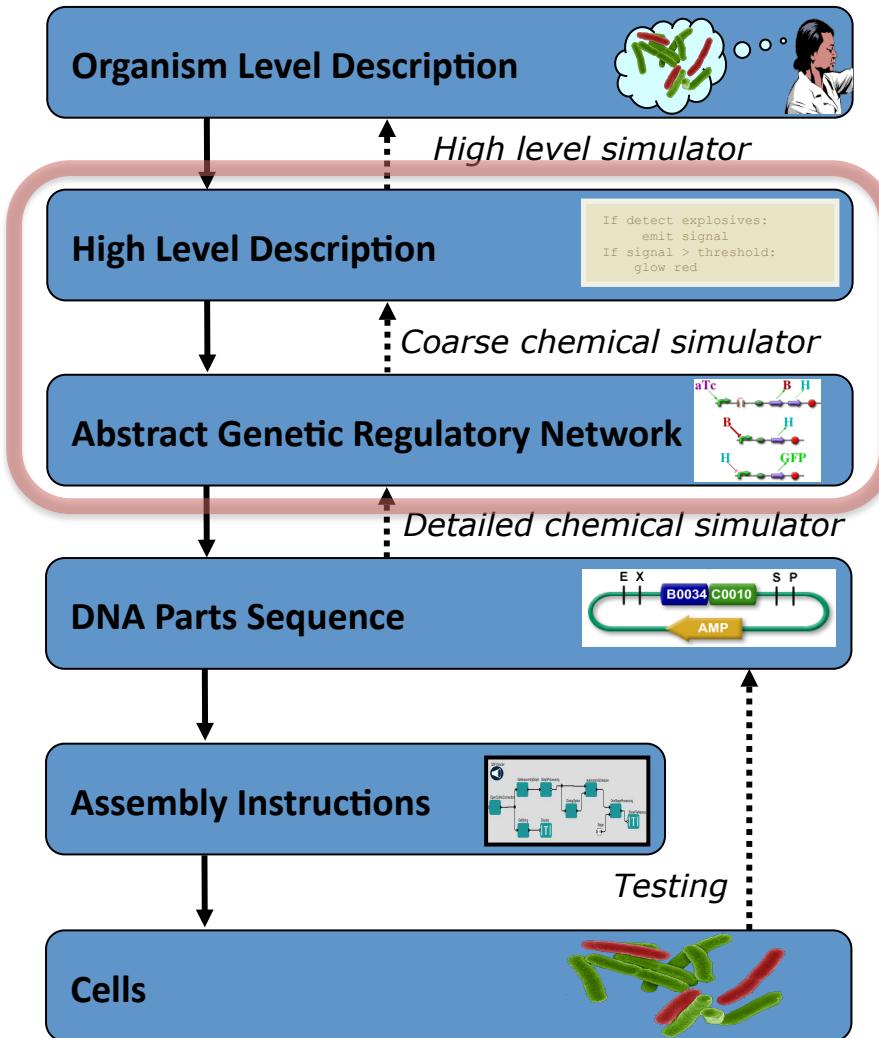
Compilation & Optimization



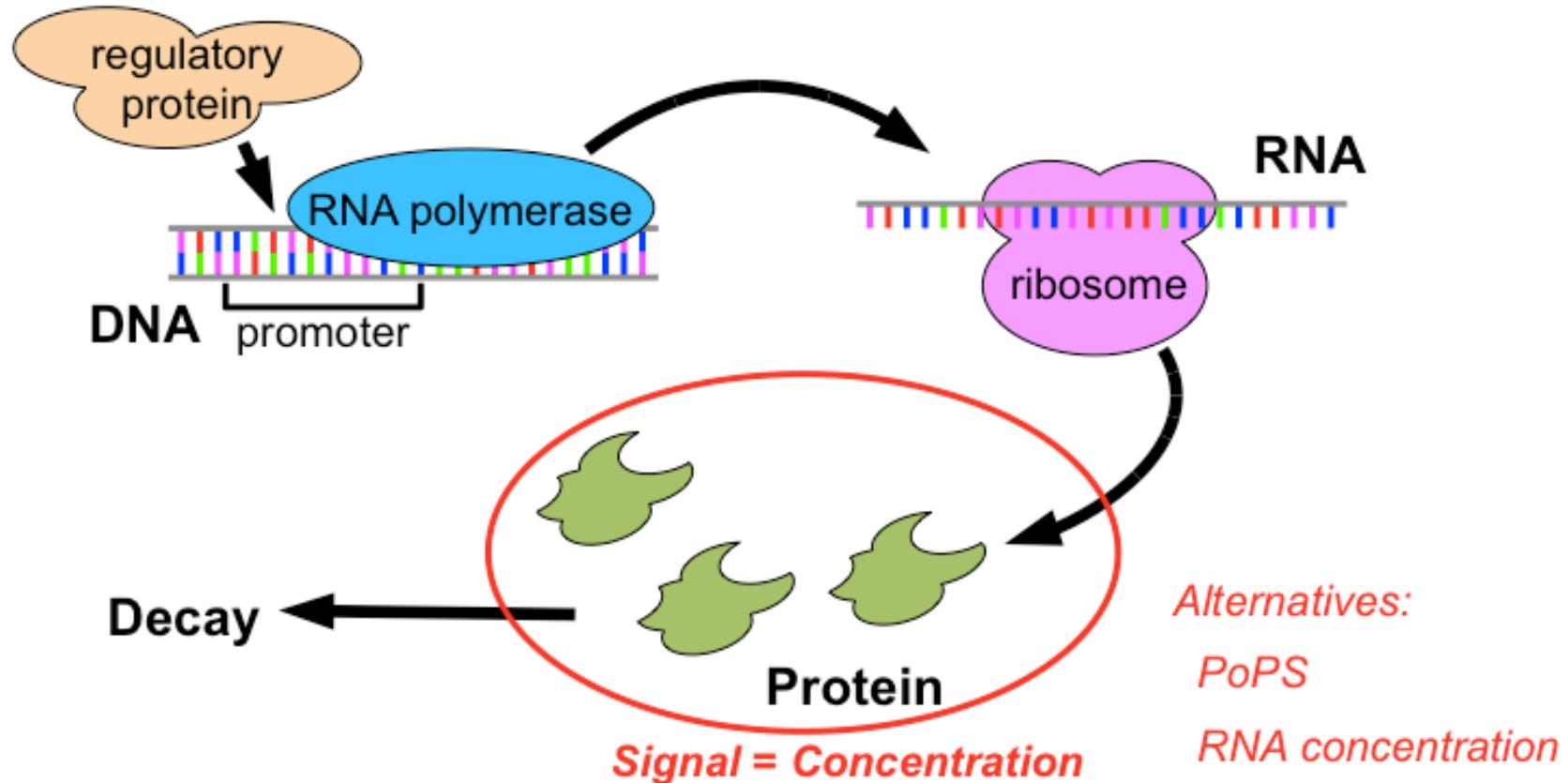
**Characterization
of Transfer Curves**

Advances on Two Key Problems:

Compilation & Optimization

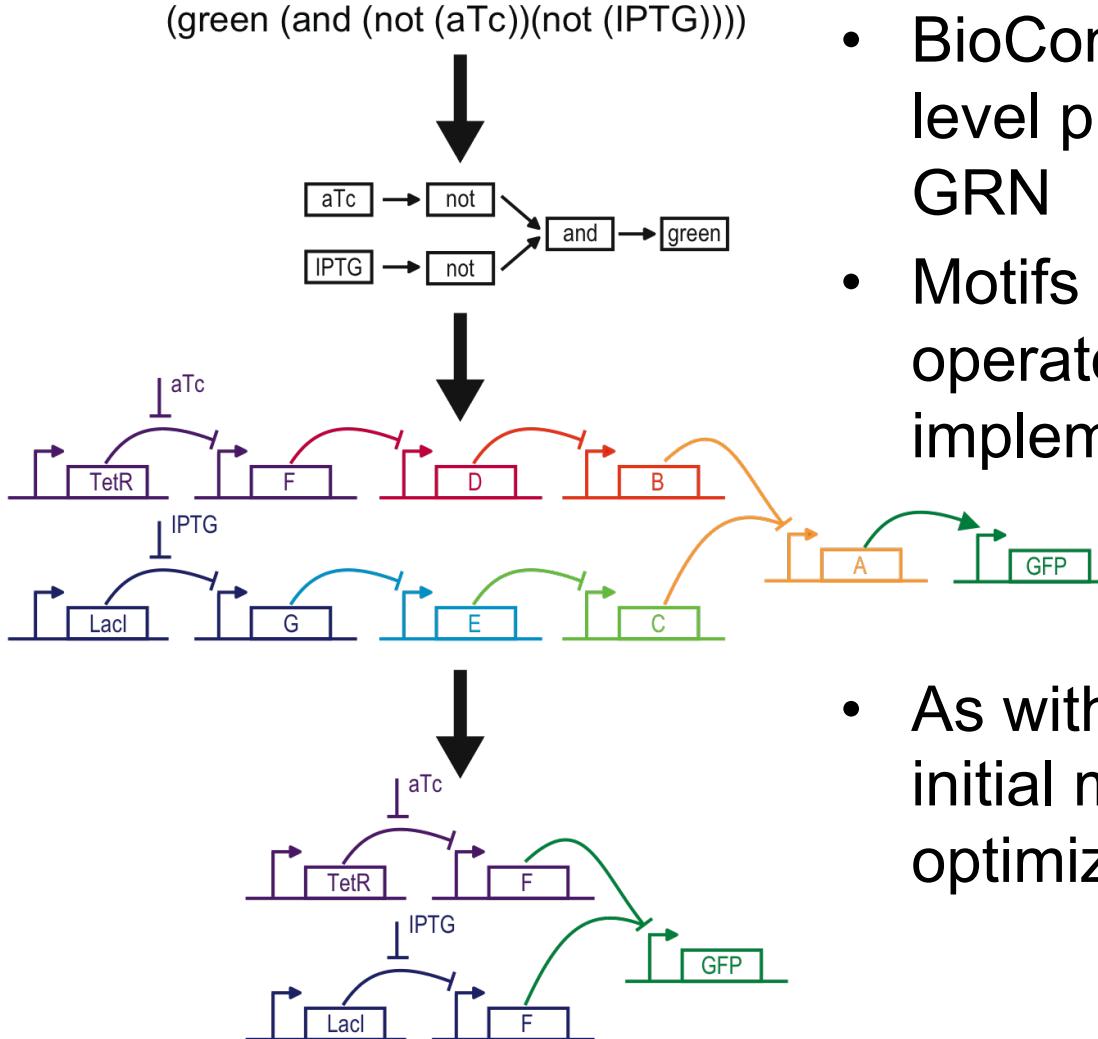


Transcriptional Logic



Stablizes at *decay* = *production*

BioCompiler Overview

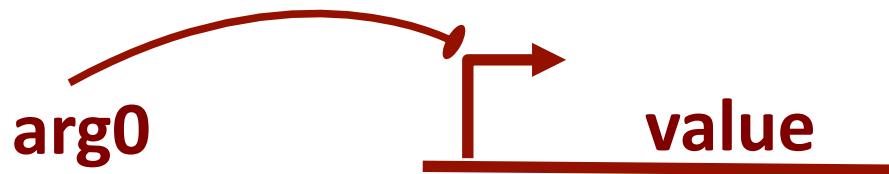


- BioCompiler converts high level program to abstract GRN
- Motifs map high level operators to parameterized implementation in biology.
- As with all compilers, the initial mapping can be greatly optimized.

Motif-Based Compilation

- High-level primitives map to GRN design motifs
 - e.g. logical operators:

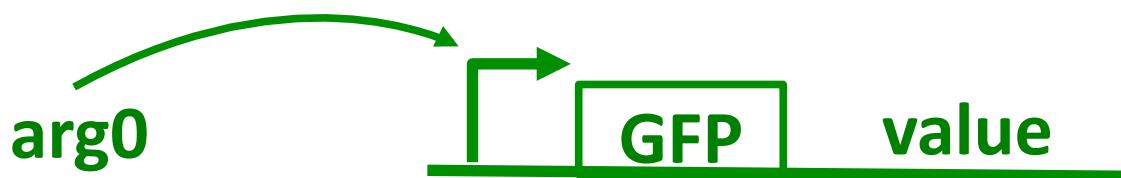
```
(primitive not (boolean) boolean  
:grn-motif ((P high R- arg0 value T)))
```



Motif-Based Compilation

- High-level primitives map to GRN design motifs
 - e.g. logical operators, **actuators**:

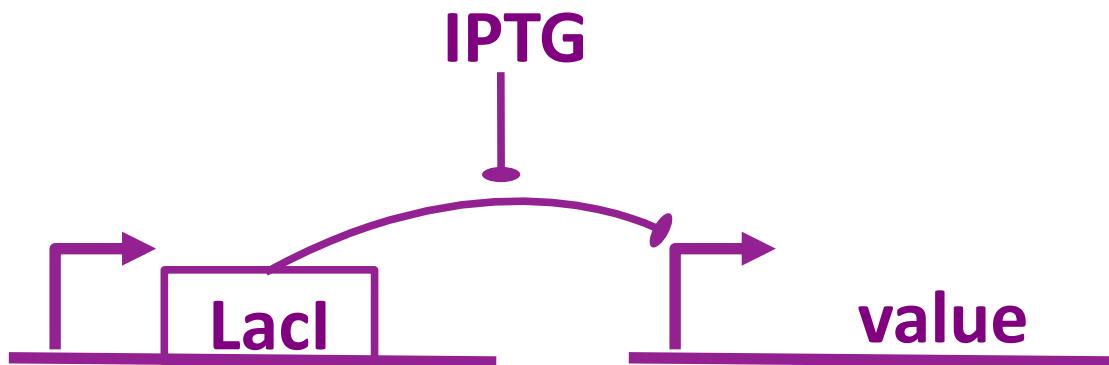
```
(primitive green (boolean) boolean :side-effect
  :type-constraints ((= value arg0))
  :grn-motif ((P R+ arg0 GFP|arg0 value T)))
```



Motif-Based Compilation

- High-level primitives map to GRN design motifs
 - e.g. logical operators, actuators, **sensors**:

```
(primitive IPTG () boolean
:grn-motif ((P high LacI|boolean T)
              (RXN (IPTG|boolean) represses LacI)
              (P high R- LacI value T)))
```



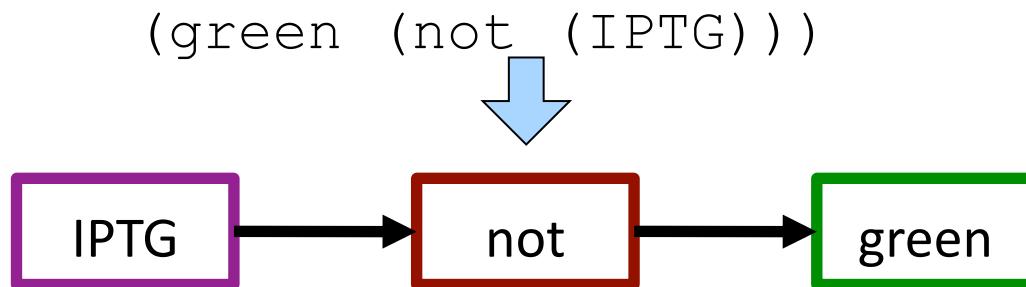
Motif-Based Compilation

- Functional program gives dataflow computation:

(green (not (IPTG)))

Motif-Based Compilation

- Functional program gives dataflow computation:



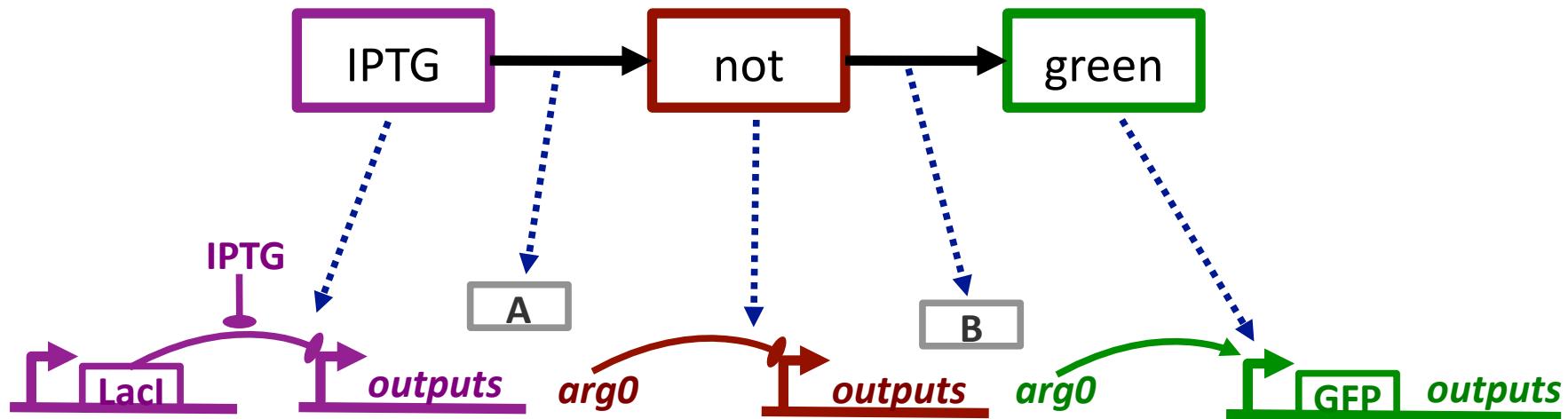
Motif-Based Compilation

- Operators translated to motifs:



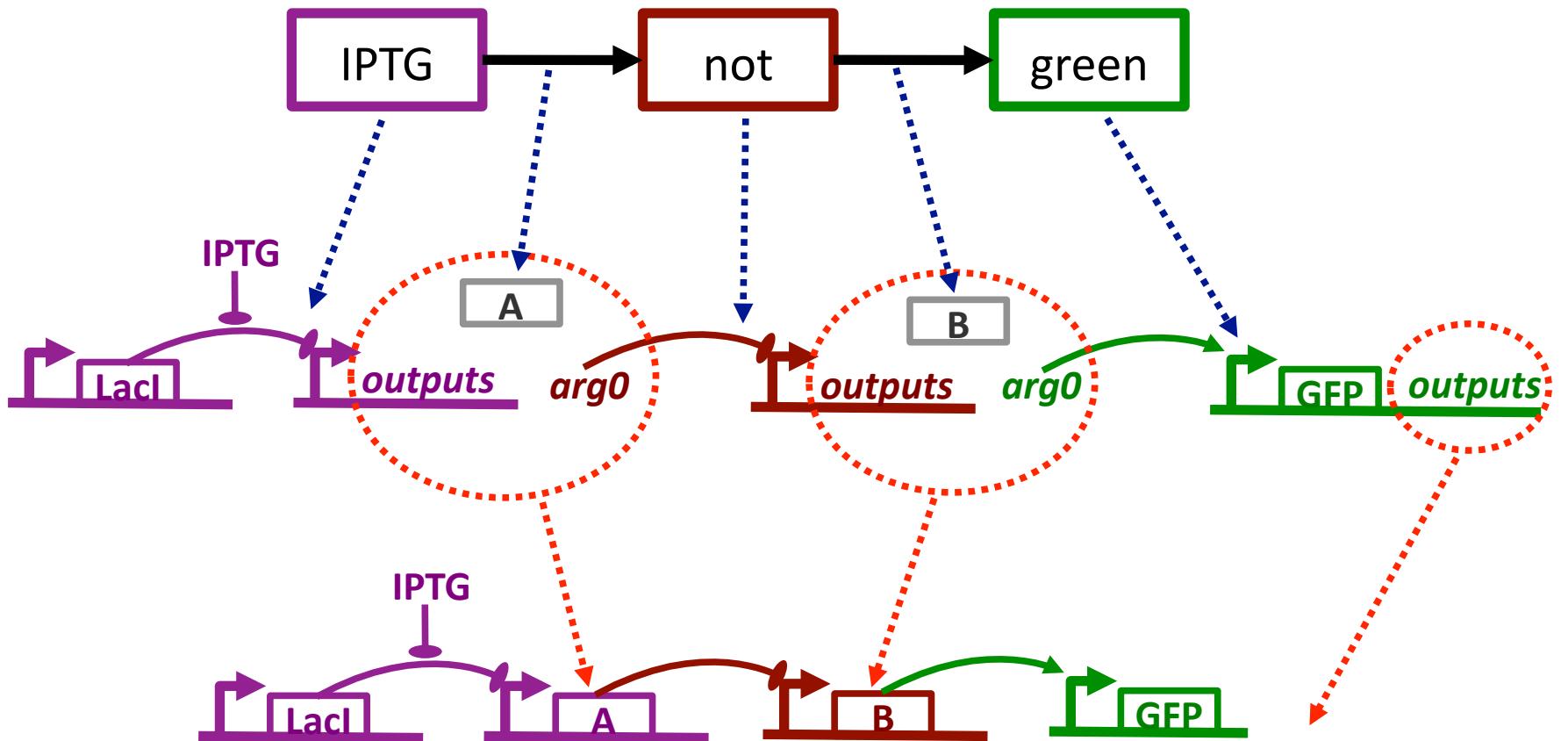
Motif-Based Compilation

- Operators translated to motifs:

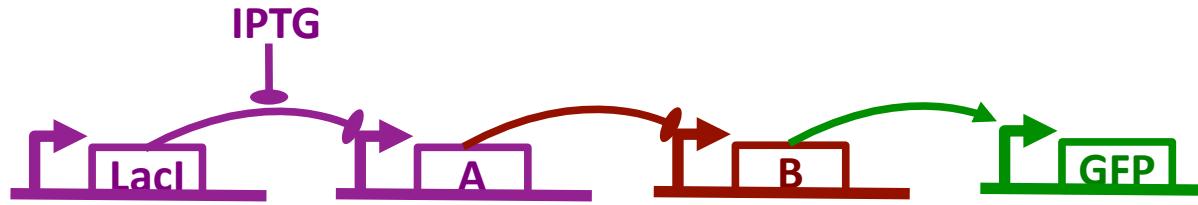


Motif-Based Compilation

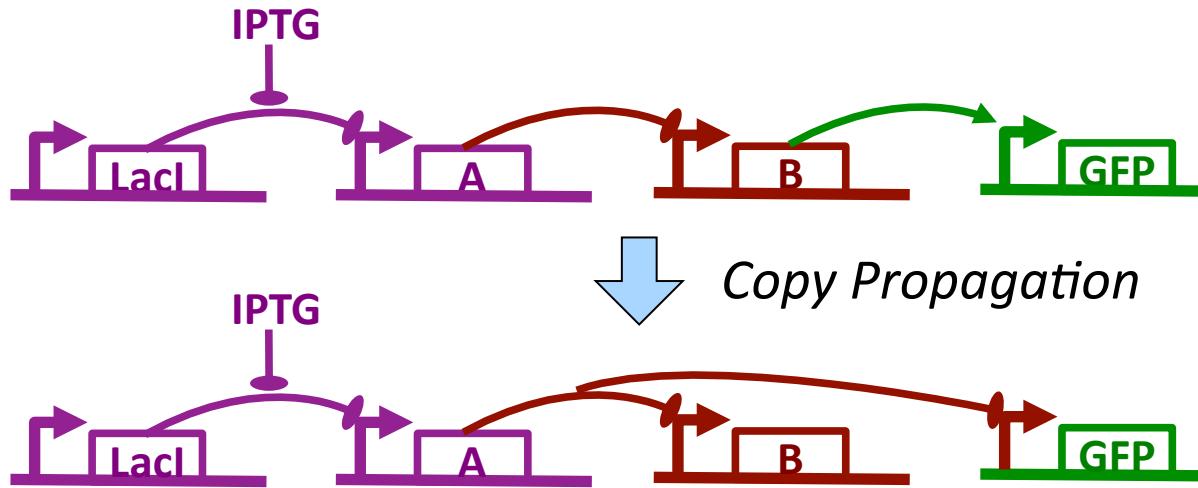
- Operators translated to motifs:



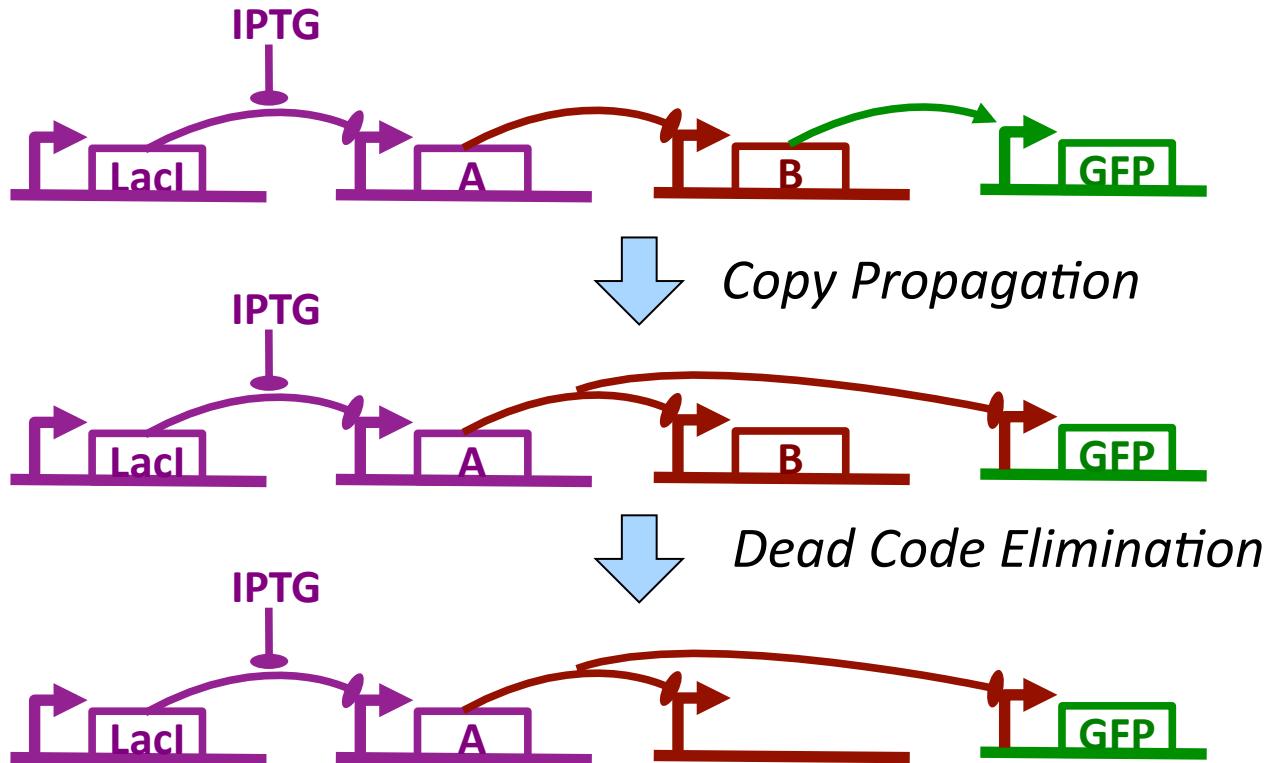
Optimization



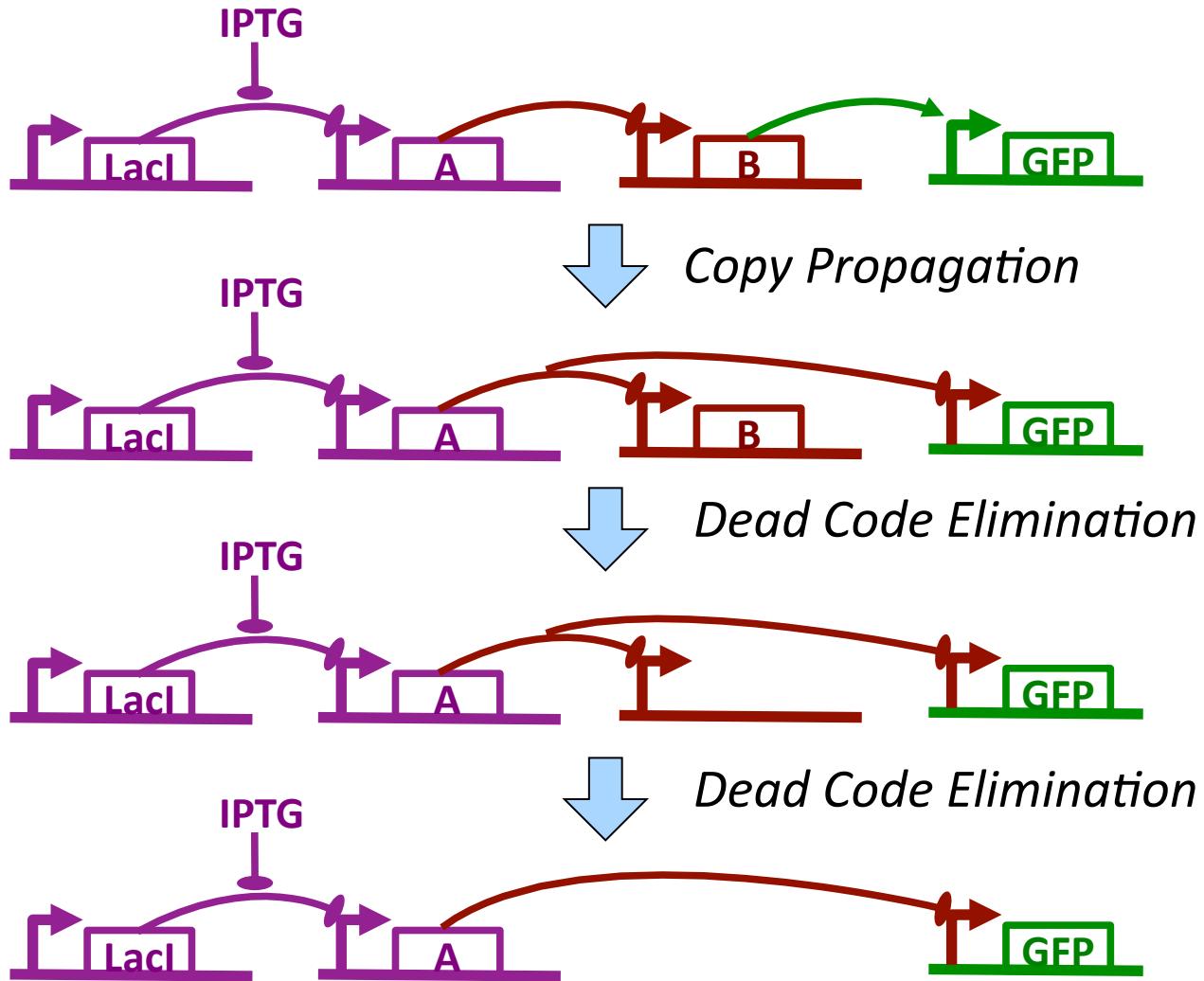
Optimization



Optimization



Optimization



Complex System: Feedback Latch

```
(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o))))))
  o))

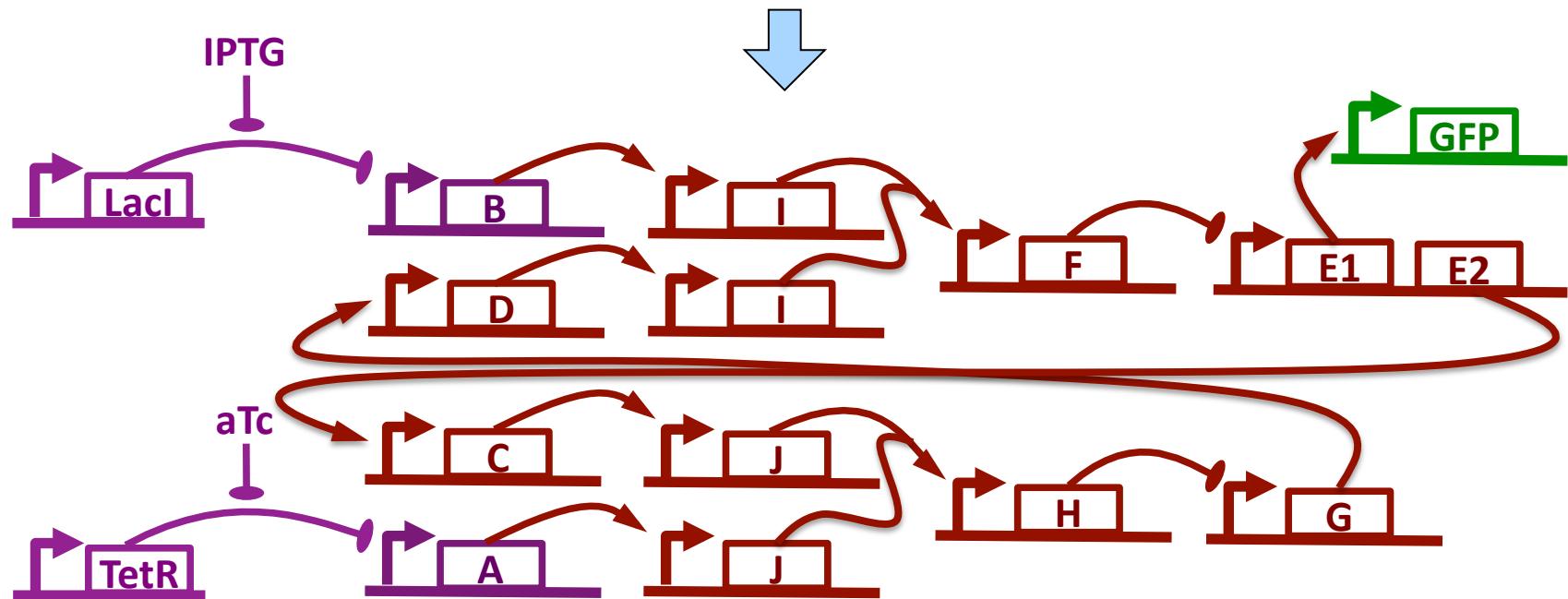
(green (sr-latch (aTc) (IPTG)) )
```

Complex System: Feedback Latch

```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))
(green (sr-latch (aTc) (IPTG)))
  
```



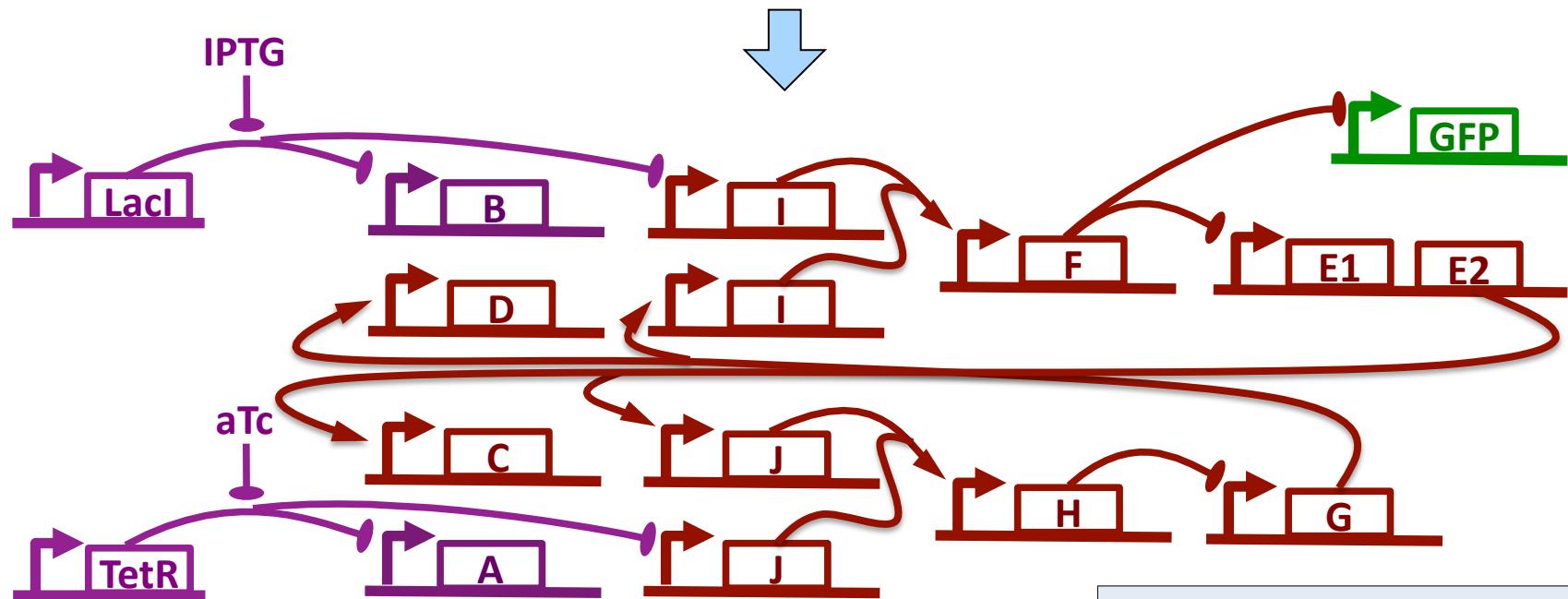
Unoptimized: 15 functional units, 13 transcription factors

Optimization of Complex Designs

```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))
(green (sr-latch (aTc) (IPTG)))
  
```



Unoptimized: 15 functional units, 13 transcription factors

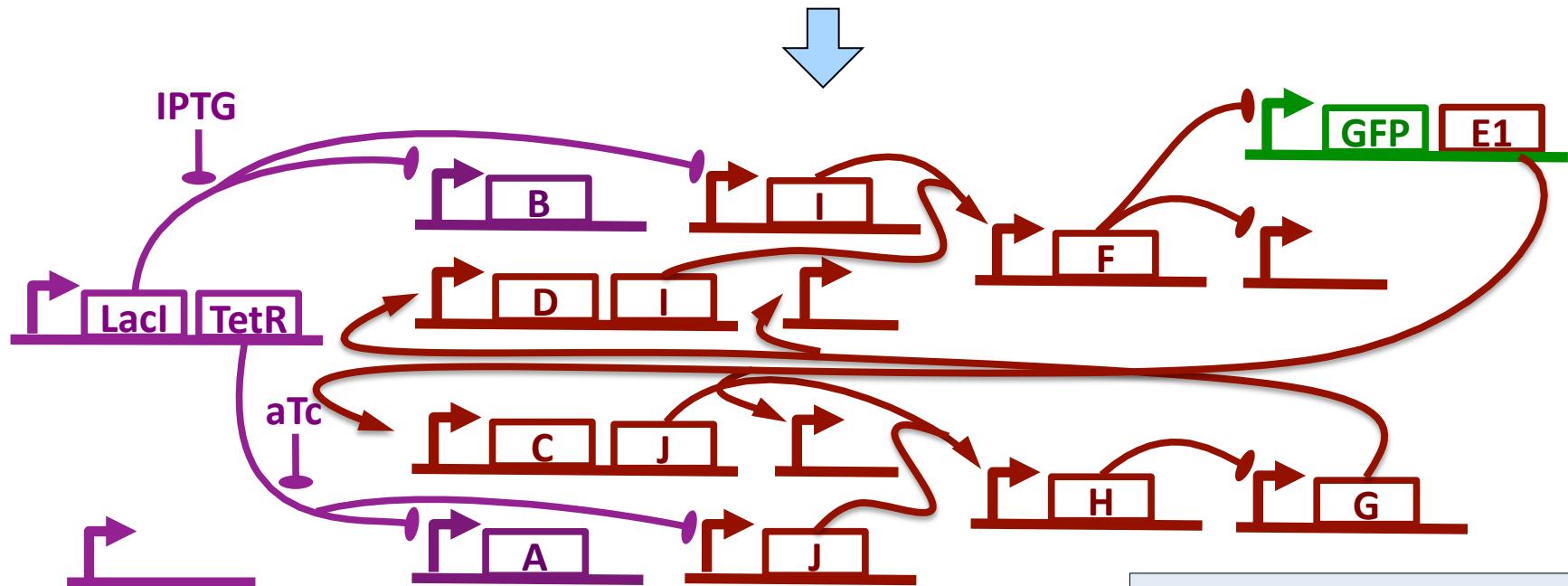
Copy Propagation

Optimization of Complex Designs

```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))
(green (sr-latch (aTc) (IPTG)))
  
```



Unoptimized: 15 functional units, 13 transcription factors

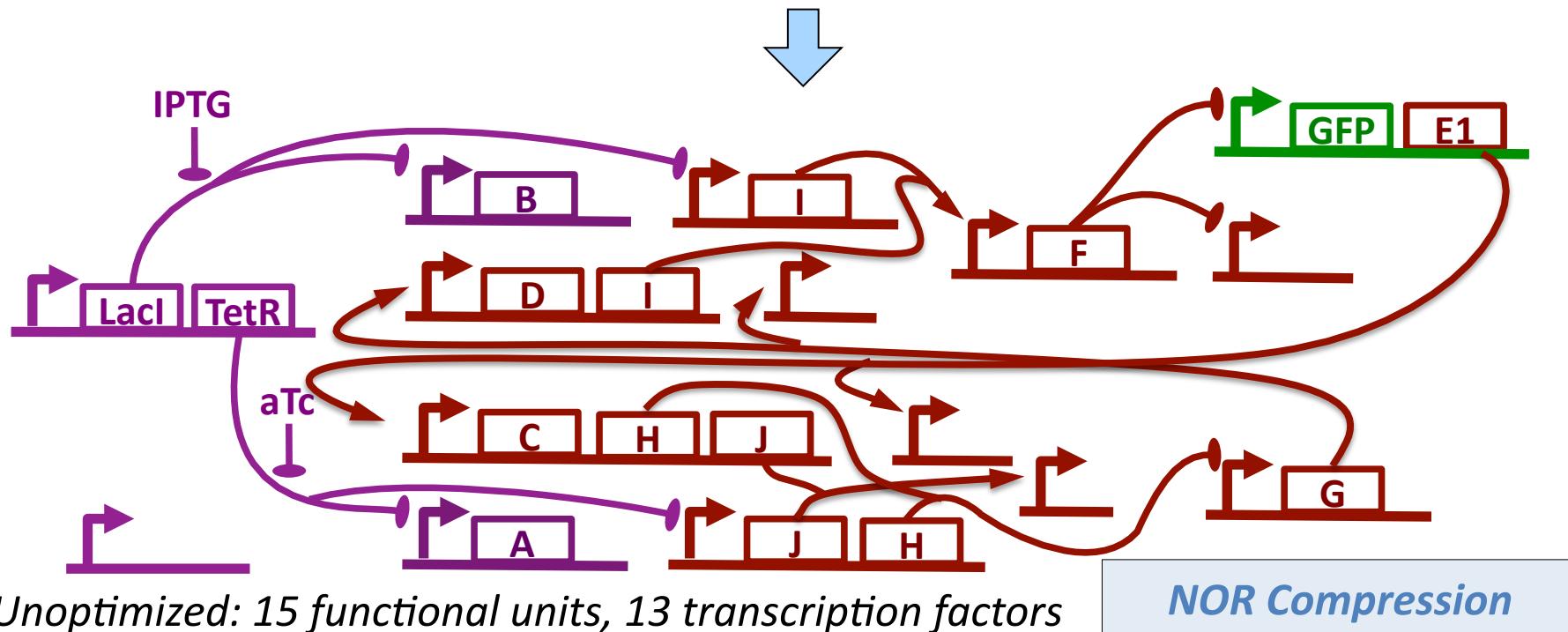
Common Subexp. Elim.

Optimization of Complex Designs

```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))
(green (sr-latch (aTc) (IPTG)))
  
```

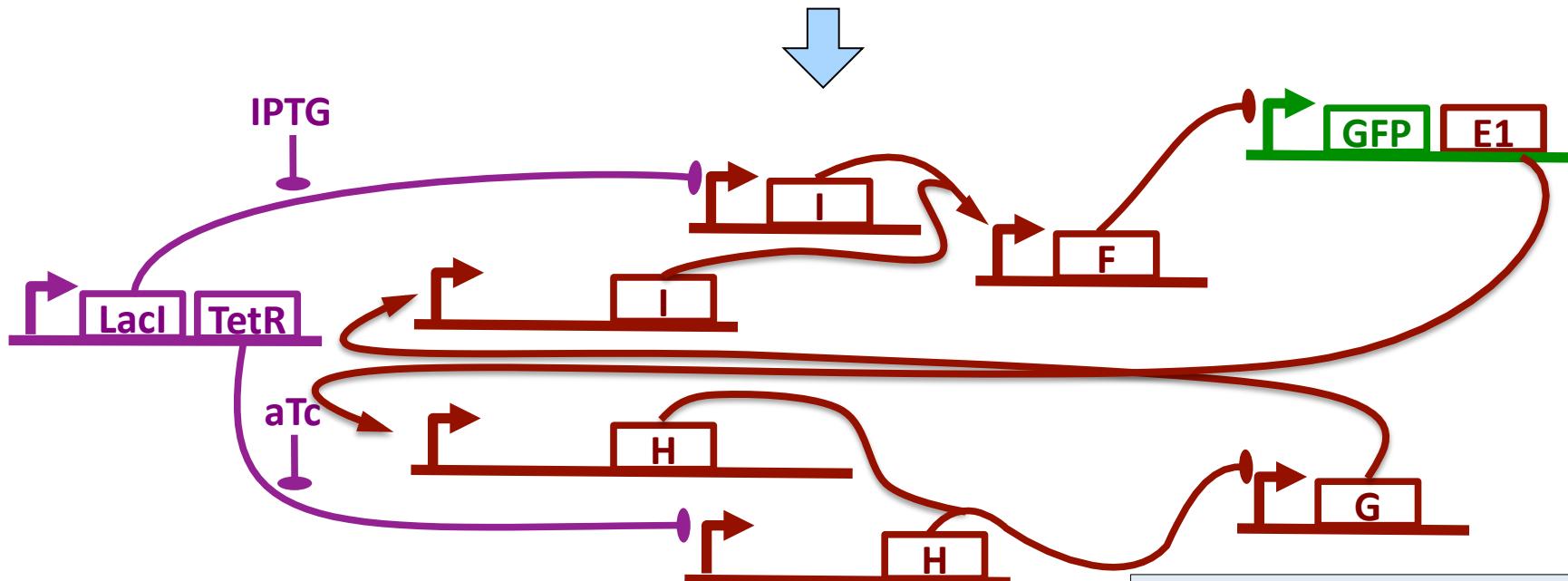


Optimization of Complex Designs

```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))
(green (sr-latch (aTc) (IPTG)))
  
```



Unoptimized: 15 functional units, 13 transcription factors

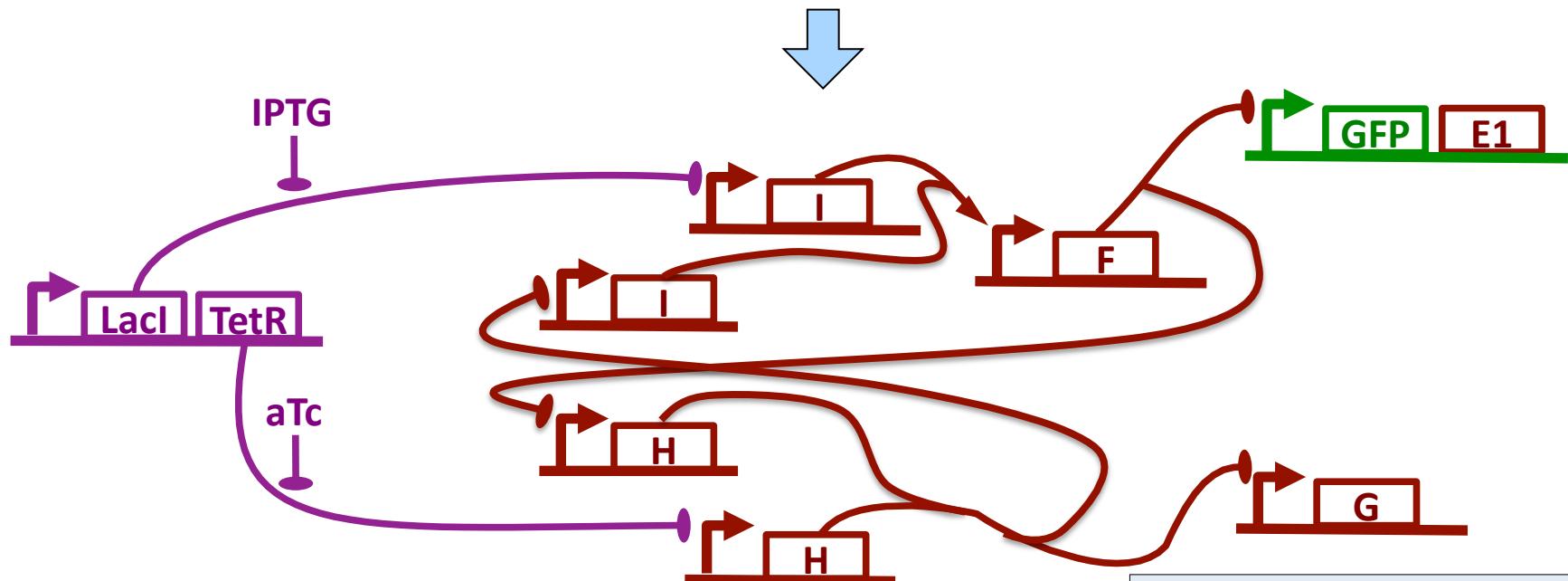
Dead Code Elimination

Optimization of Complex Designs

```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))
(green (sr-latch (aTc) (IPTG)))
  
```



Unoptimized: 15 functional units, 13 transcription factors

Copy Propagation

Optimization of Complex Designs

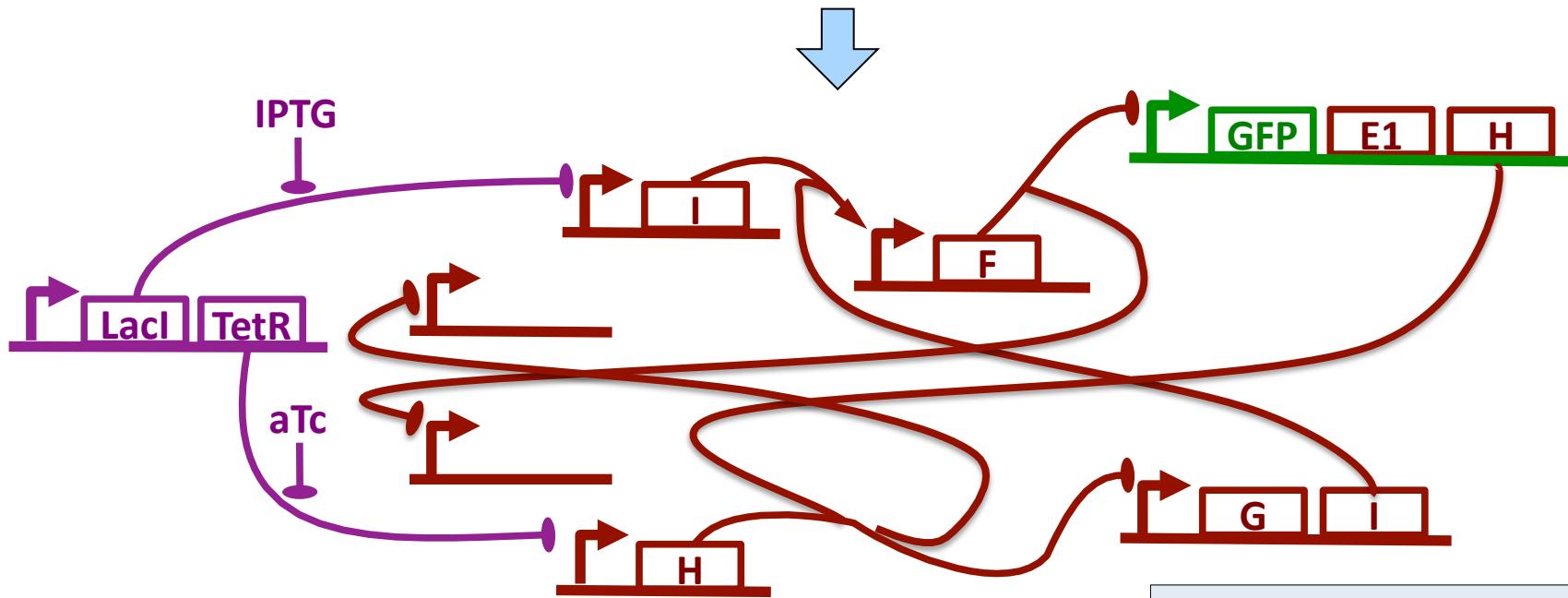
```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))

```

(**green** (**sr-latch** (**aTc**) (**IPTG**)))



Unoptimized: 15 functional units, 13 transcription factors

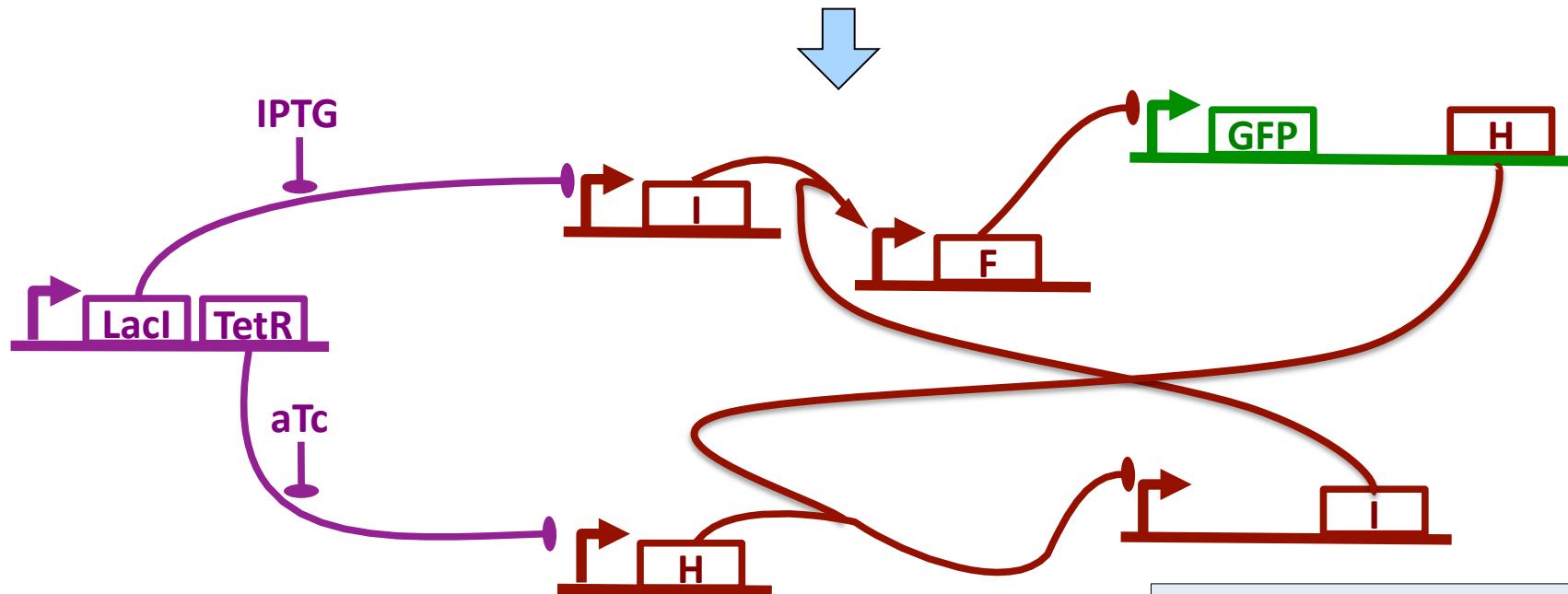
Common Subexp. Elim.

Optimization of Complex Designs

```
(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))

(green (sr-latch (aTc) (IPTG)))
```



Unoptimized: 15 functional units, 13 transcription factors

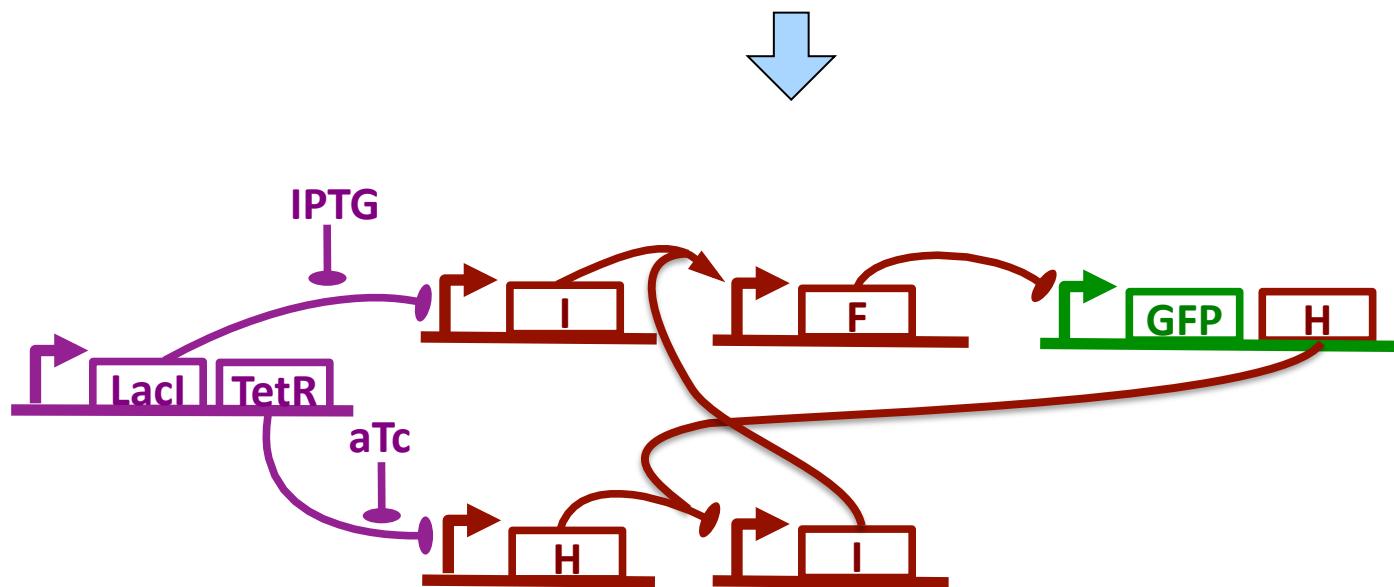
Dead Code Elimination

Optimization of Complex Designs

```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))
(green (sr-latch (aTc) (IPTG)))
  
```



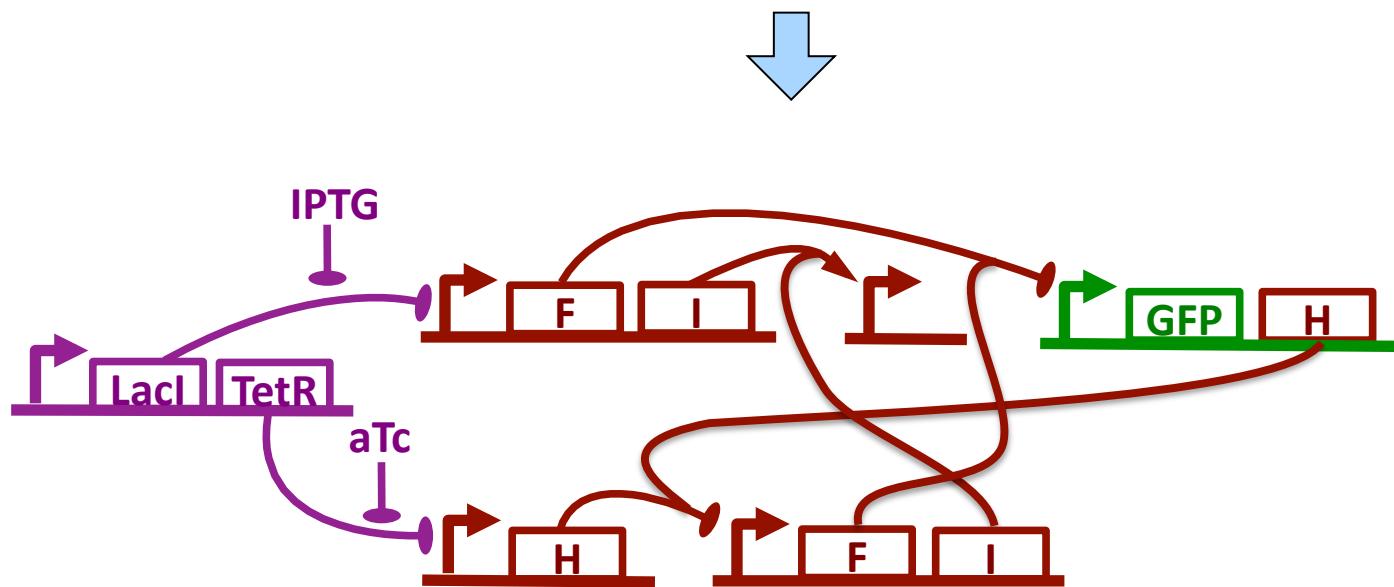
Unoptimized: 15 functional units, 13 transcription factors

Optimization of Complex Designs

```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))
(green (sr-latch (aTc) (IPTG)))
  
```



Unoptimized: 15 functional units, 13 transcription factors

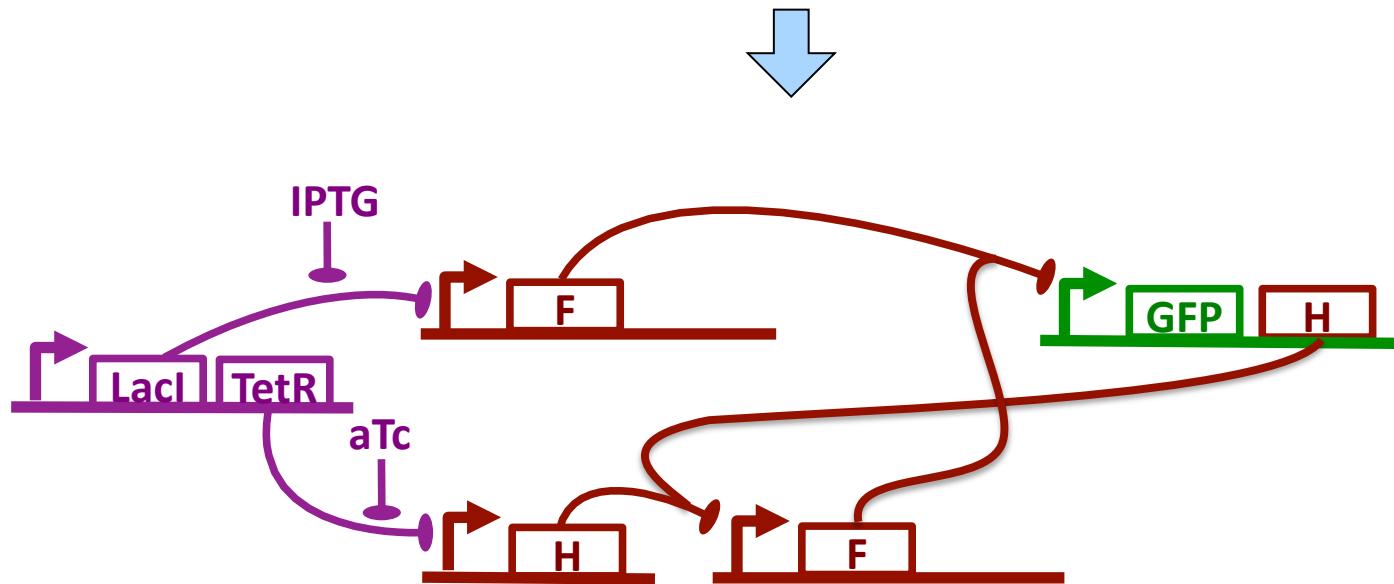
NOR Compression

Optimization of Complex Designs

```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))
(green (sr-latch (aTc) (IPTG)))
  
```



Unoptimized: 15 functional units, 13 transcription factors

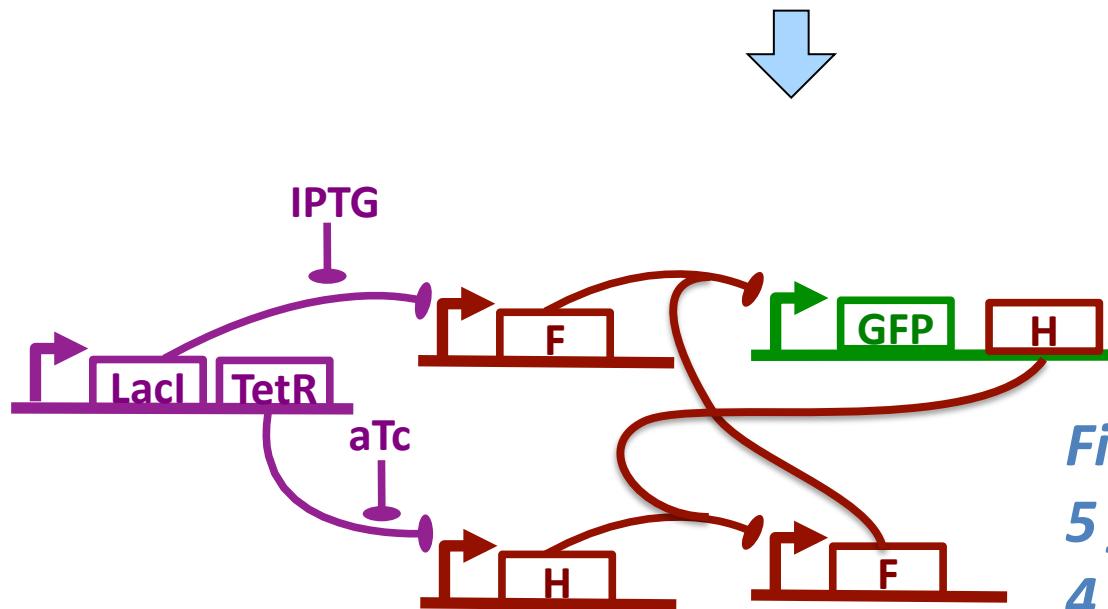
Dead Code Elimination

Optimization of Complex Designs

```

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
            (o-bar boolean (not (or s o)))))

  o))
(green (sr-latch (aTc) (IPTG)))
  
```



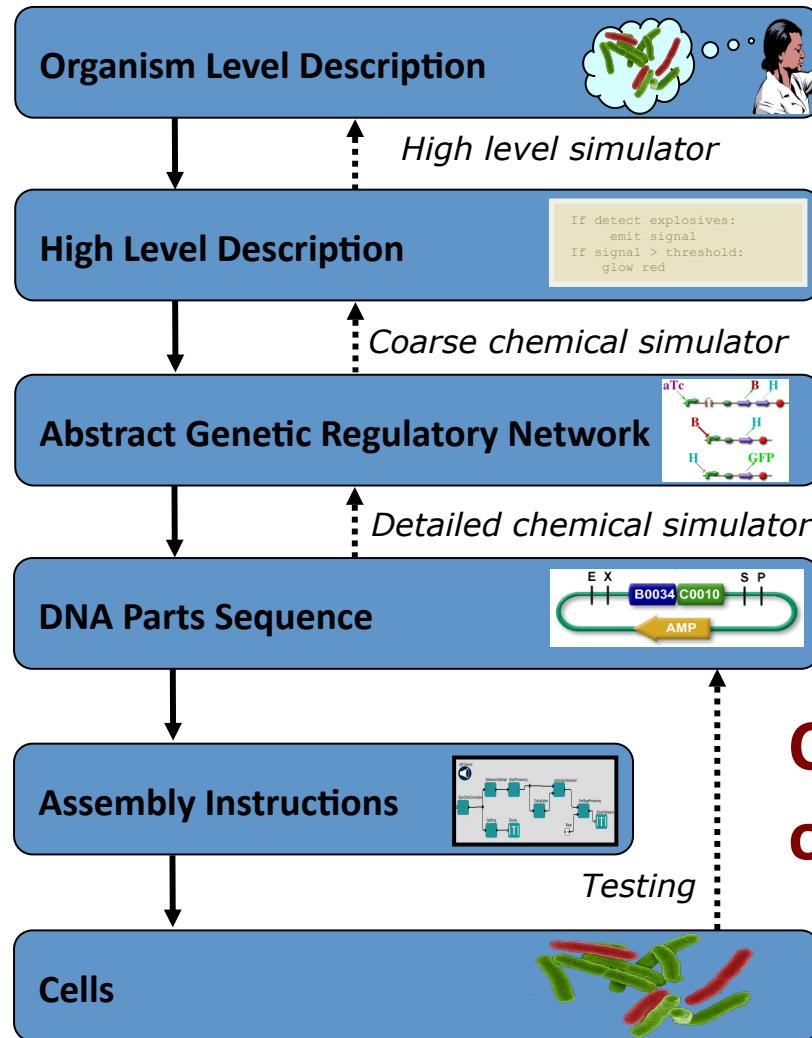
*Final Optimized:
5 functional units
4 transcription factors*

Unoptimized: 15 functional units, 13 transcription factors

Compilation & Optimization Results:

- Automated GRN design for arbitrary boolean logic and feedback systems
- Optimization competitive with human experts:
 - Test systems have 25% to 71% complexity reduction
 - Optimized systems homologous with hand design

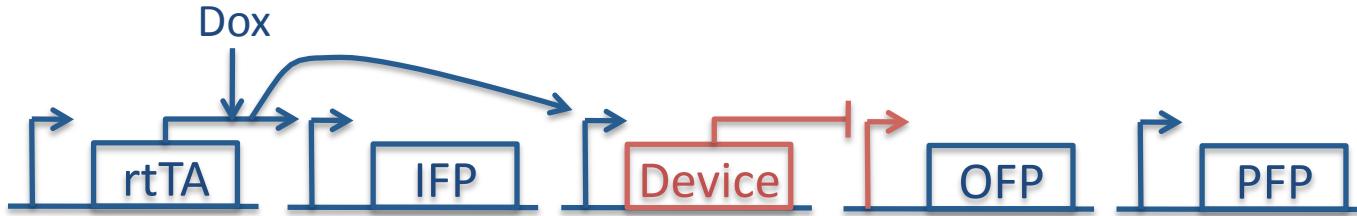
Advances on Two Key Problems:



**Characterization
of Transfer Curves**

Key Problem: Device Characterization

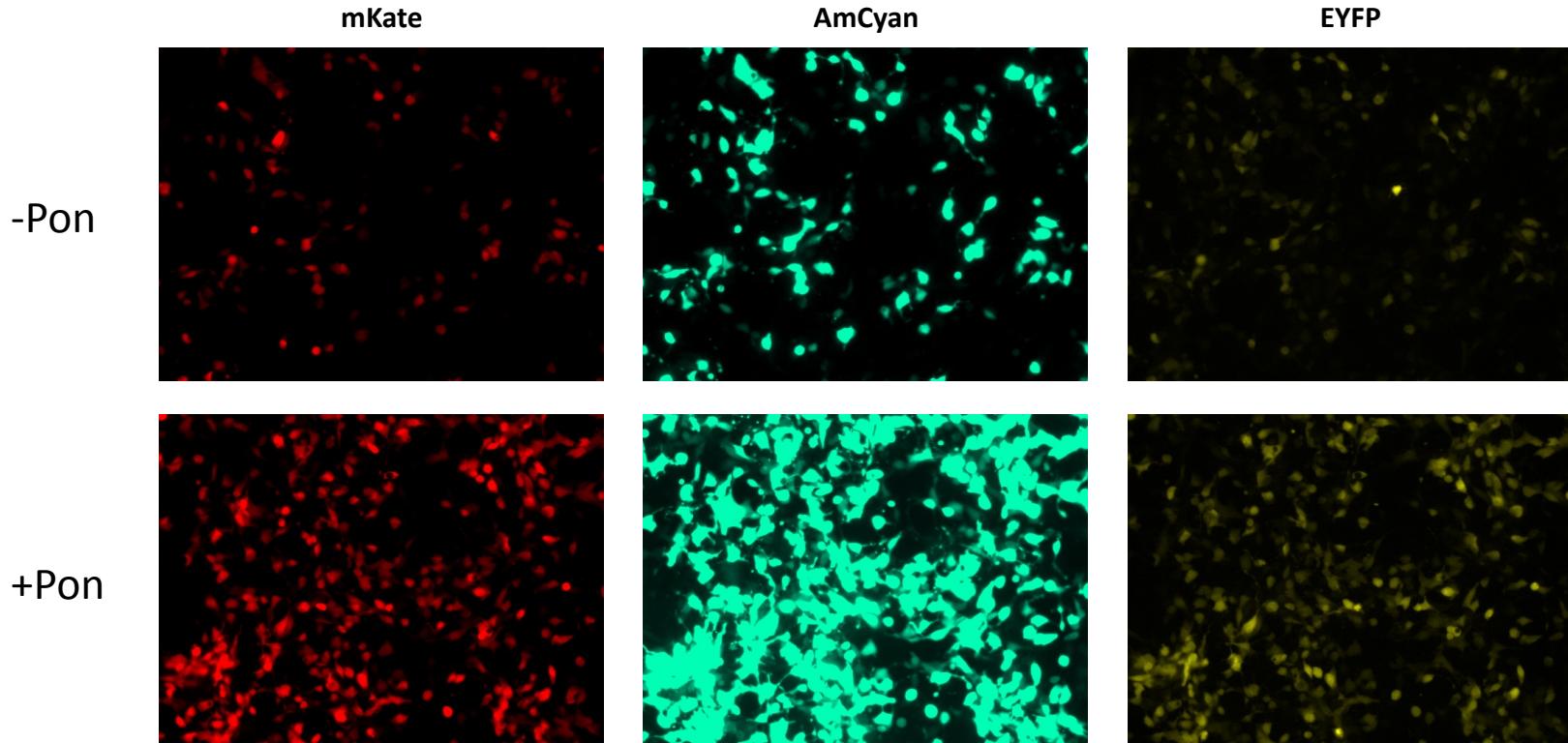
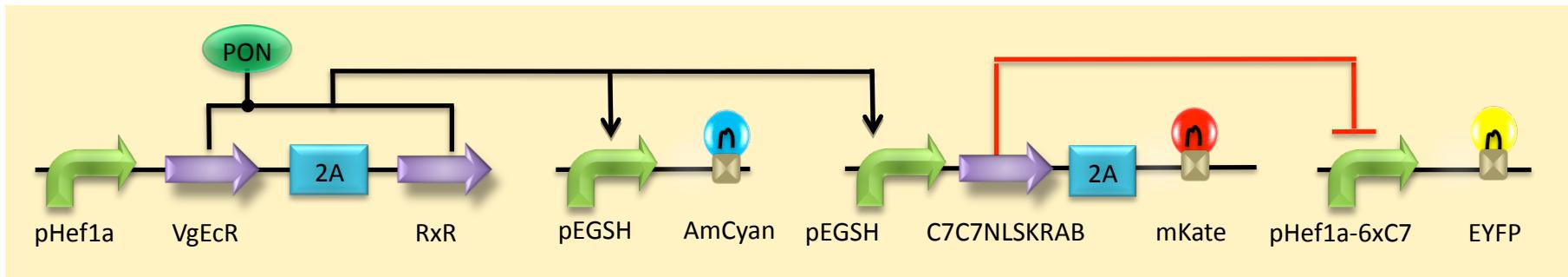
Goal: quantify single-cell I/O concentration relation



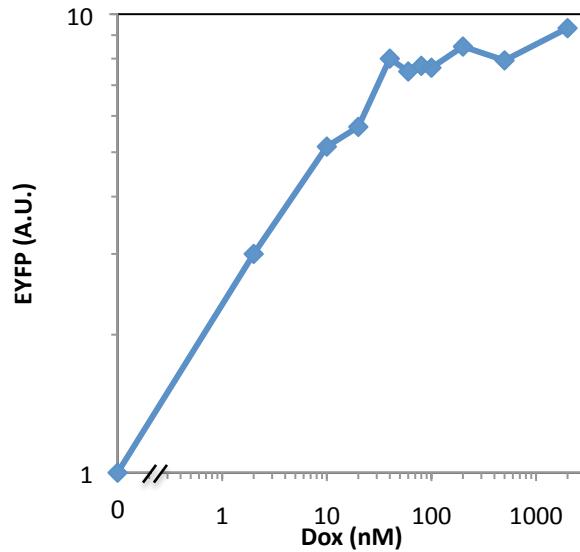
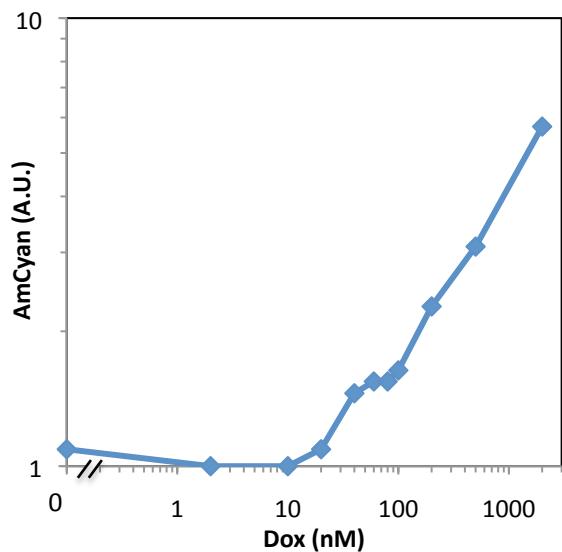
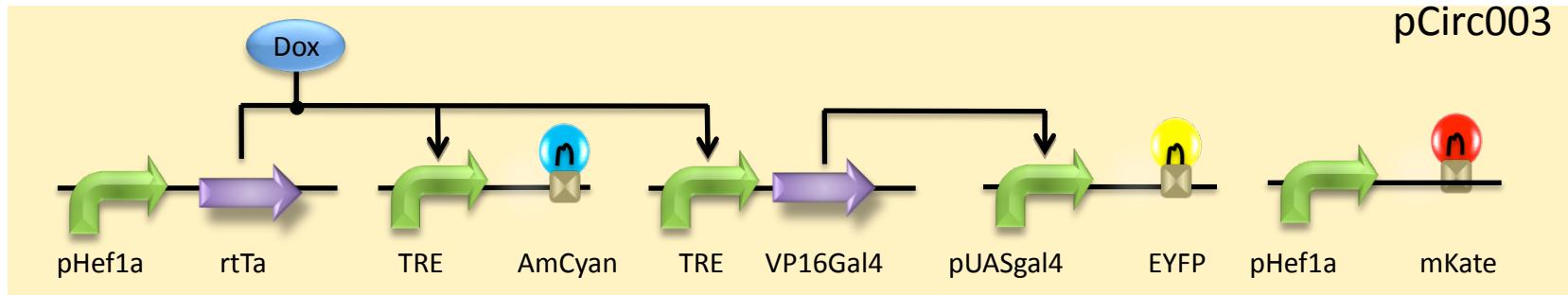
Three phases of lab work:

- Multi-plasmid (qualitative test)
- Single-plasmid (rough quantitative)
- Chromosomal integration (fine quantitative)

Example System: C7C7 repressor

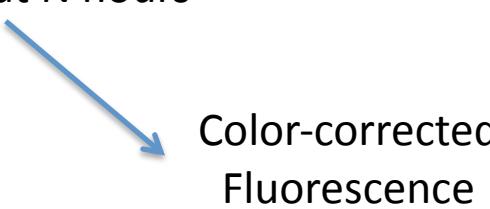


Example System: VP16Gal4 activator

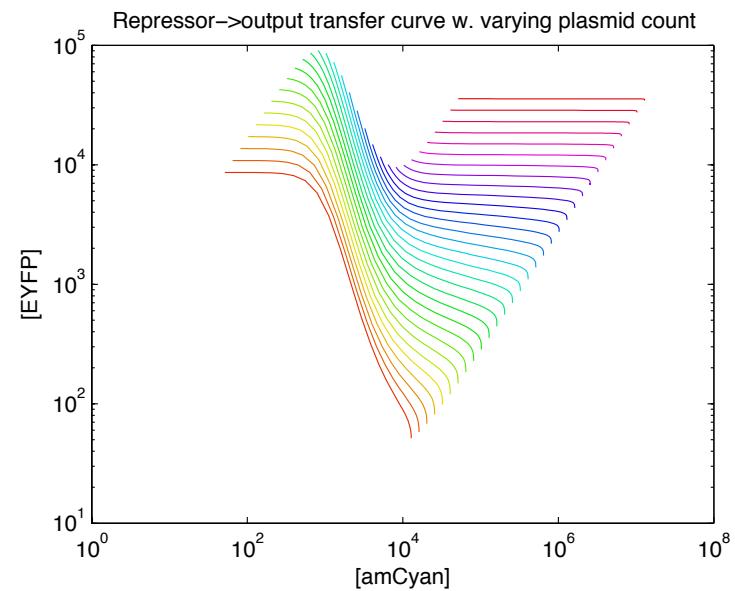
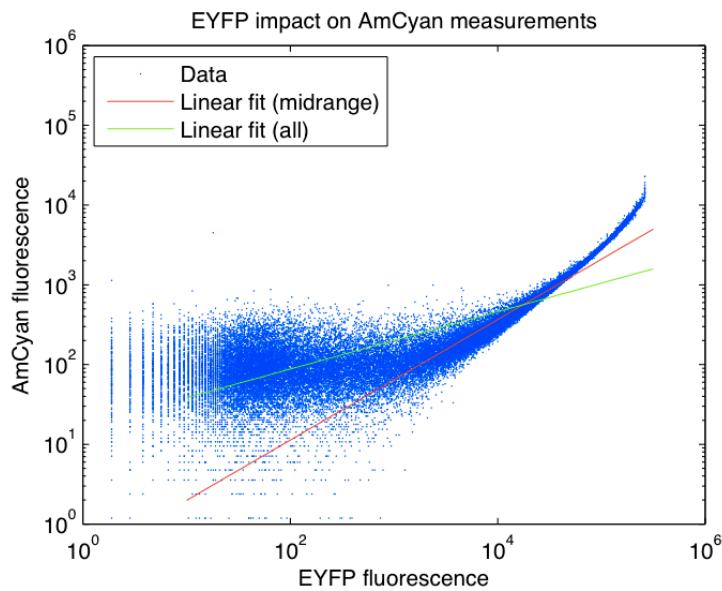


From Fluorescence to Static Discipline

Fluorescence of proxy proteins at N hours



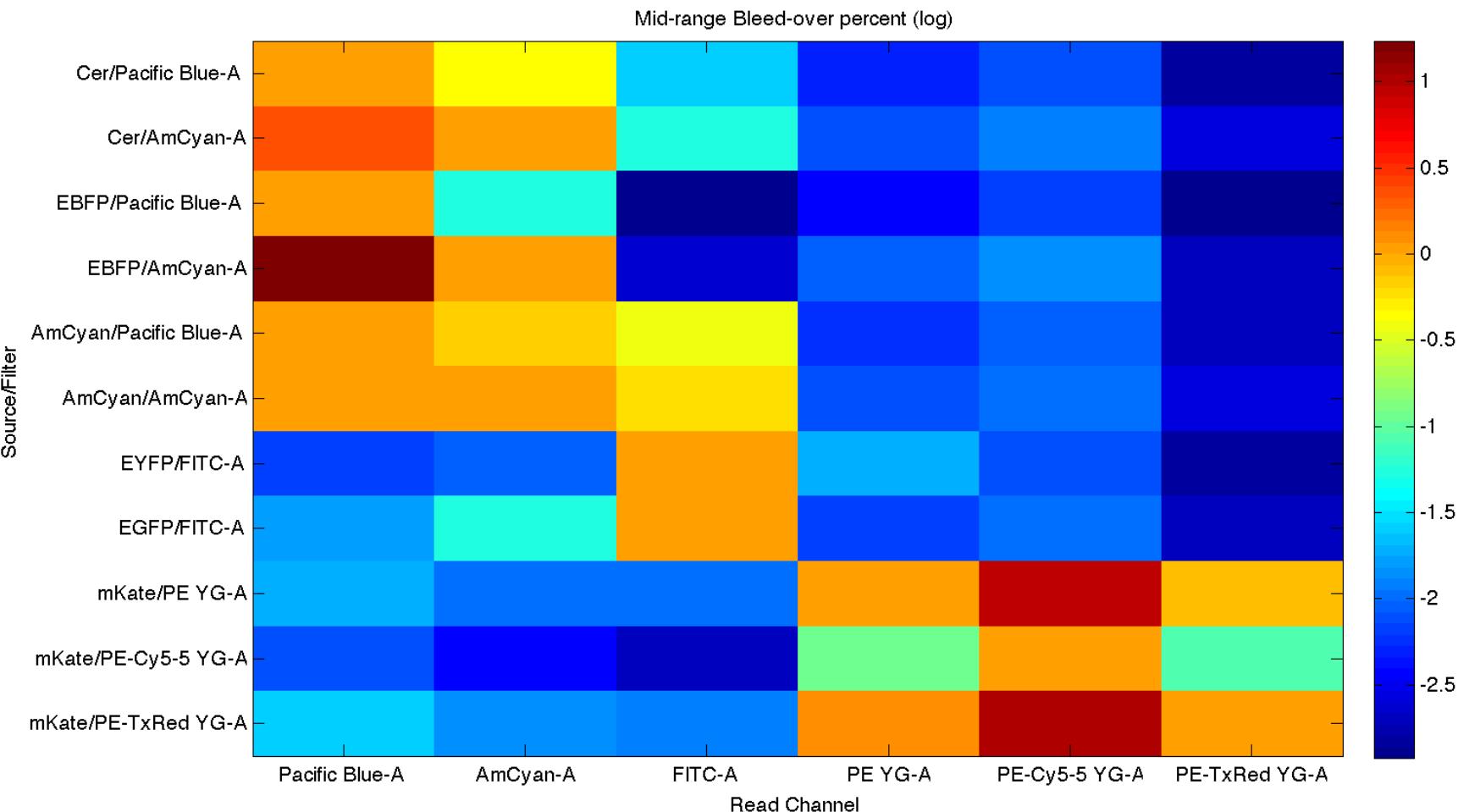
Model-Compensated Transfer Curve



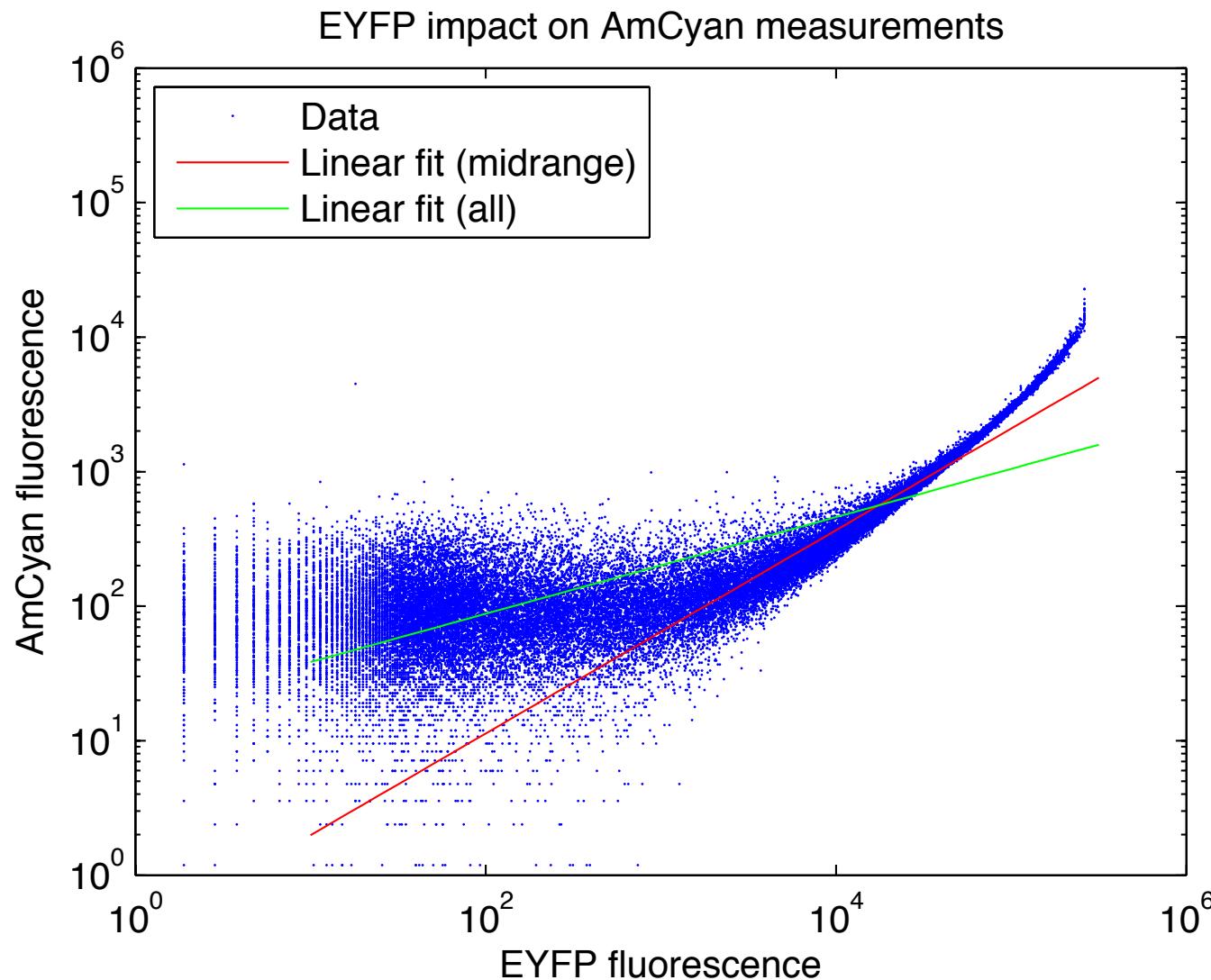
Inflection Points

Color Correction: Bleed-over Matrix

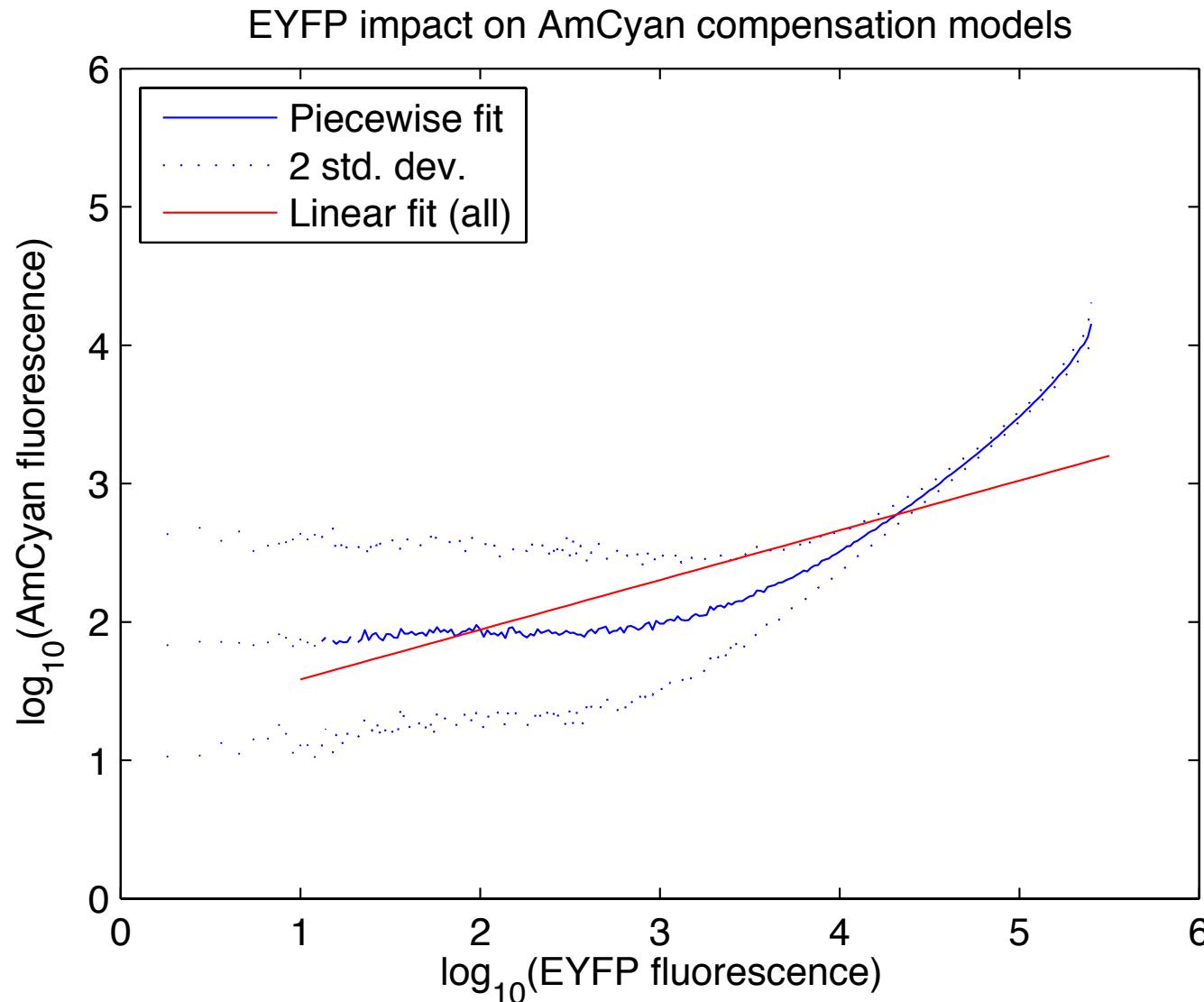
- Selecting an optimal combination of proteins:



Color Correction: Piecewise Models



Color Correction: Piecewise Models



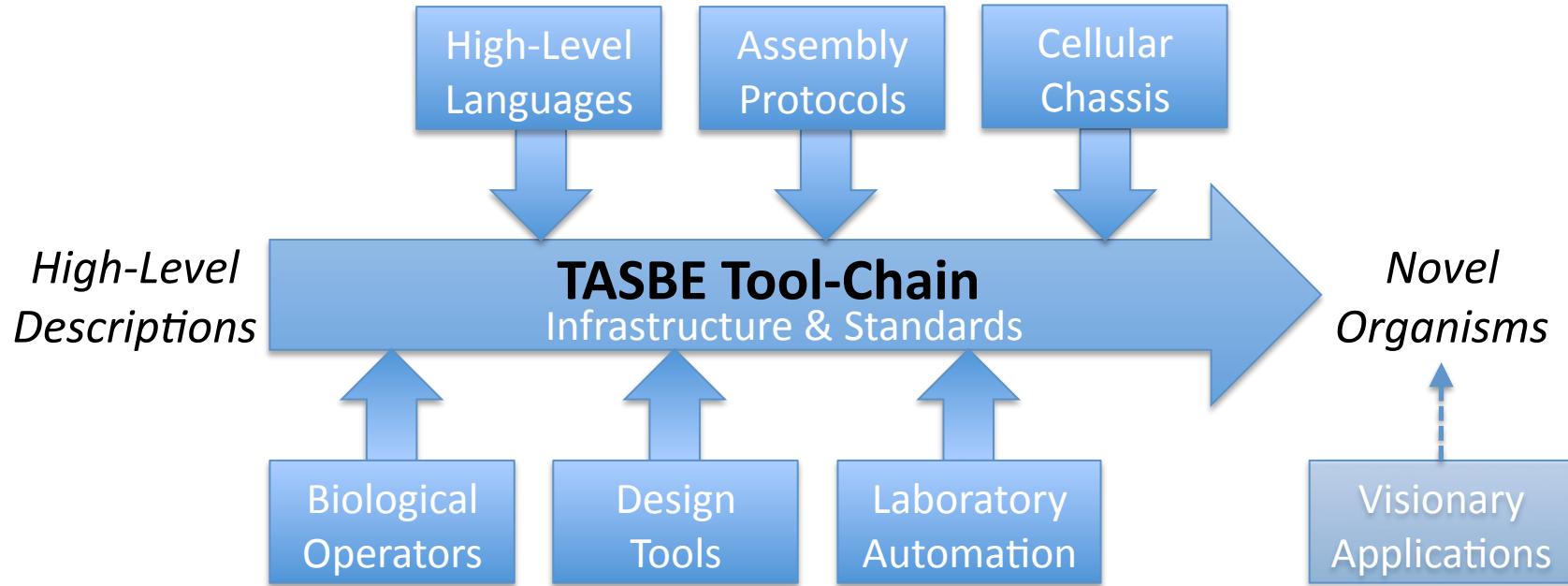
Characterization Contributions

- Multi-stage characterization protocol for transcriptional devices
- Model mapping multi-plasmid behavior to predictions of single copy behavior
- Improved FACS color correction
- Preliminary characterization results

Contributions:

- TASBE: open tool-chain architecture
- Demonstration of end-to-end automated design
- Advances on key sub-problems:
 - Compilation and Optimization
 - DNA Part Selection *[Next talk]*
 - Flexible Protocol Automation *[Following talk]*
 - Characterization of Transfer Curves

Toward a community platform...



- Free, open source core
 - Proto, Clotho available now, others by arrangement
- Work on interchange standards (SBOL, CHRIS)