

# Toward breaking the complexity barrier for synthetic biology therapeutics

Jacob Beal

IEEE EMBC  
August, 2011

Work sponsored by DARPA I2O under contract HR0011-10-C-0168; the views and conclusions contained in this document are those of the authors and not DARPA or the U.S. Government.

**Raytheon**  
**BBN Technologies**

# Project Team:

---

**Raytheon  
BBN Technologies**

Jacob Beal (PI)

Joseph Loyall (PM)

Aaron Adler

Fusun Yaman

Rick Schantz



Ron Weiss (co-PI)

Jonathan Babb

Noah Davidsohn



Douglas Densmore (co-PI)

Swapnil Bhatia

Traci Haddock

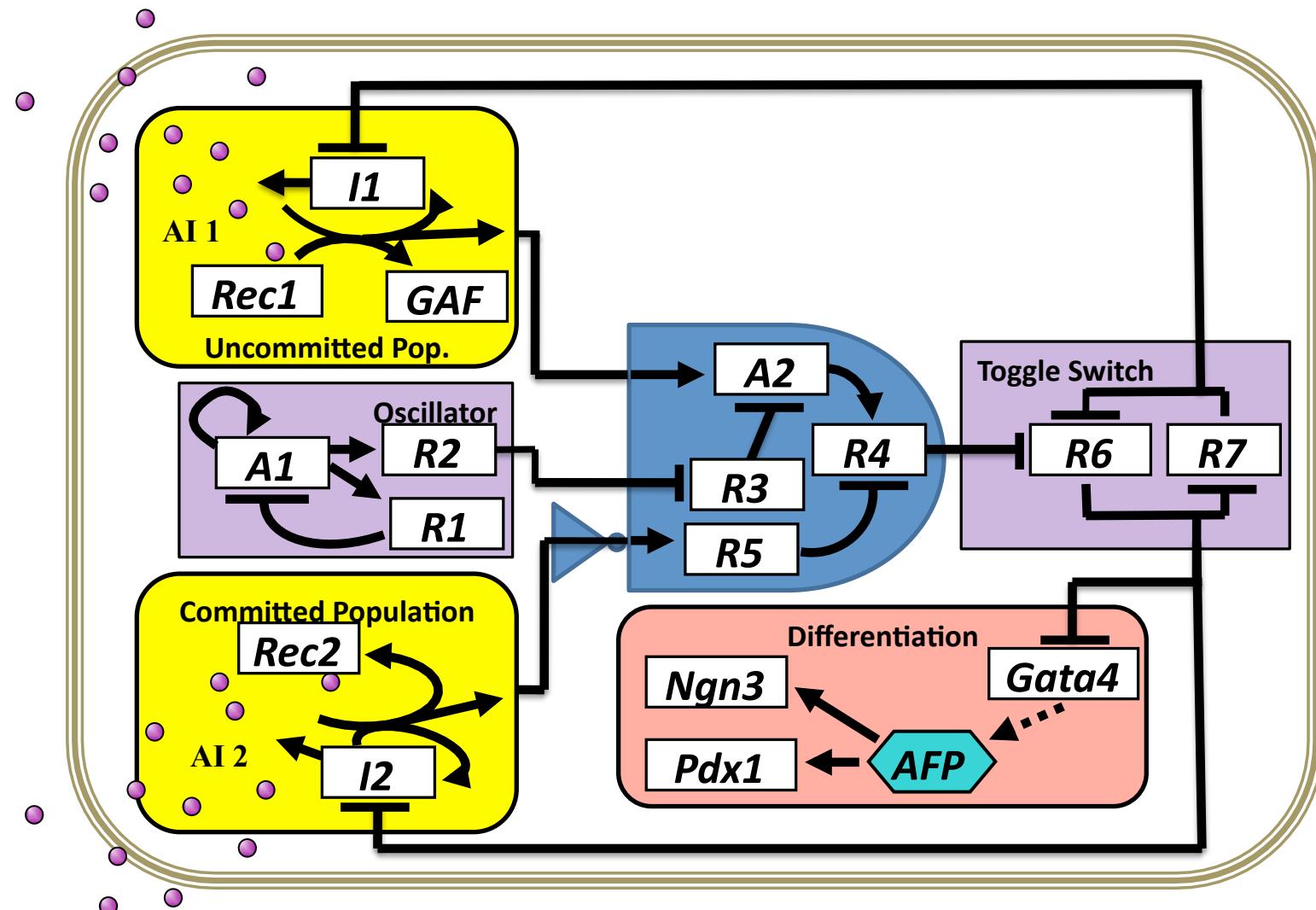
Viktor Vasilev

Chenkai Liu



Sponsored by:

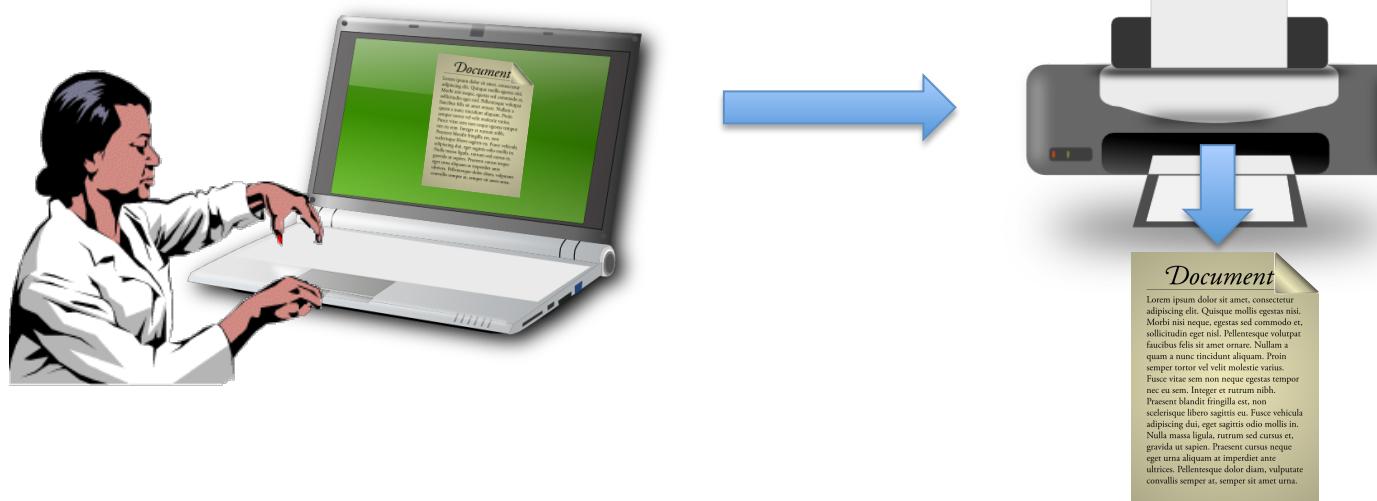
# Therapeutic systems will be complex:



[Weiss lab design for artificial tissue homeostasis]

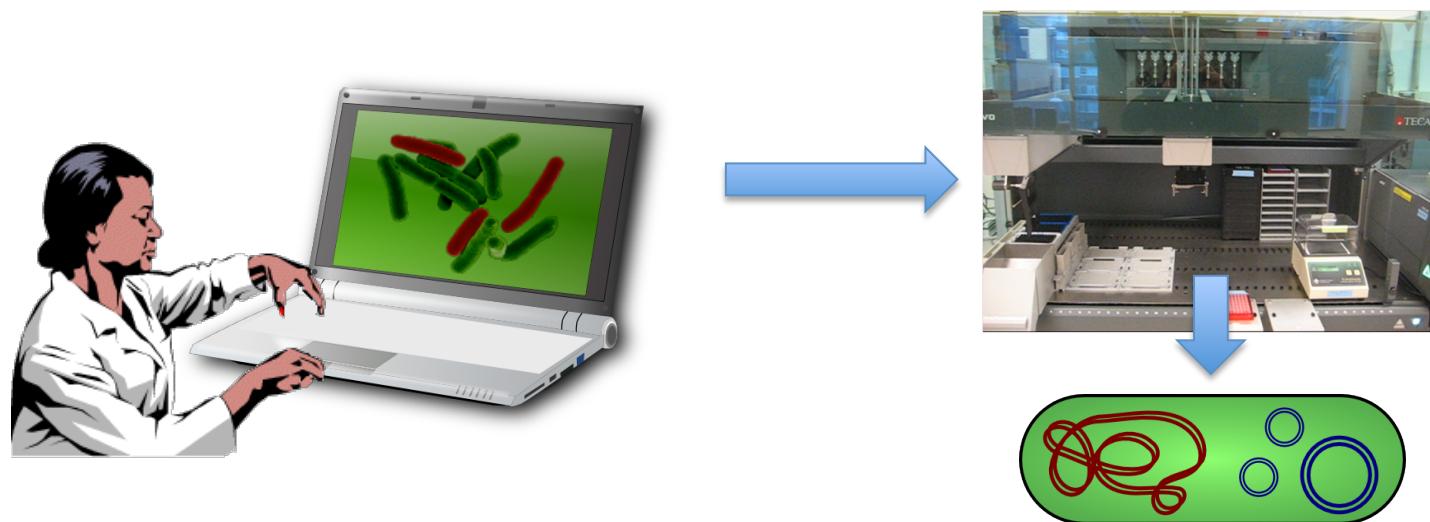
# Vision: WYSIWYG Synthetic Biology

Bioengineering should be like document preparation:



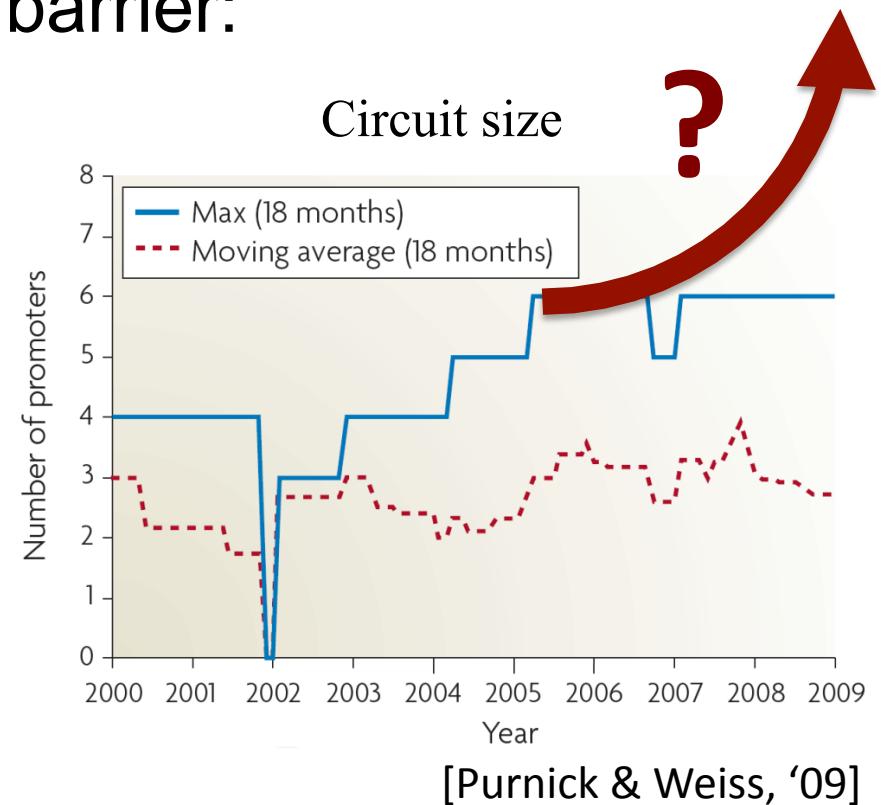
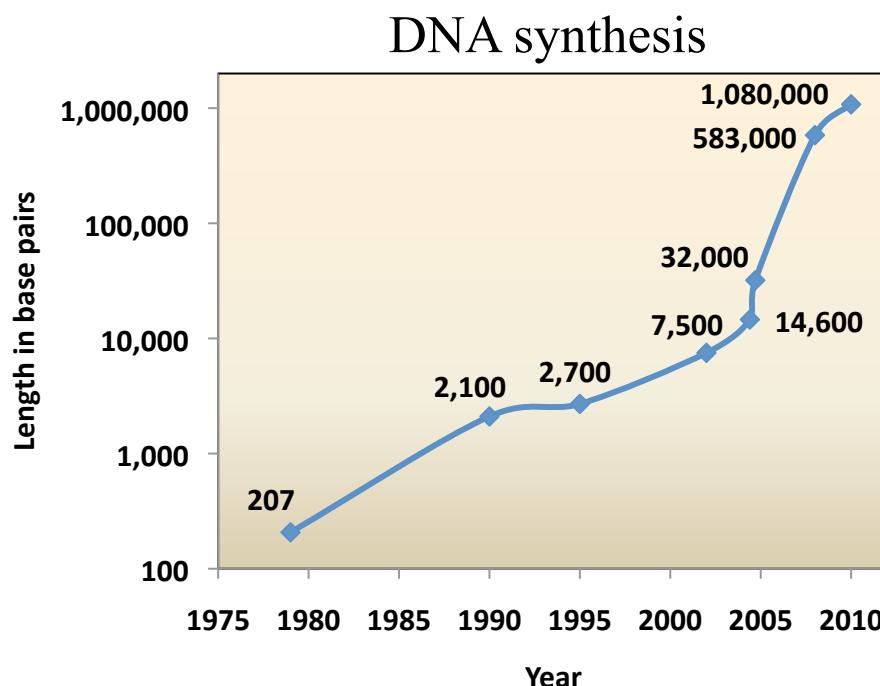
# Vision: WYSIWYG Synthetic Biology

Bioengineering should be like document preparation:



# Why is this important?

- Breaking the complexity barrier:



- Multiplication of research impact
- Reduction of barriers to entry

\*Sampling of systems in publications with experimental circuits

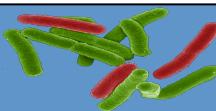
# Why a tool-chain?

Organism Level Description

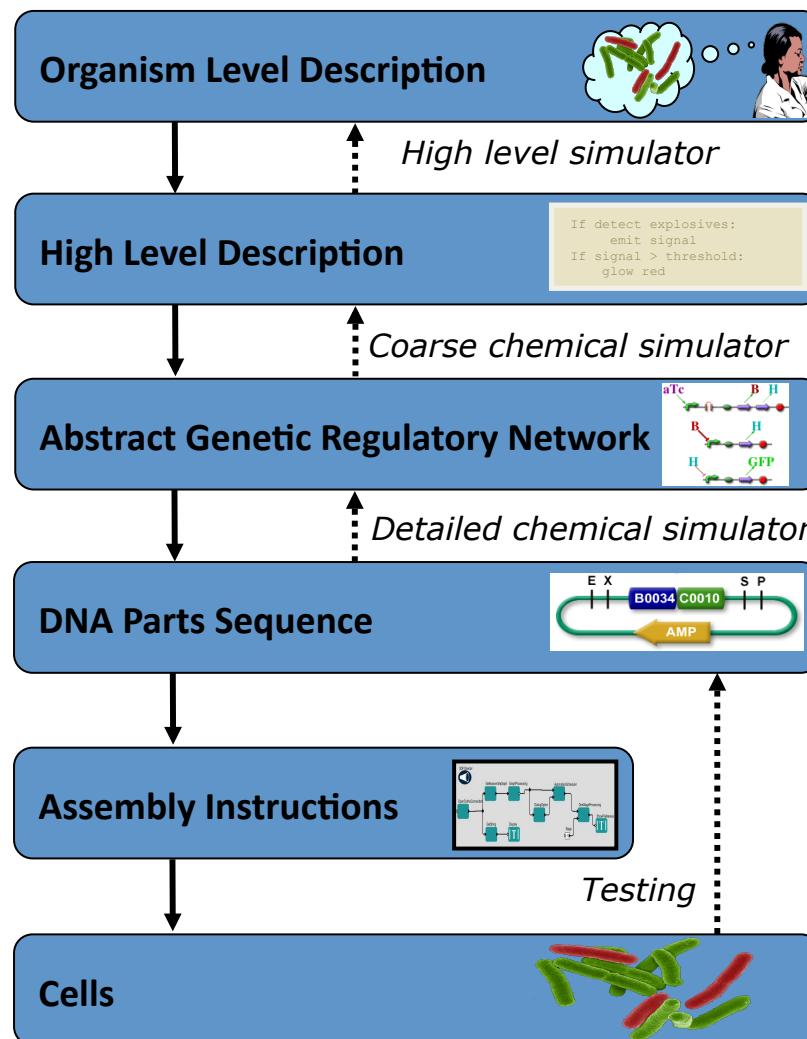


*This gap is too big  
to cross with a  
single method!*

Cells



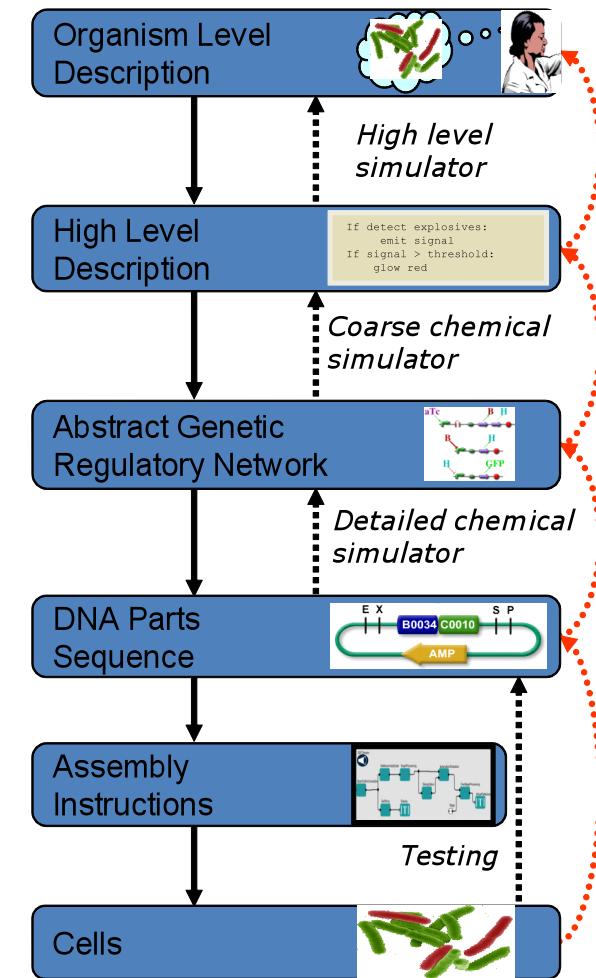
# The TASBE architecture:



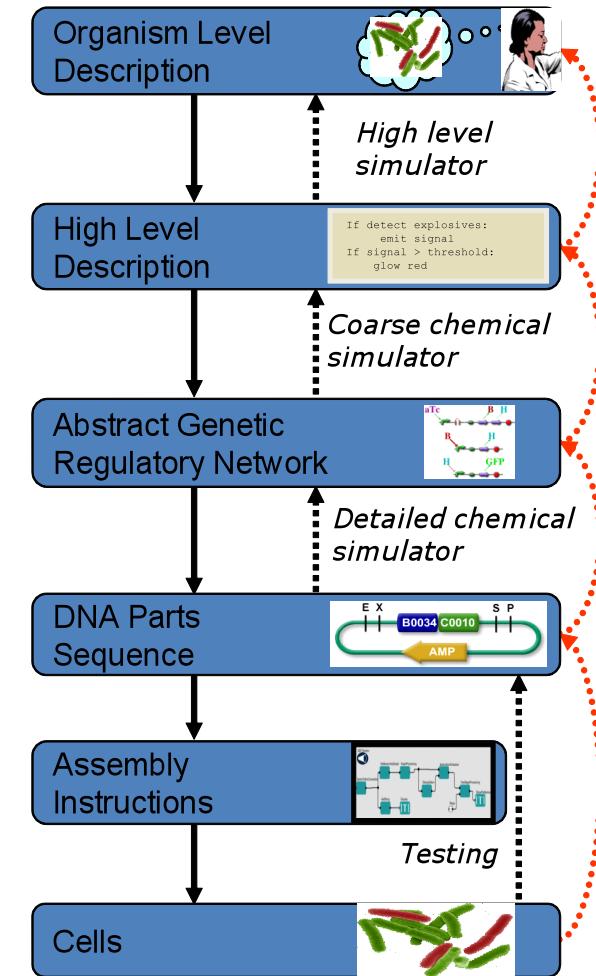
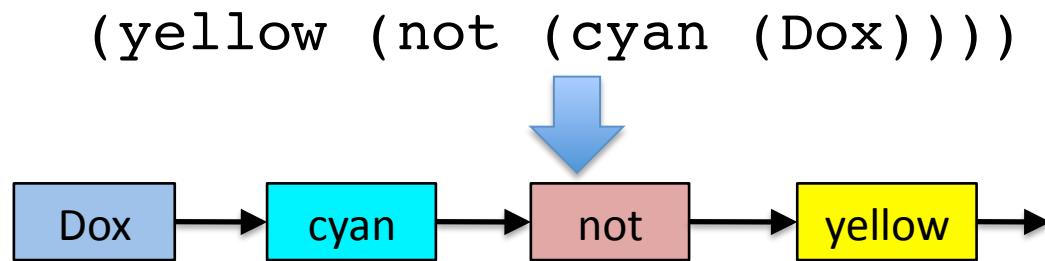
*Modular architecture  
also open for flexible  
choice of organisms,  
protocols, methods, ...*

# A Tool-Chain Example

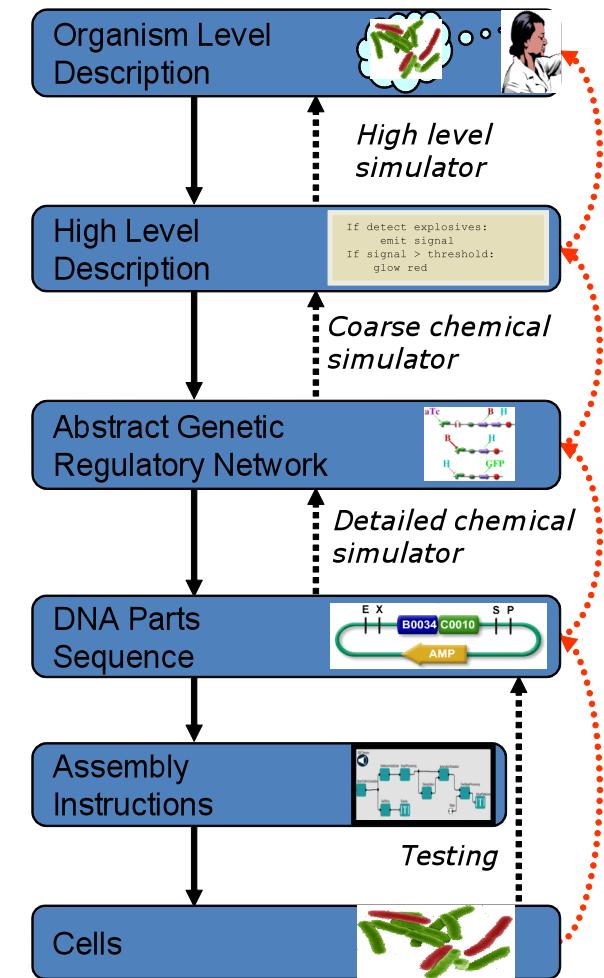
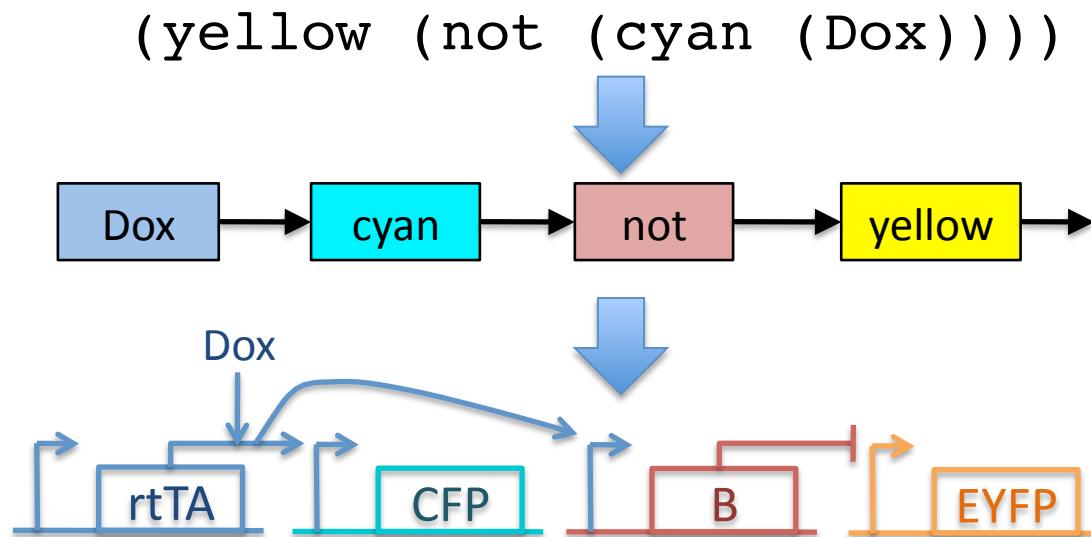
(yellow (not (cyan (Dox)))))



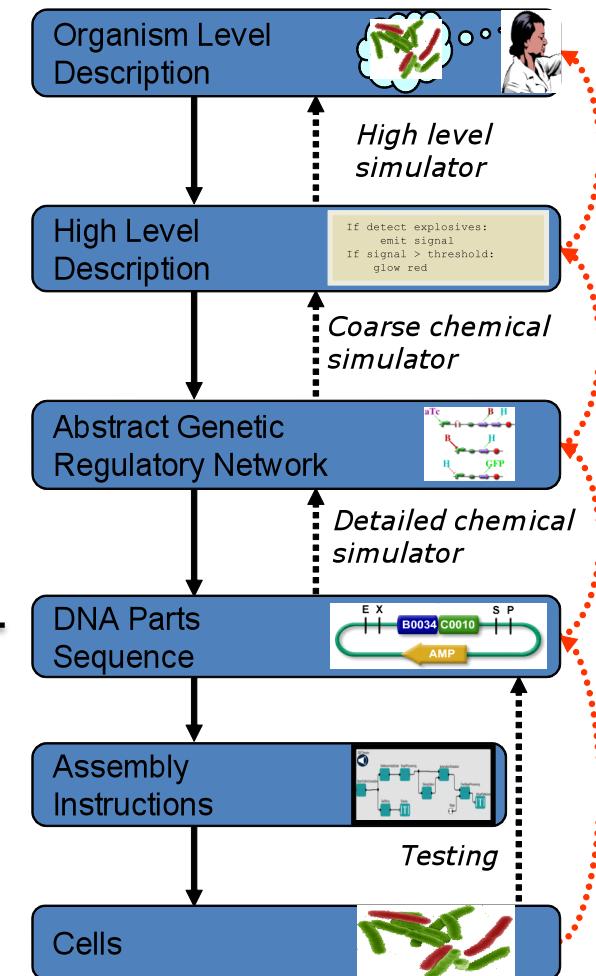
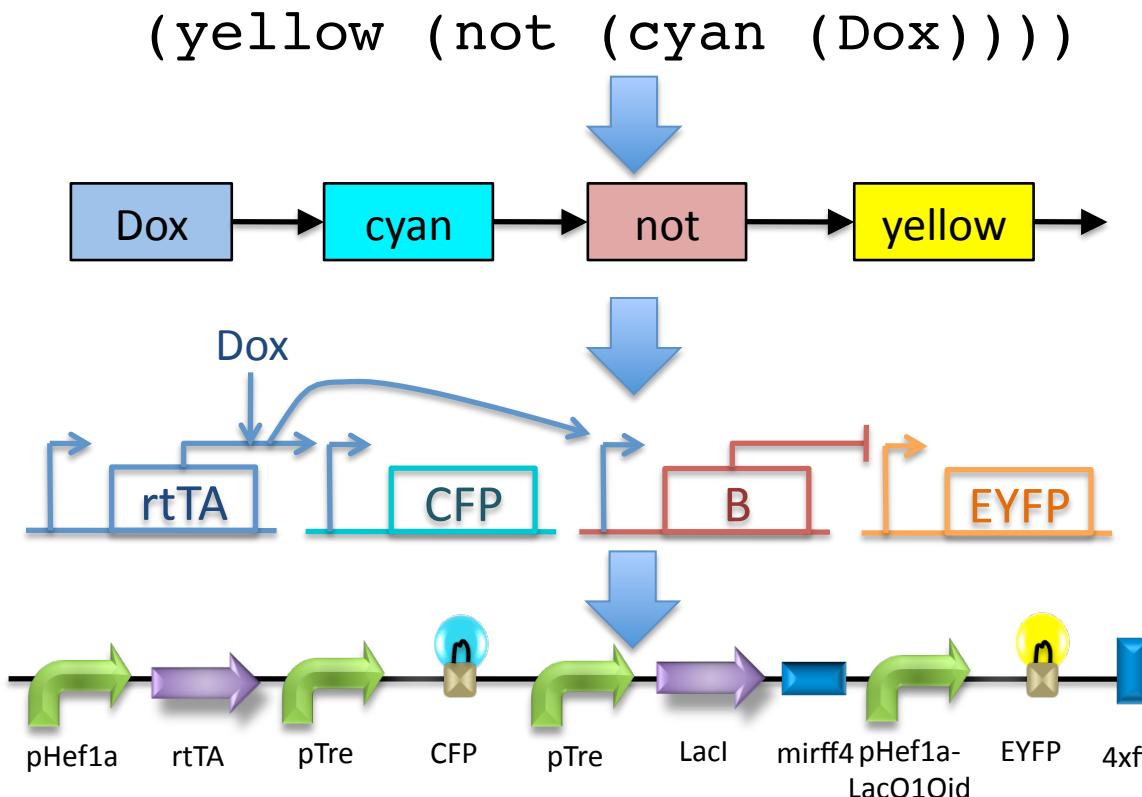
# A Tool-Chain Example



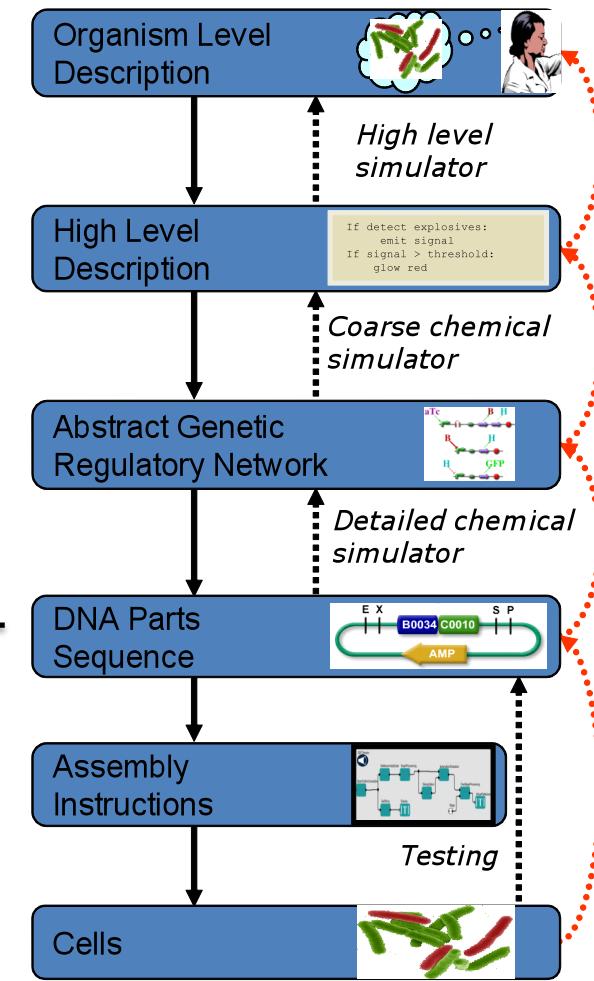
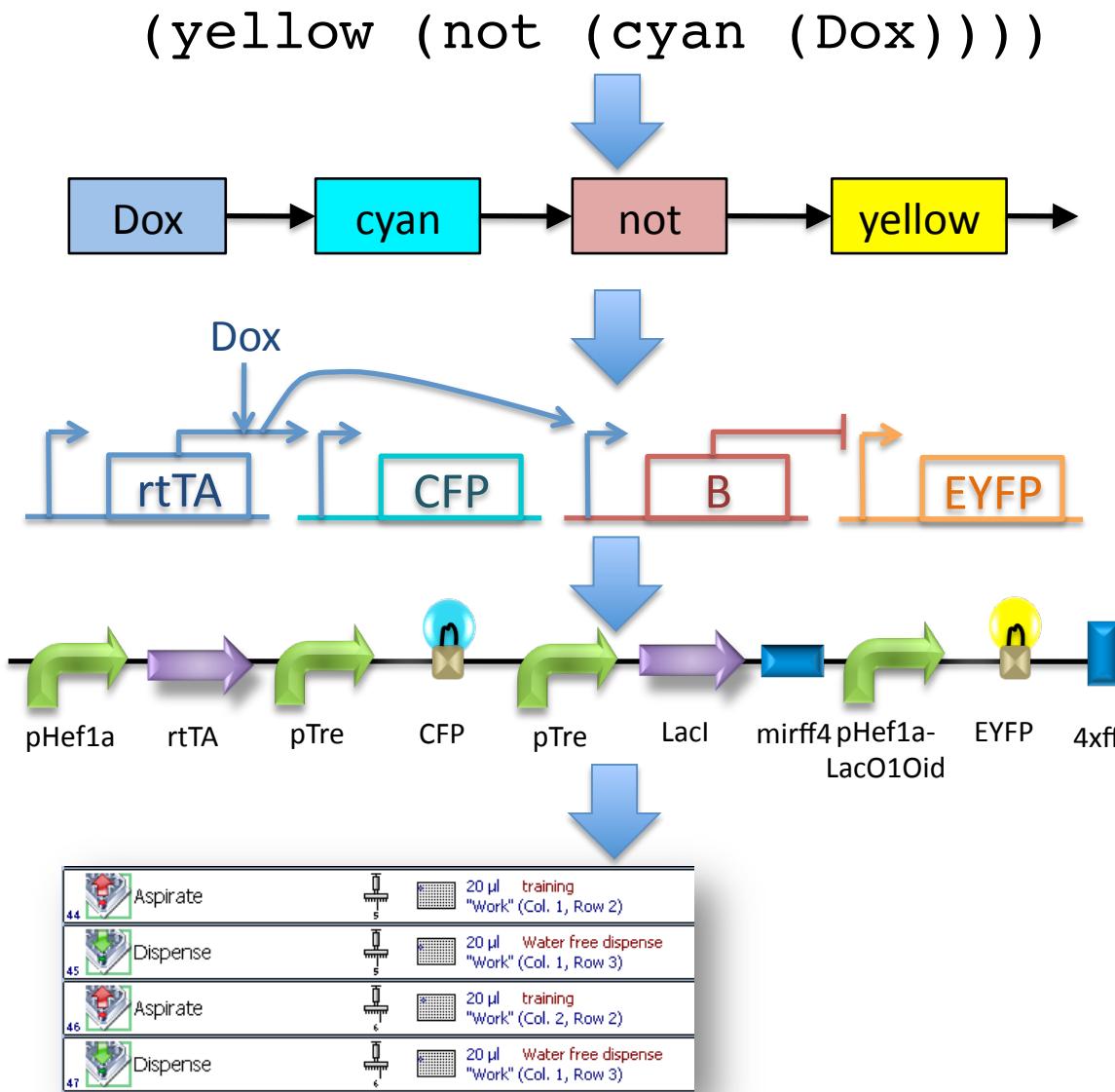
# A Tool-Chain Example



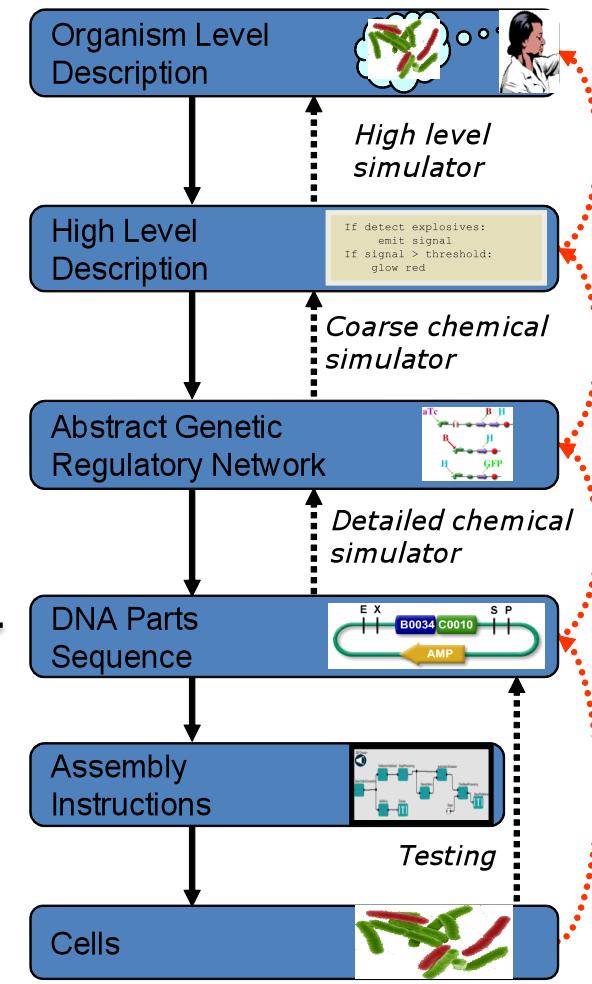
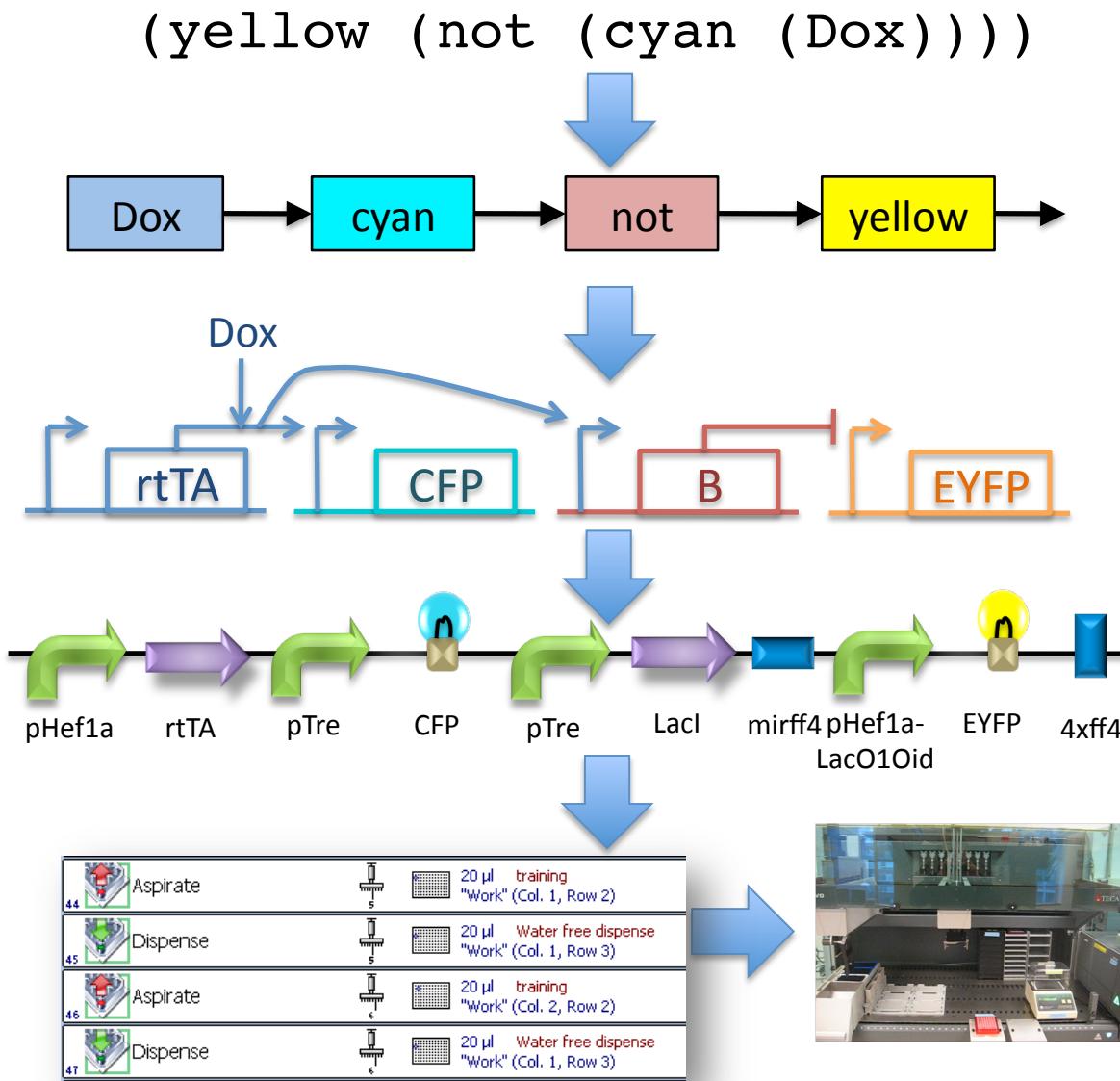
# A Tool-Chain Example



# A Tool-Chain Example



# A Tool-Chain Example



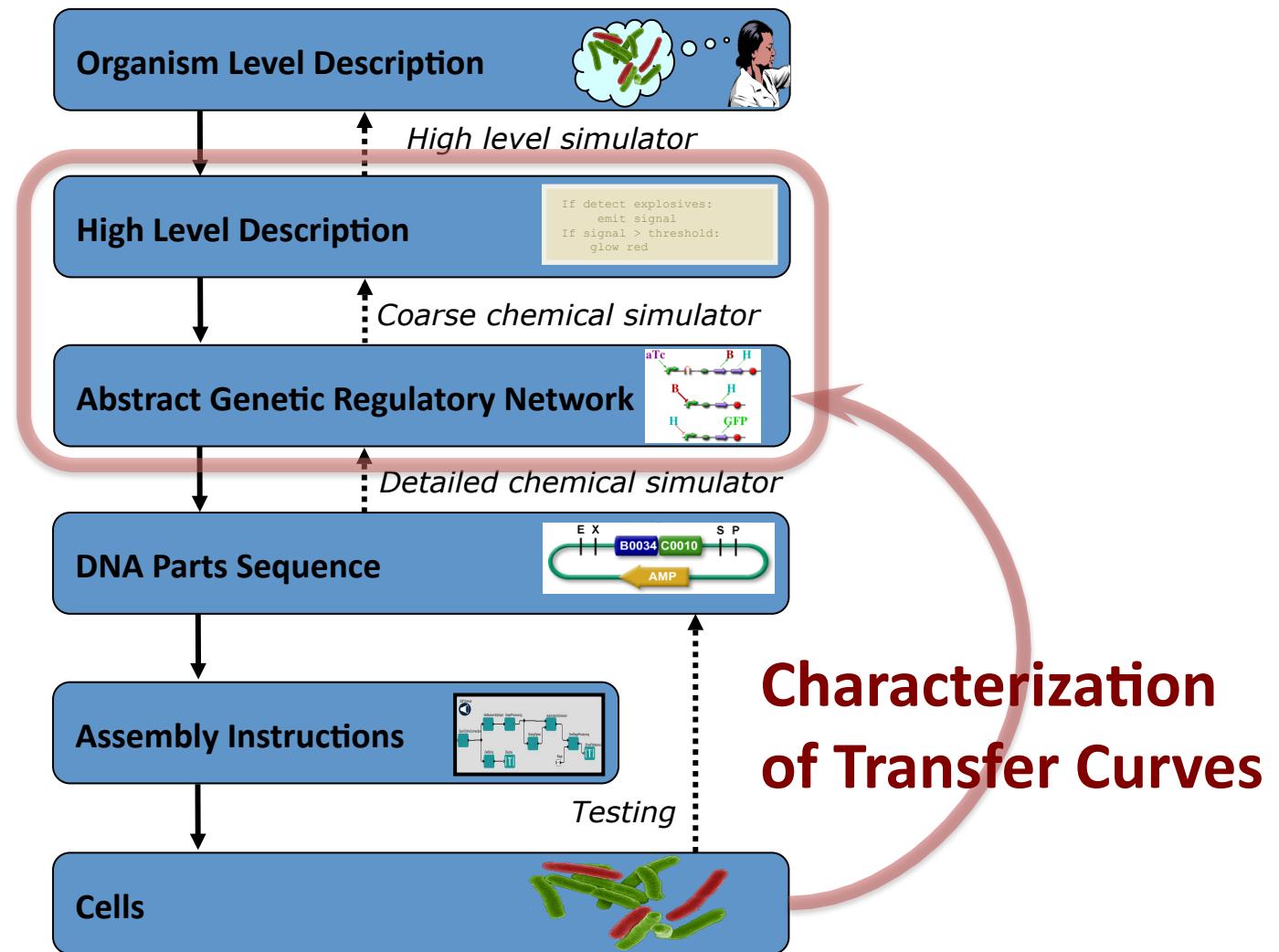
# Current state of the tool-chain:

---

- End-to-end software integration
- Automated designs match hand-generated systems verified *in vivo*
- Automated (mostly) plasmid assembly
- Quantitative prediction of design behavior

# Advances on Two Key Problems:

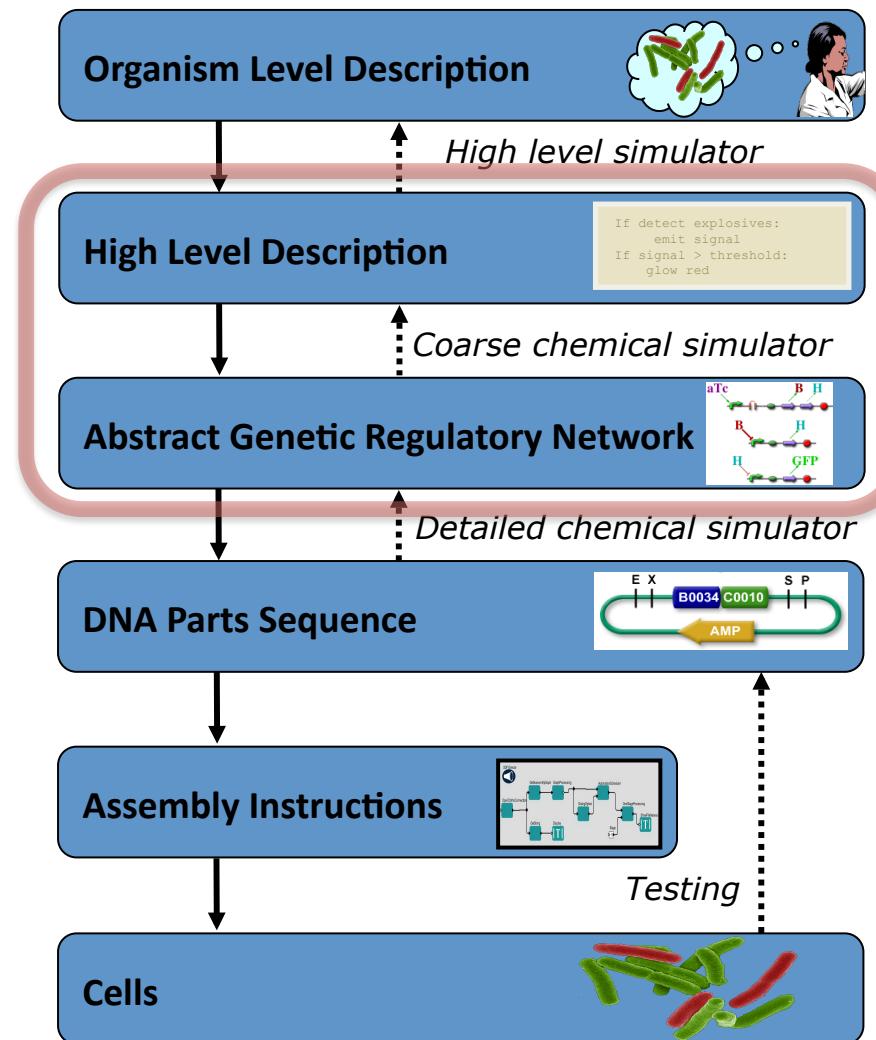
**Compilation & Optimization**



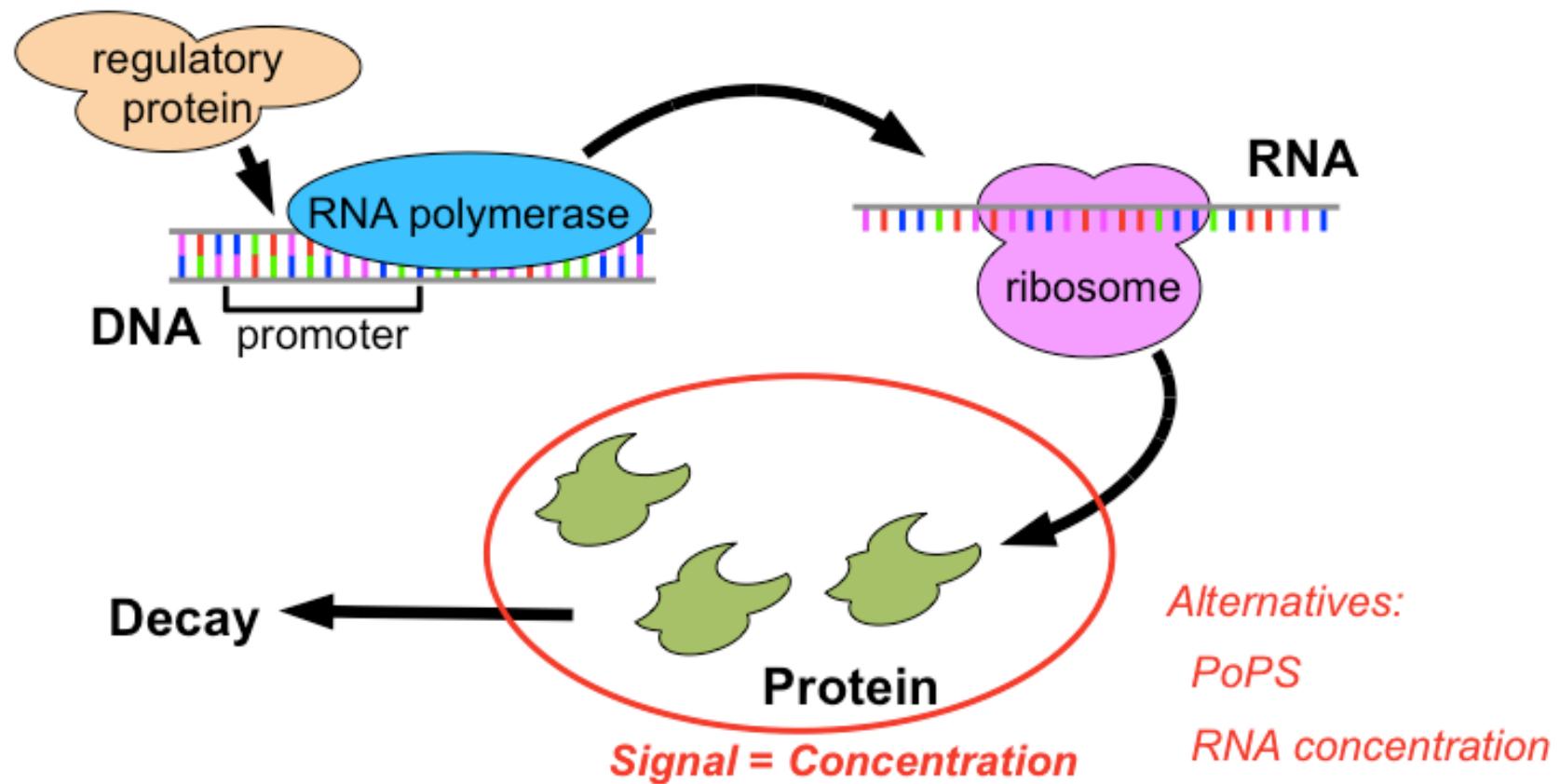
**Characterization  
of Transfer Curves**

# Advances on Two Key Problems:

## Compilation & Optimization

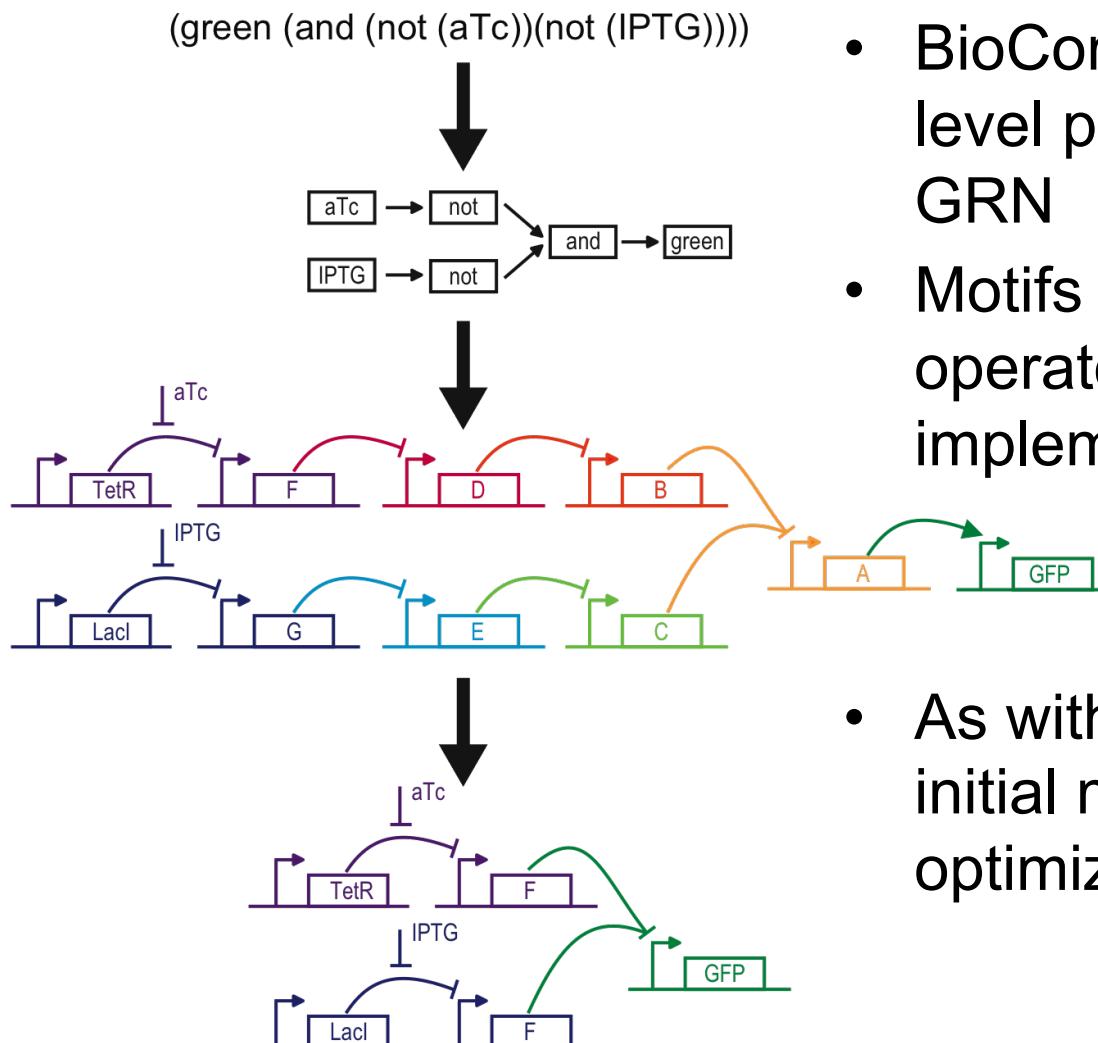


# Transcriptional Logic



Stabilizes at *decay = production*

# BioCompiler Overview

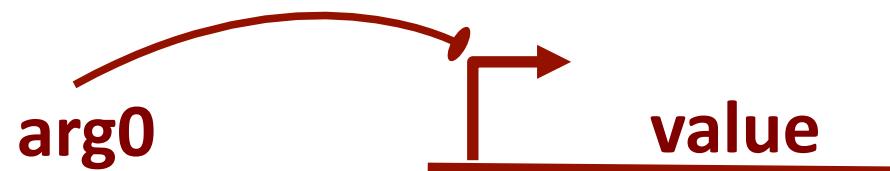


- BioCompiler converts high level program to abstract GRN
- Motifs map high level operators to parameterized implementation in biology.
- As with all compilers, the initial mapping can be greatly optimized.

# Motif-Based Compilation

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

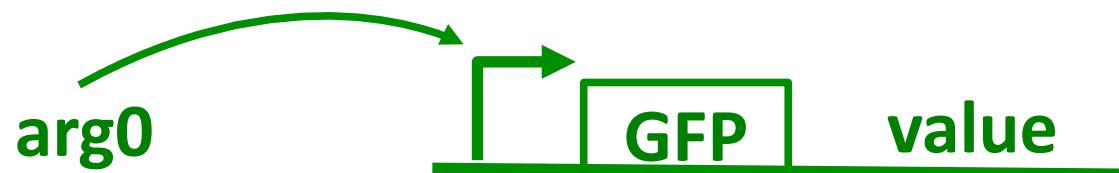
(primitive not (boolean) boolean  
:grn-motif ((P high R- arg0 value T)))



# Motif-Based Compilation

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

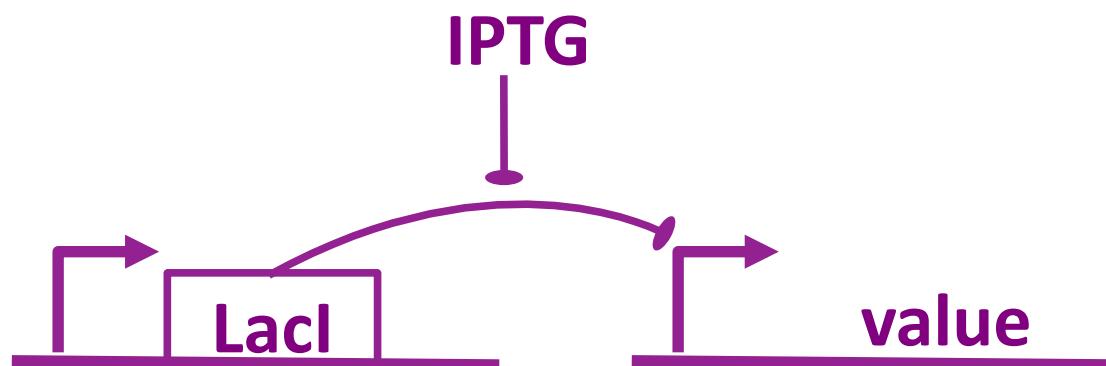
```
(primitive green (boolean) boolean :side-effect  
:type-constraints ((= value arg0))  
:grn-motif ((P R+ arg0 GFP|arg0 value T)))
```



# 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

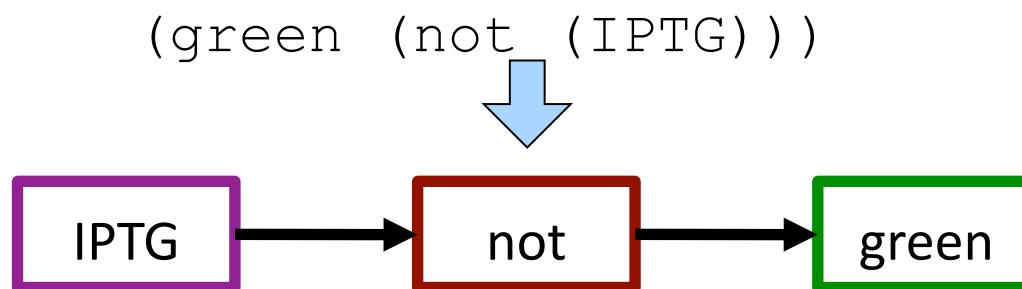
---

- Functional program gives dataflow computation:

```
(green (not (IPTG)) )
```

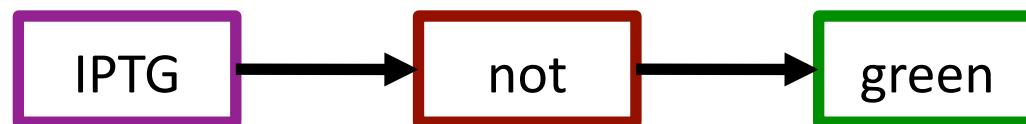
# Motif-Based Compilation

- Functional program gives dataflow computation:



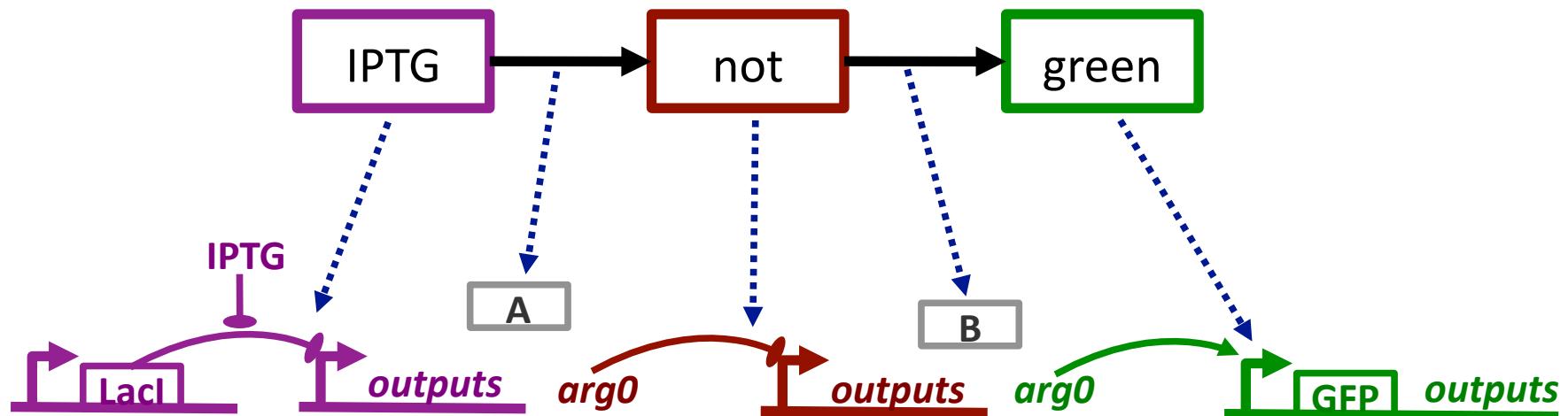
# Motif-Based Compilation

- Operators translated to motifs:



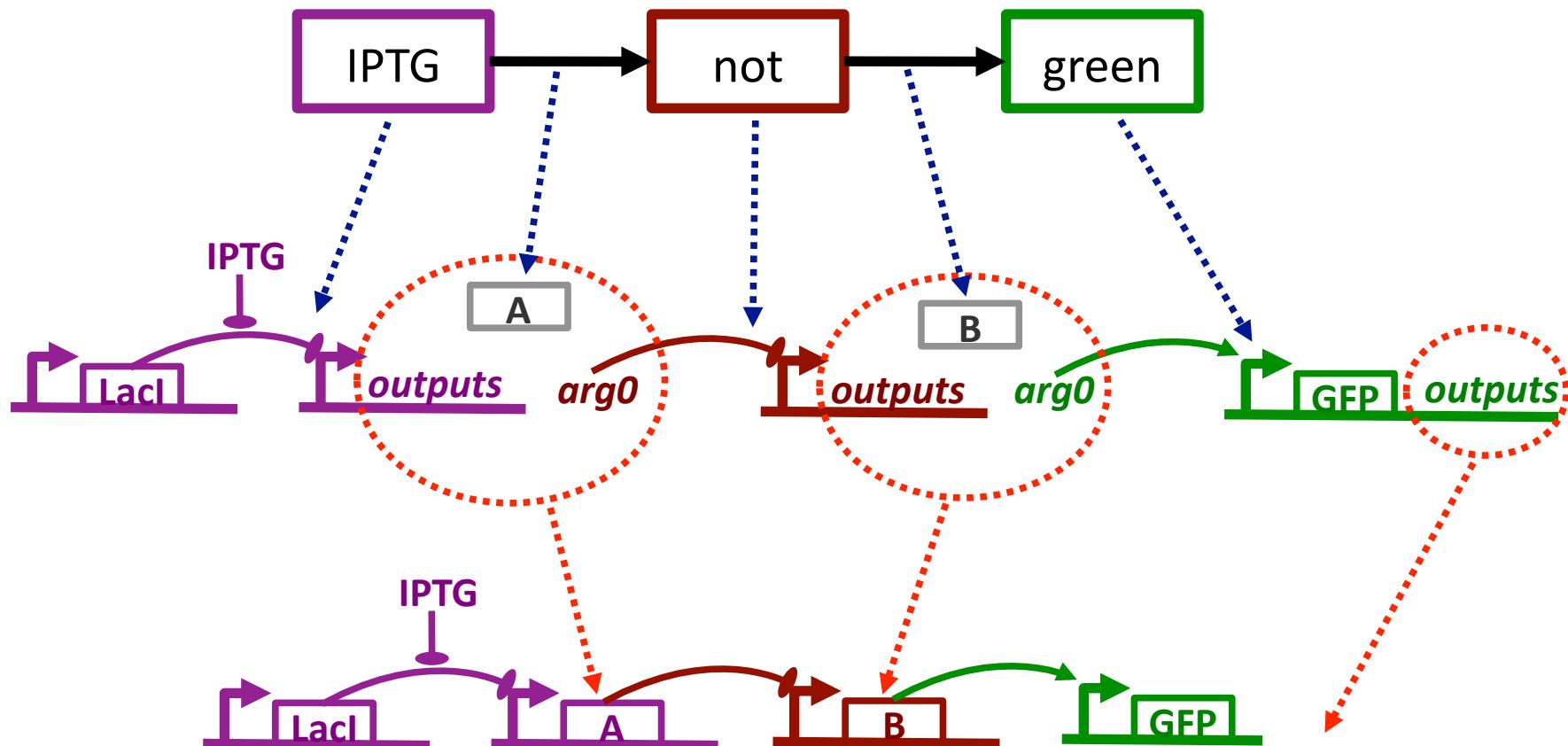
# Motif-Based Compilation

- Operators translated to motifs:



# Motif-Based Compilation

- Operators translated to motifs:



# Complex System: Feedback Latch

---

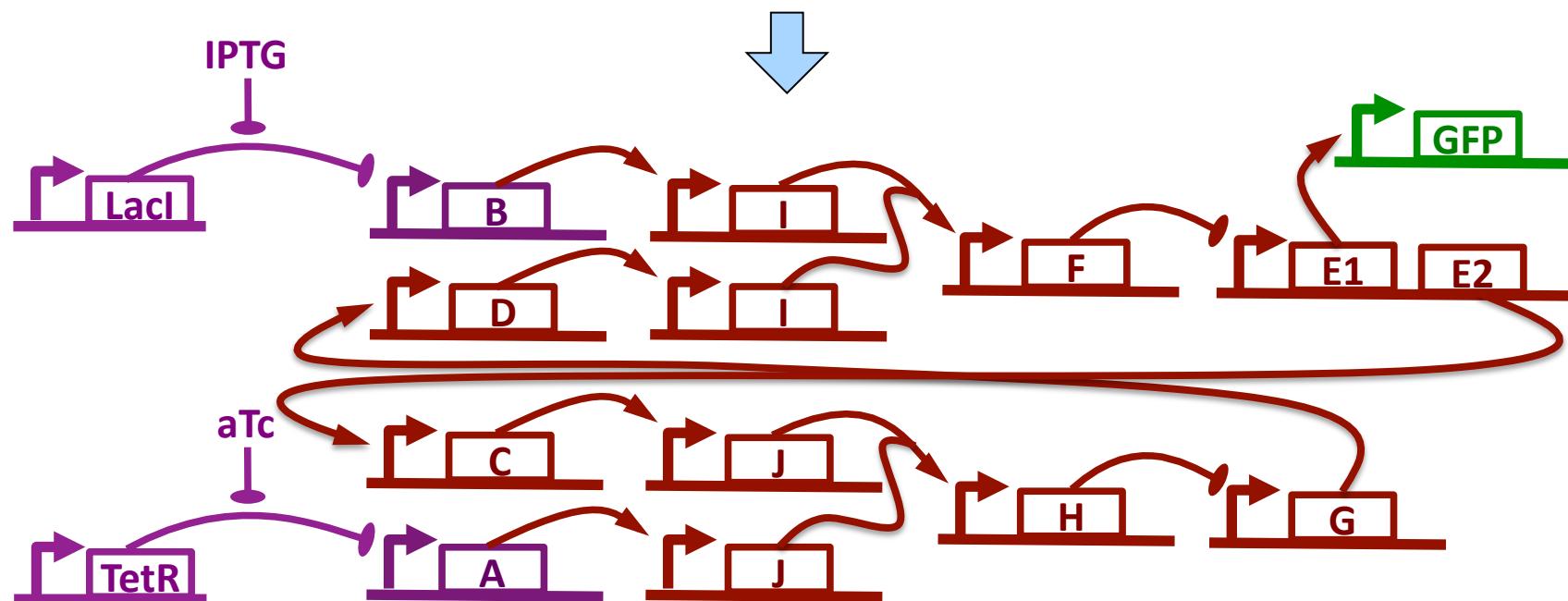
```
(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar) ))
            (o-bar boolean (not (or s o) )) )
    o) )

(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**)) )



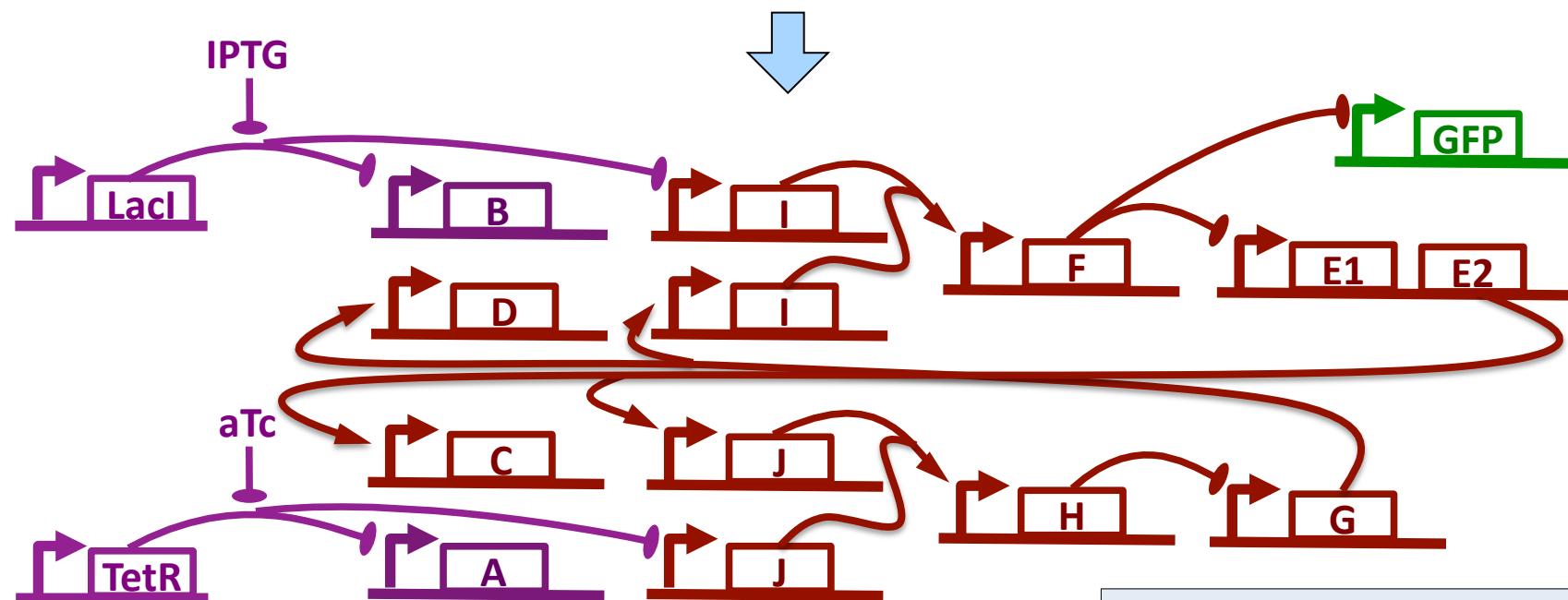
*Unoptimized: 15 functional units, 13 transcription factors*

# 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**)))



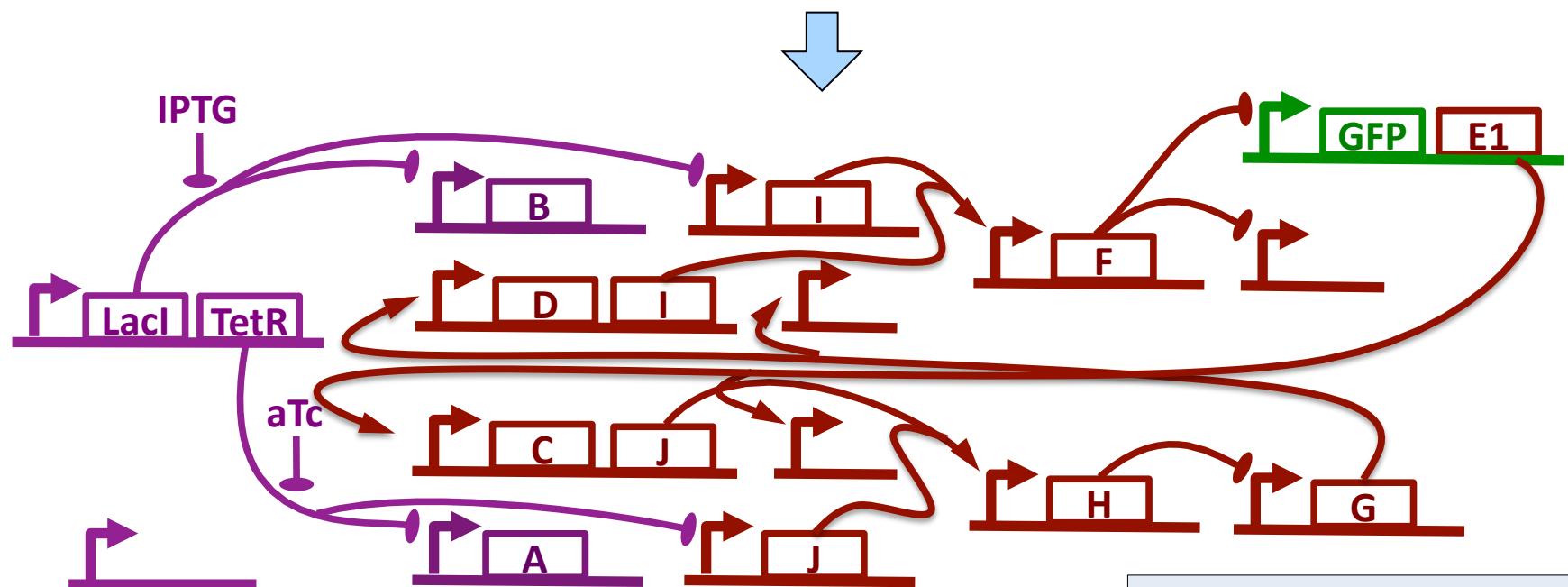
Unoptimized: 15 functional units, 13 transcription factors

*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**)) )



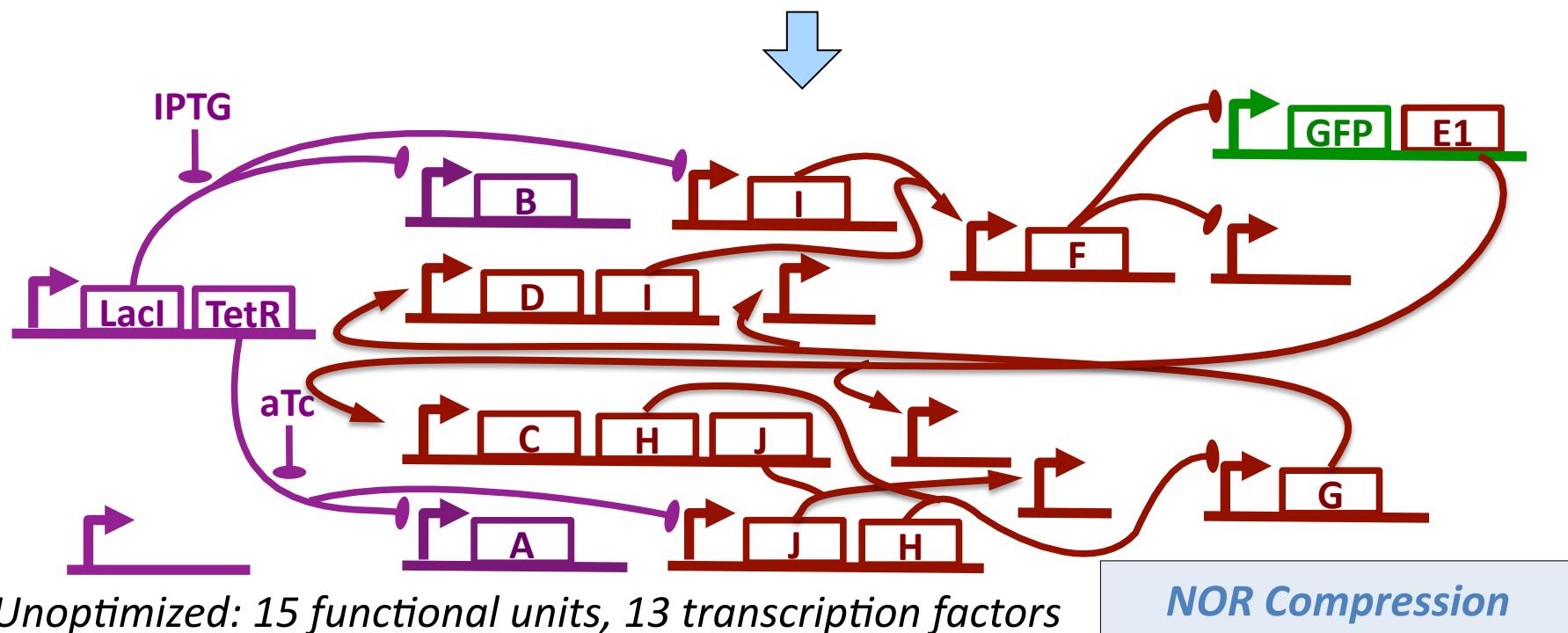
*Unoptimized: 15 functional units, 13 transcription factors*

**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**)) )

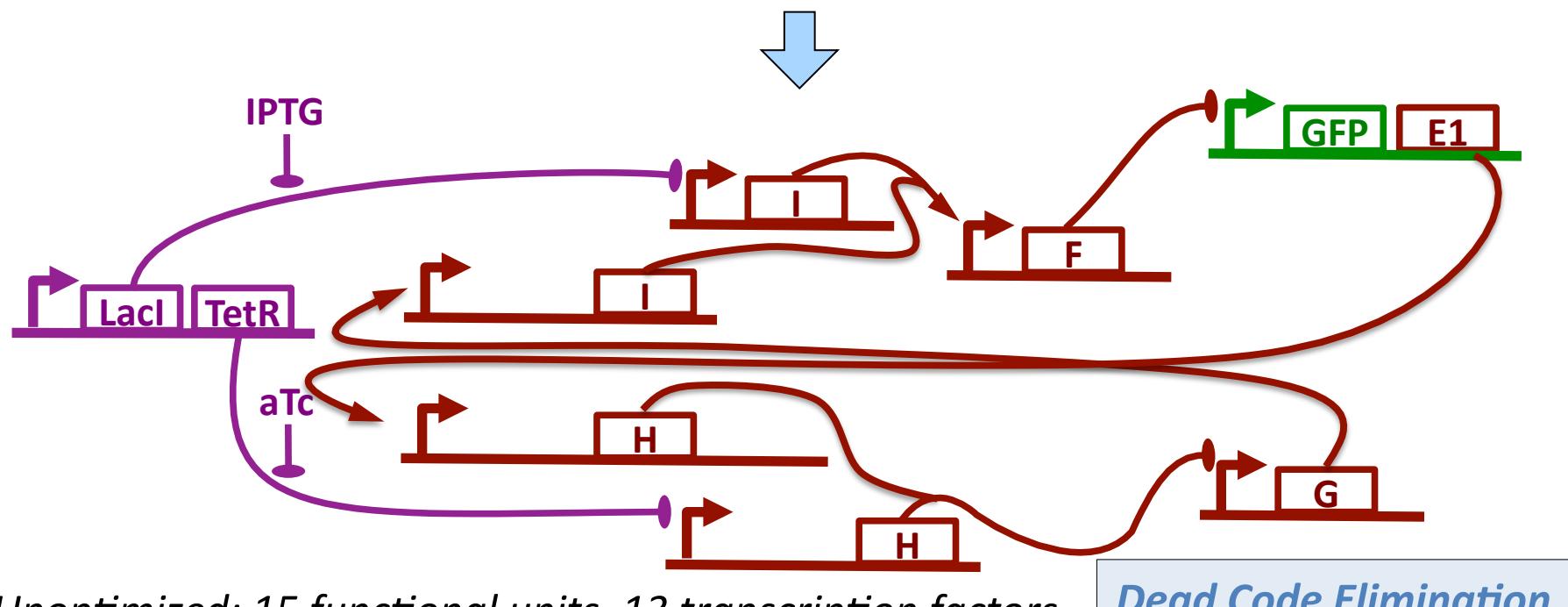


# Optimization of Complex Designs

```
(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**)))

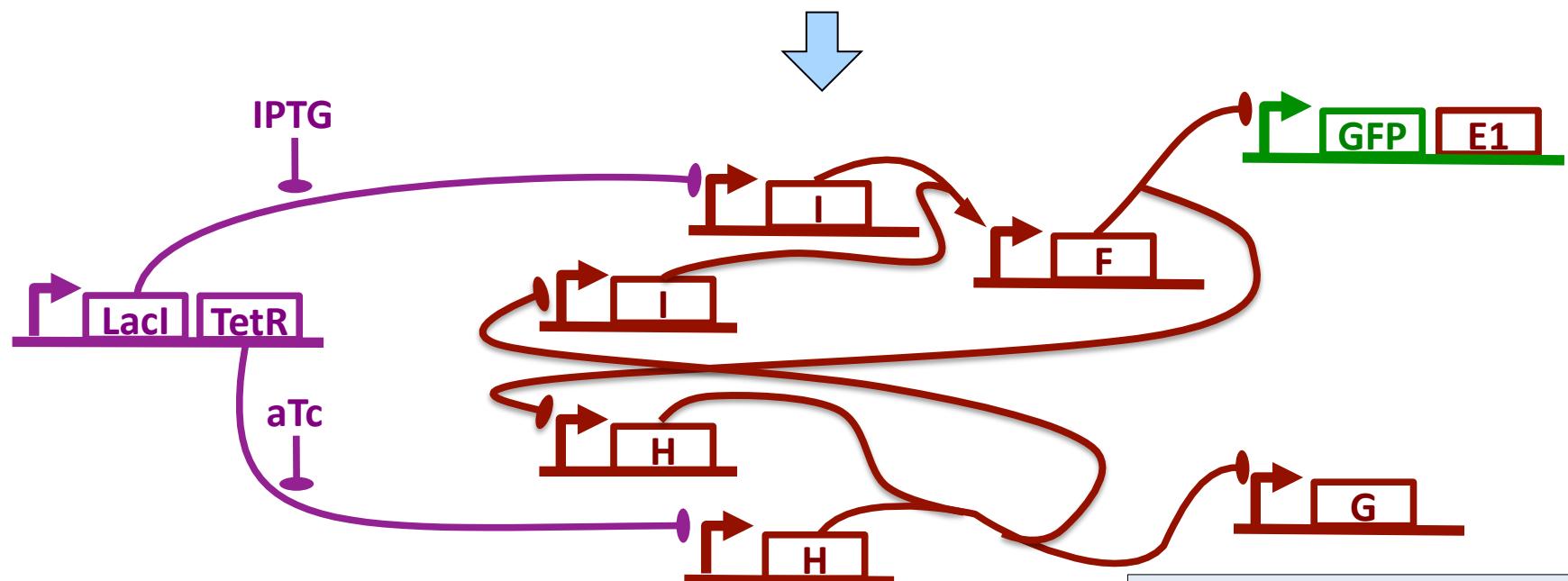


# Optimization of Complex Designs

```
(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**)))



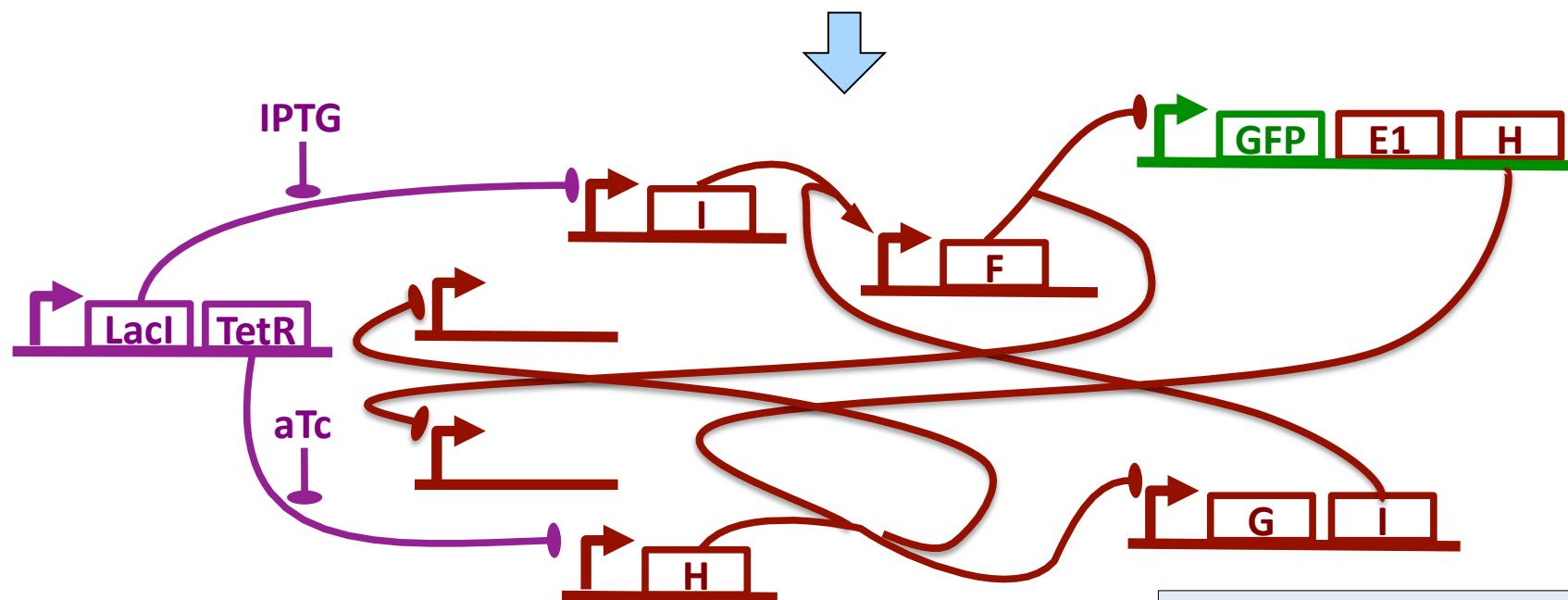
*Unoptimized: 15 functional units, 13 transcription factors*

*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**)) )



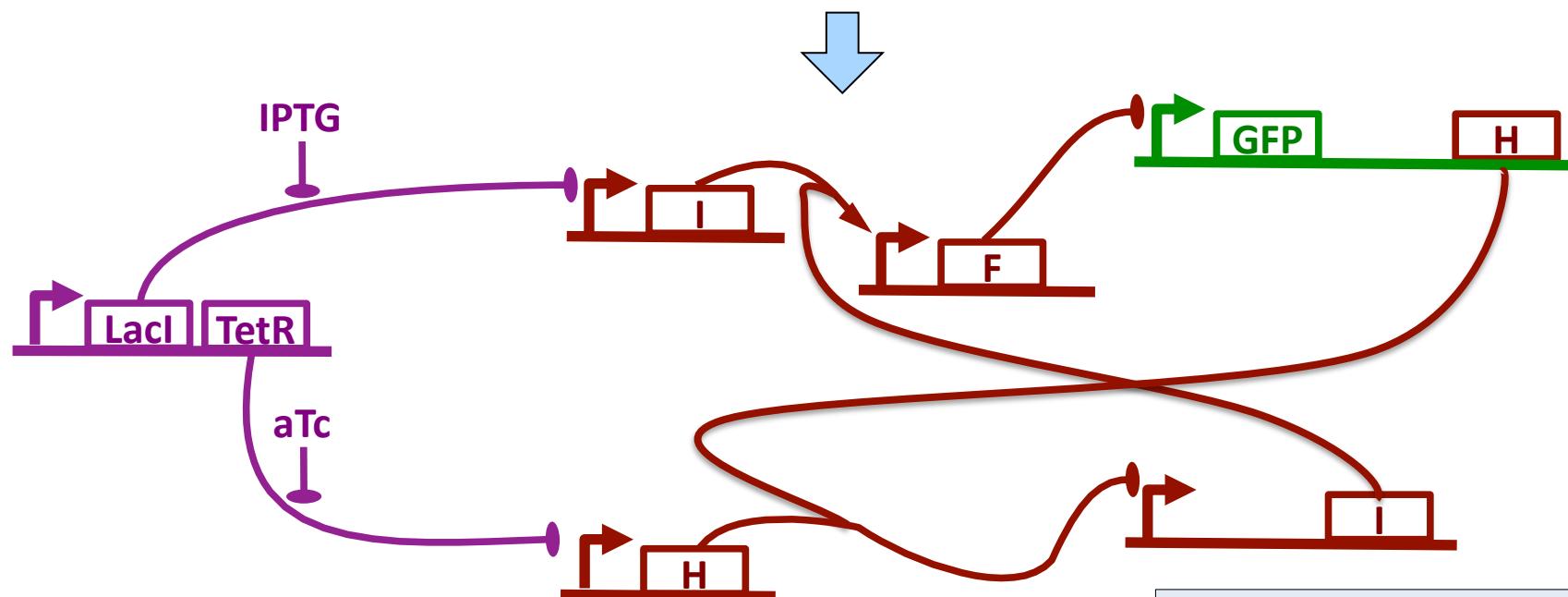
*Unoptimized: 15 functional units, 13 transcription factors*

**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*

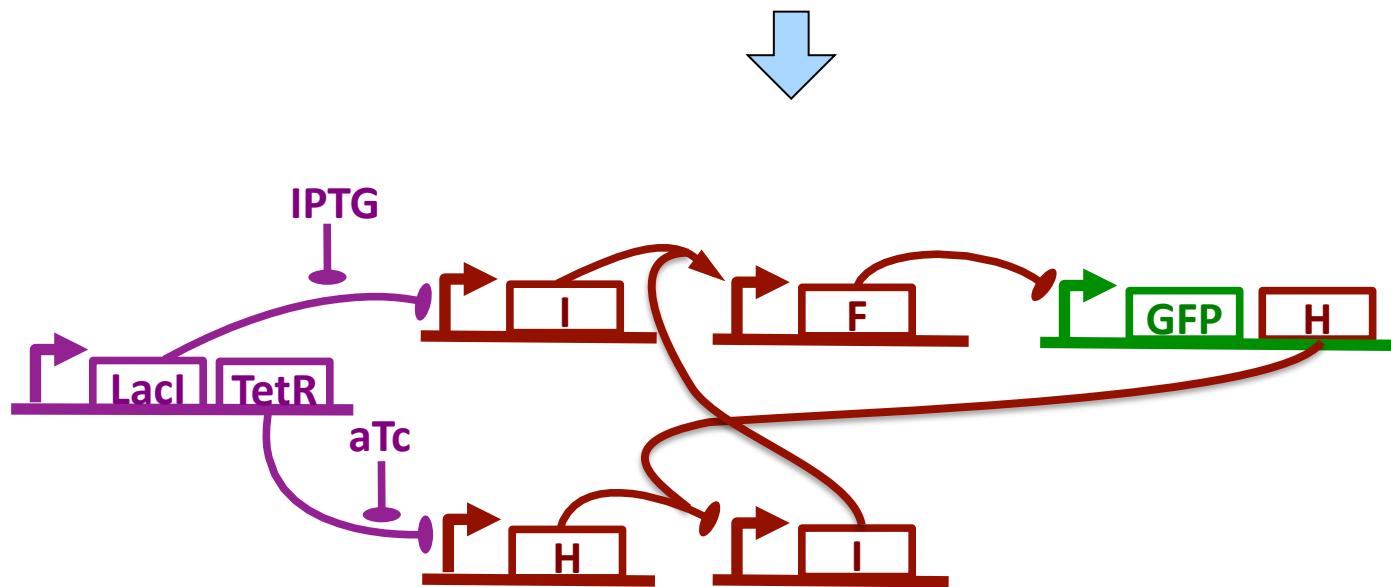
**Dead Code Elimination**

# 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)))
```

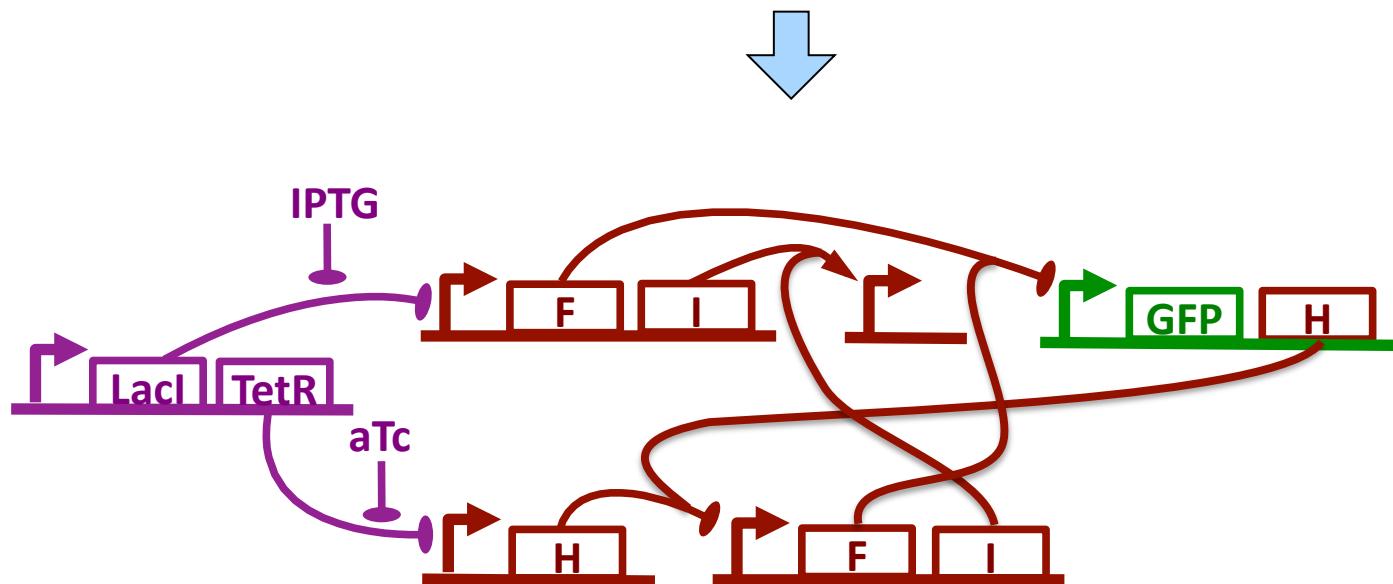


*Unoptimized: 15 functional units, 13 transcription factors*

# 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**)))



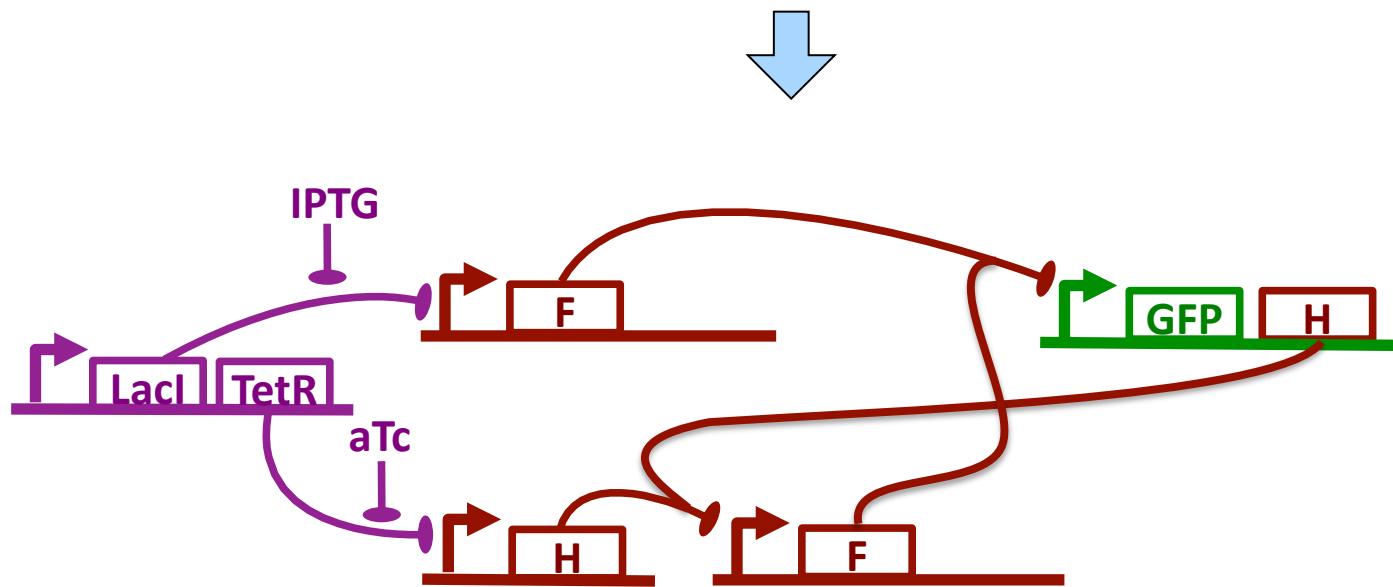
*Unoptimized: 15 functional units, 13 transcription factors*

**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**)))



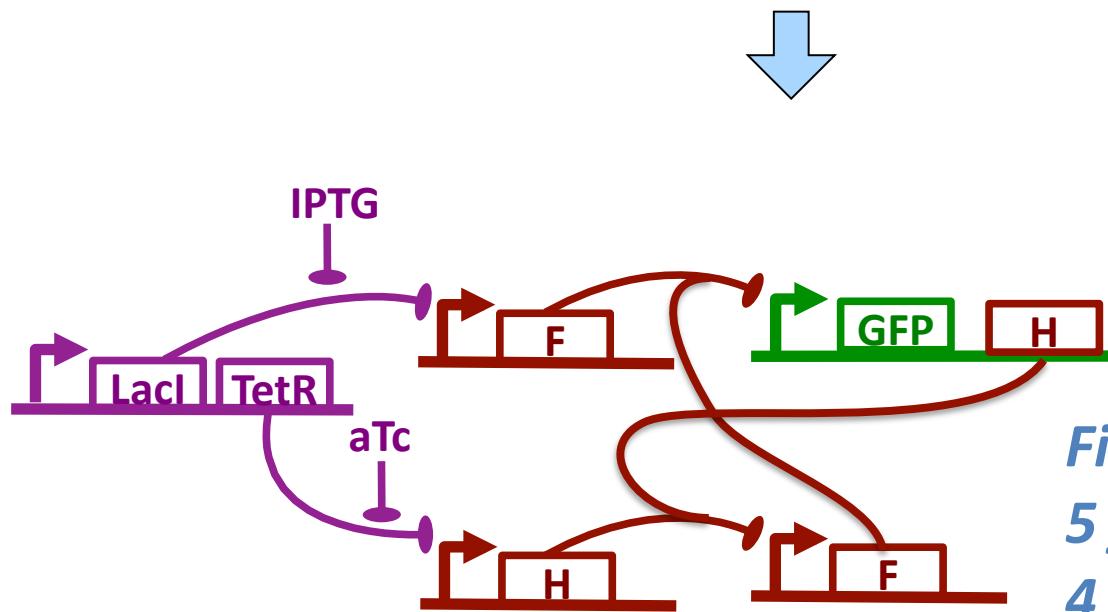
*Unoptimized: 15 functional units, 13 transcription factors*

**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*

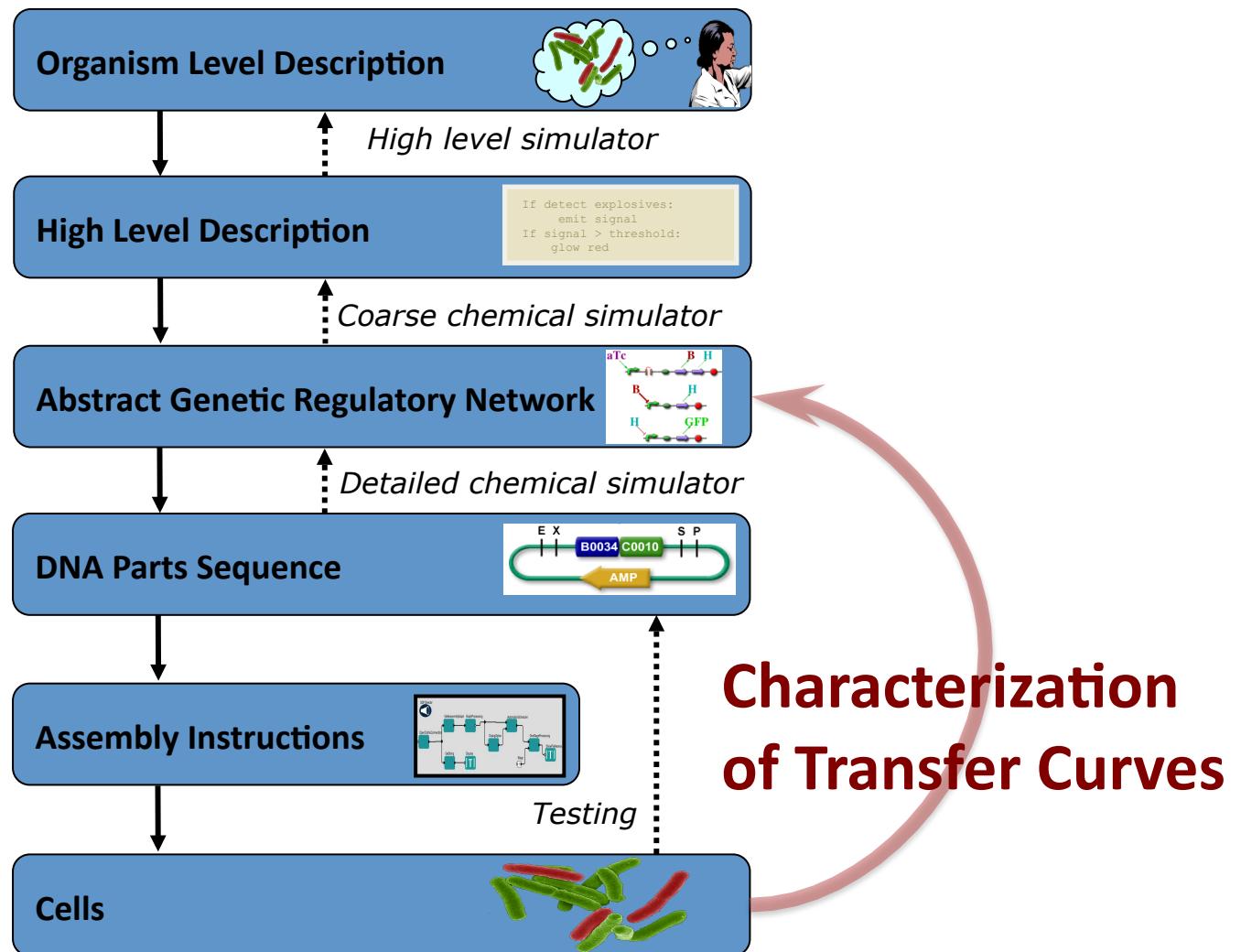
*Unoptimized: 15 functional units, 13 transcription factors*

# Compilation & Optimization Results:

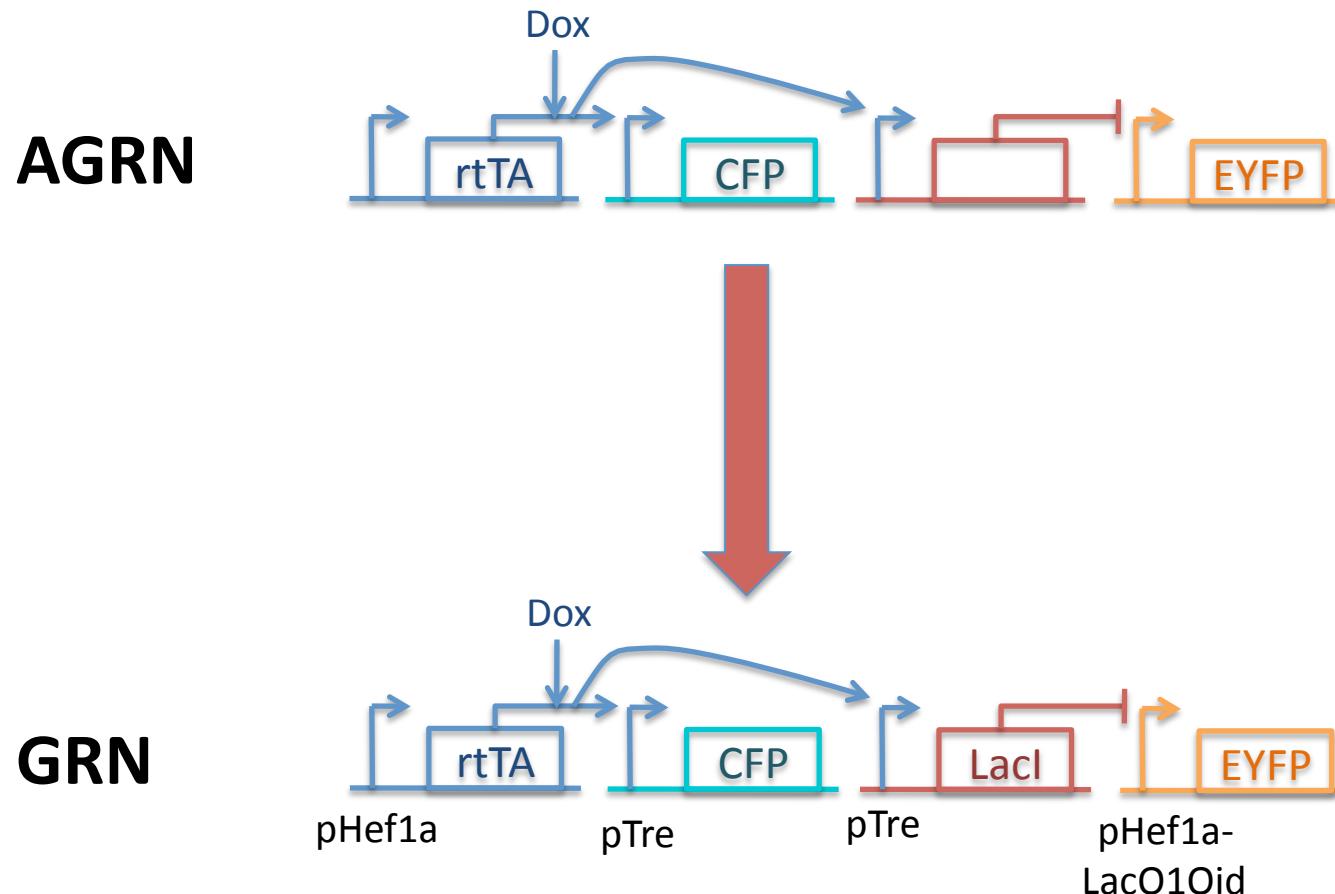
---

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

# Advances on Two Key Problems:

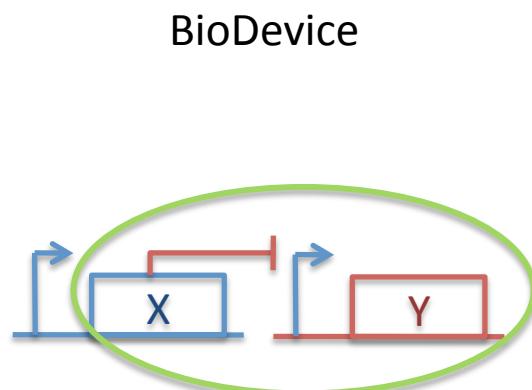


# From Abstract GRN to Part Sequence

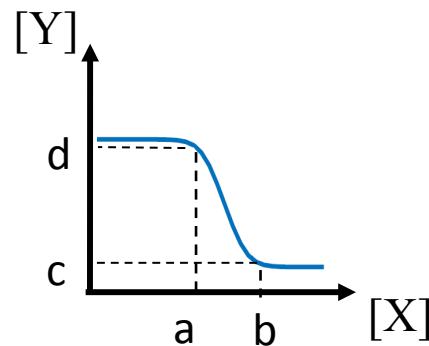


# Signal Matching

Abstract GRN specifies logical relationships.  
Correct implementation depends on signal ranges



Transfer Curve

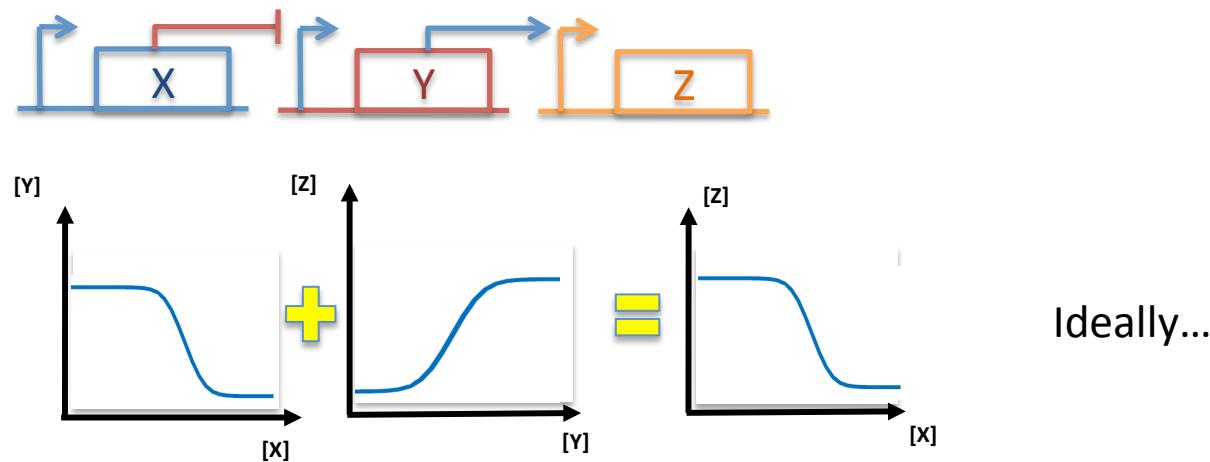


Digital Behavior

X	Y
Off [x] < a	On [y] > d
On [x] > b	Off [y] < c

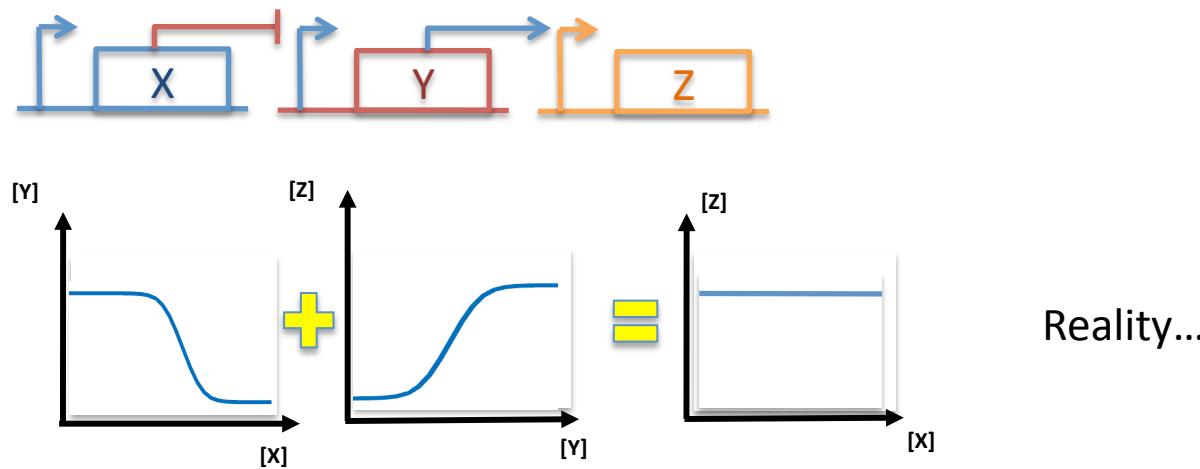
# Signal Matching

- Composition should preserve digital behavior



# Signal Matching

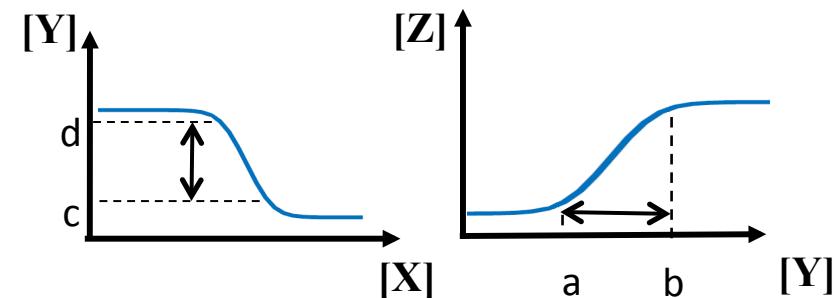
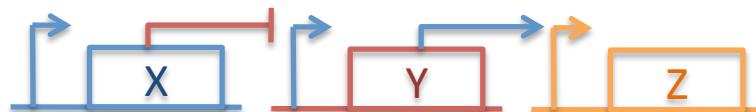
- Composition should preserve digital behavior



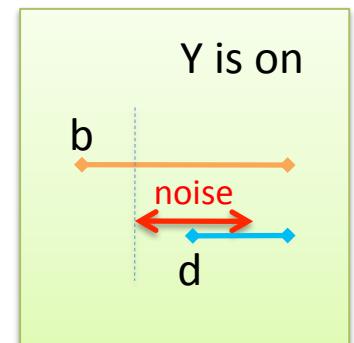
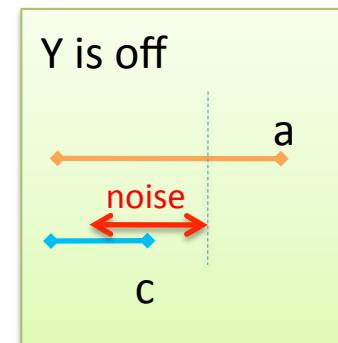
**Signal Matching Problem:** How do we pick the parts that have compatible interpretations for on/off so that when composed will preserve digital behavior?

# Solution

- Pick the parts that are **signal compatible**
  - operate in same signal range where Signal =Concentration

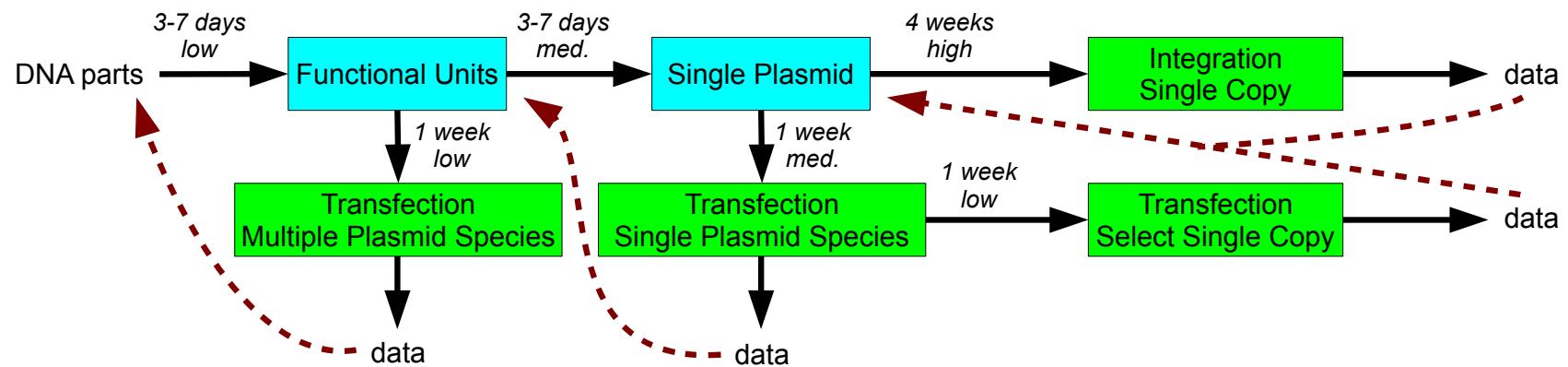
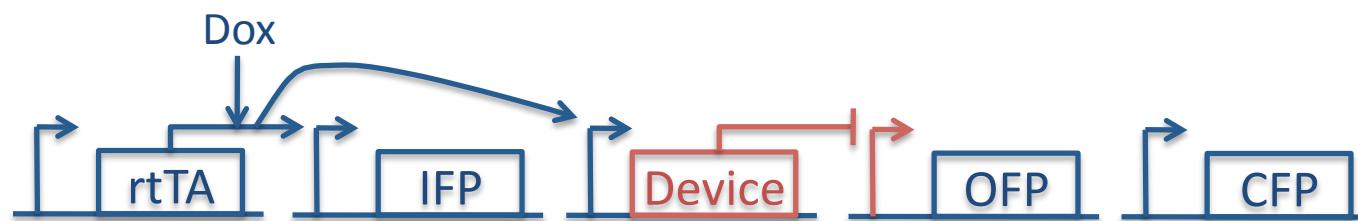


- Parts are signal compatible iff noisy output range is contained in valid input range



# Key Problem: Device Characterization

Goal: quantify single-cell I/O concentration relation



*Pipelined protocol trades experimental for analytic complexity*

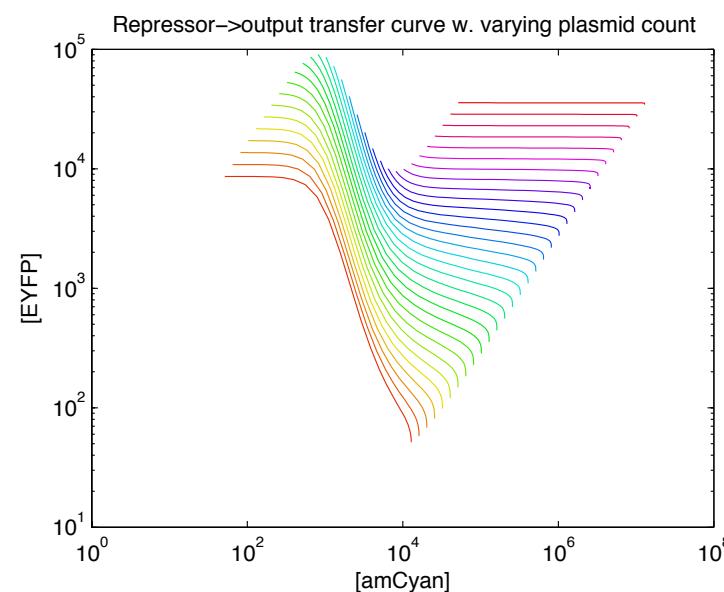
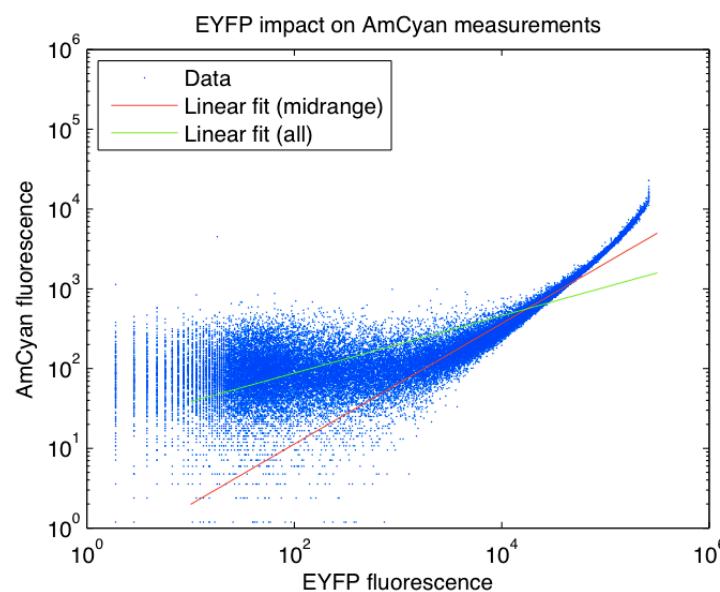
# From Fluorescence to Static Discipline

Fluorescence of proxy  
proteins at N hours

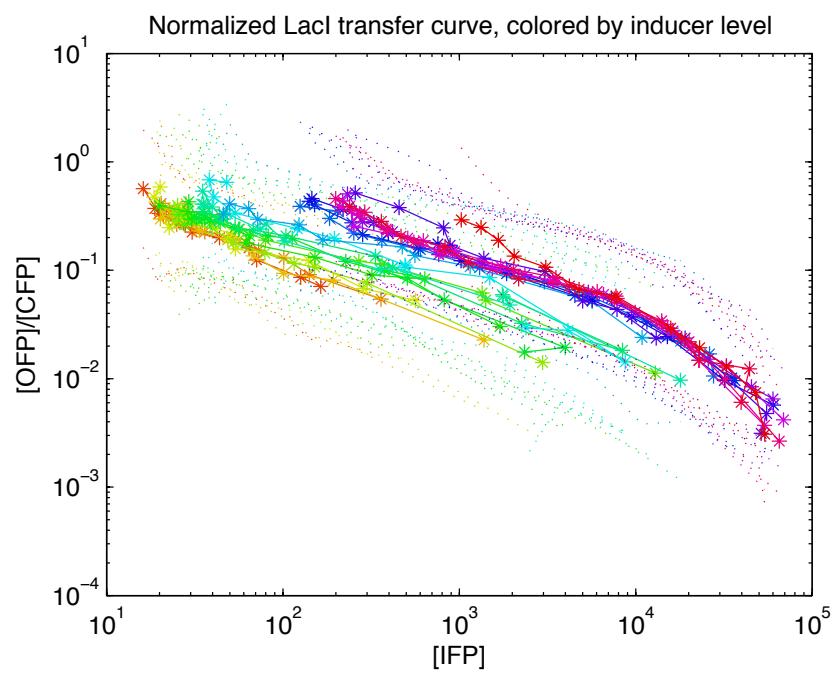
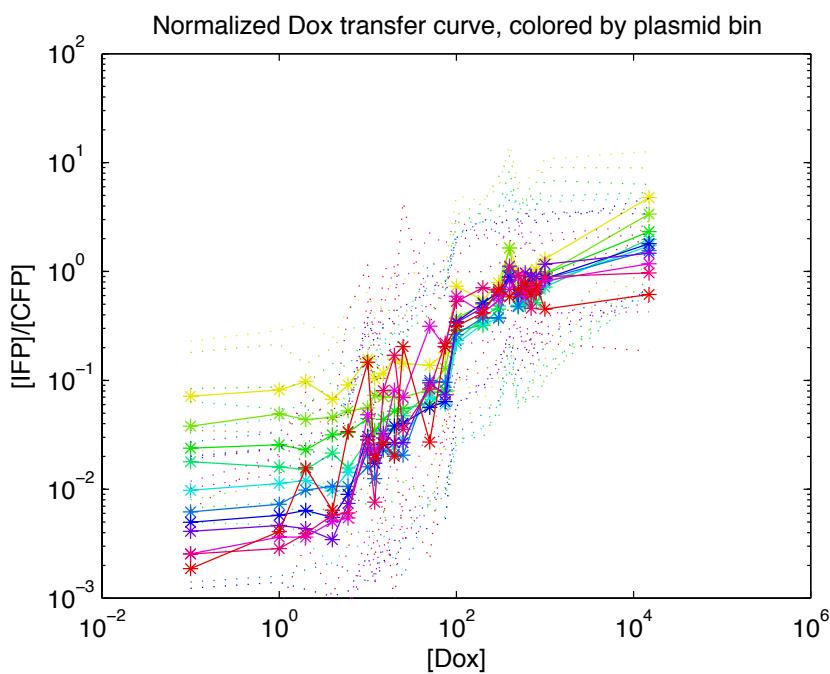
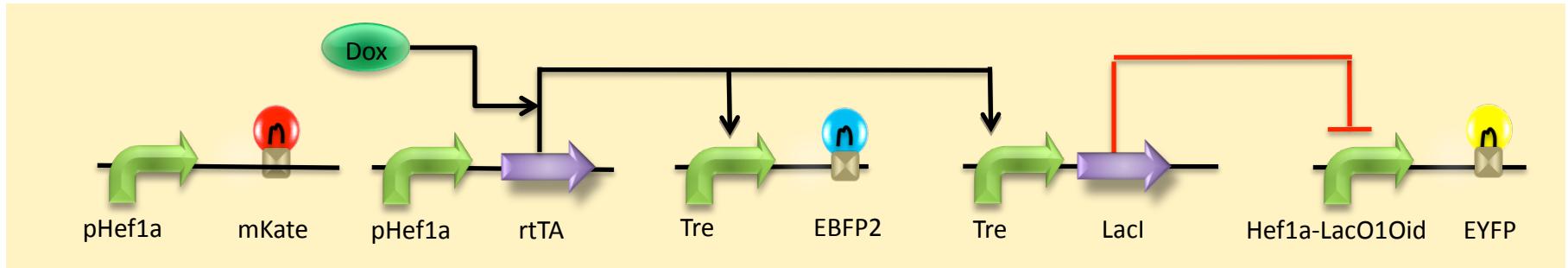
Standardized  
Fluorescence

Model-Compensated  
Transfer Curve

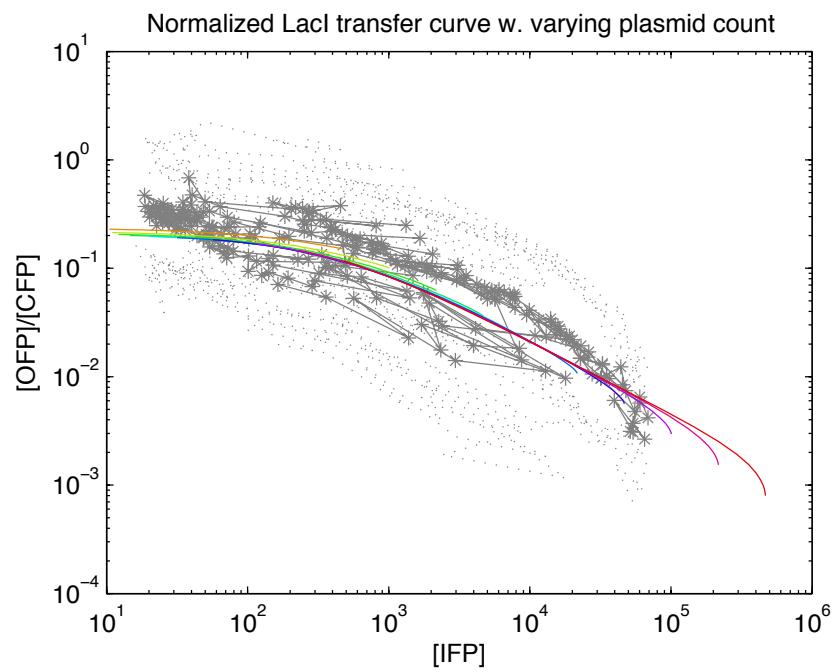
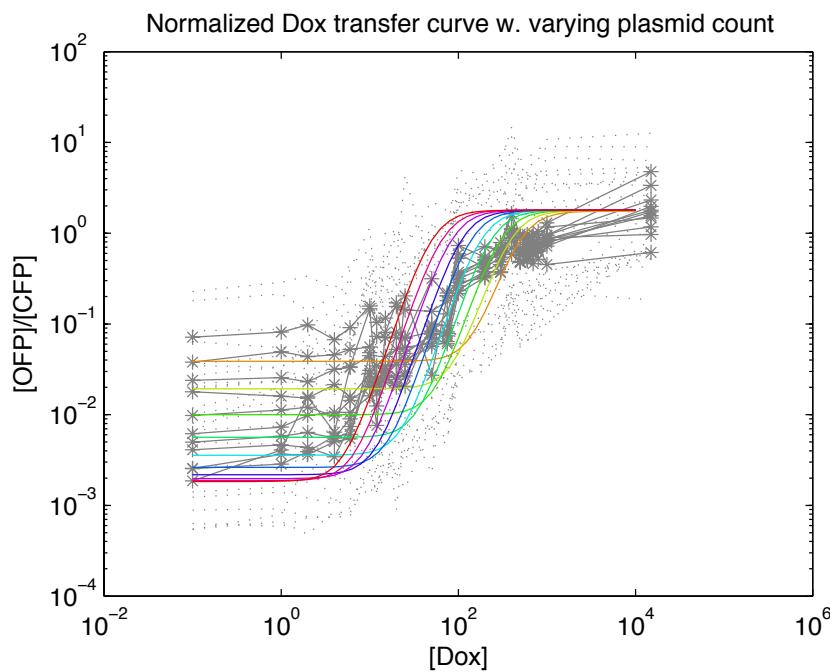
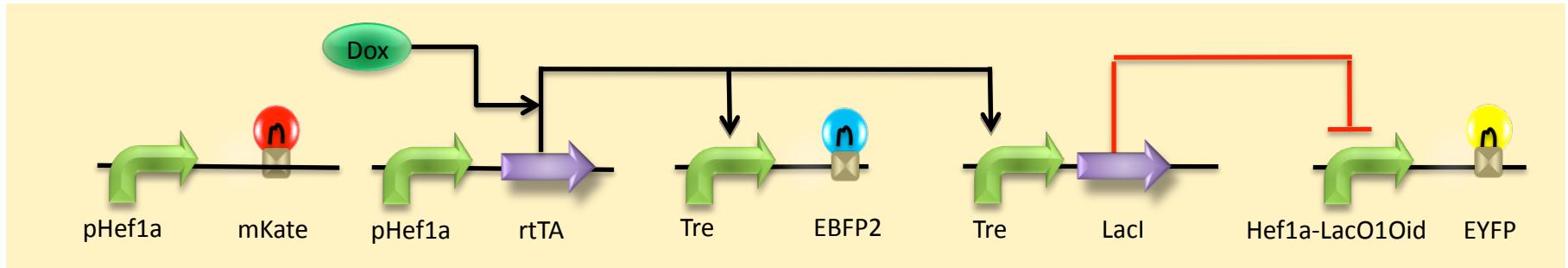
Inflection  
Points



# Example Characterization: LacI repressor



# Example Characterization: LacI repressor



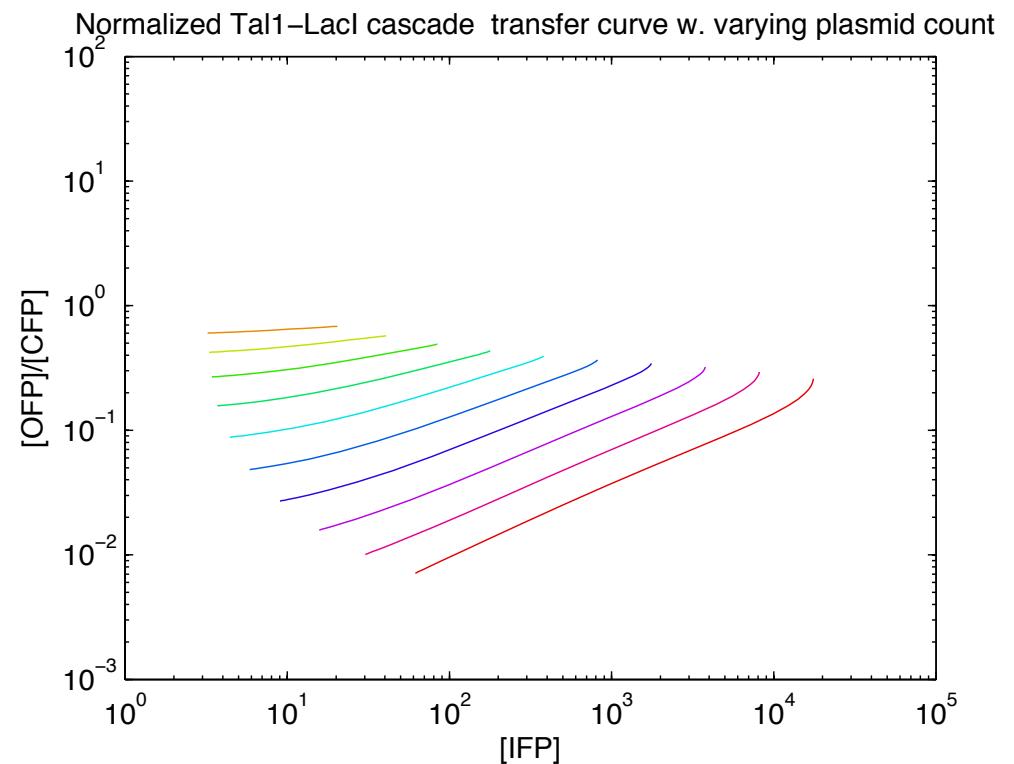
# Preliminary Results for Prediction

- Two-inverter cascade:

**(red (not (not (yellow (Dox))))))**

- Model prediction:

- Low copy: no effect
- High copy: 30x
- Gradual transition



# Preliminary Results for Prediction

- Two-inverter cascade:

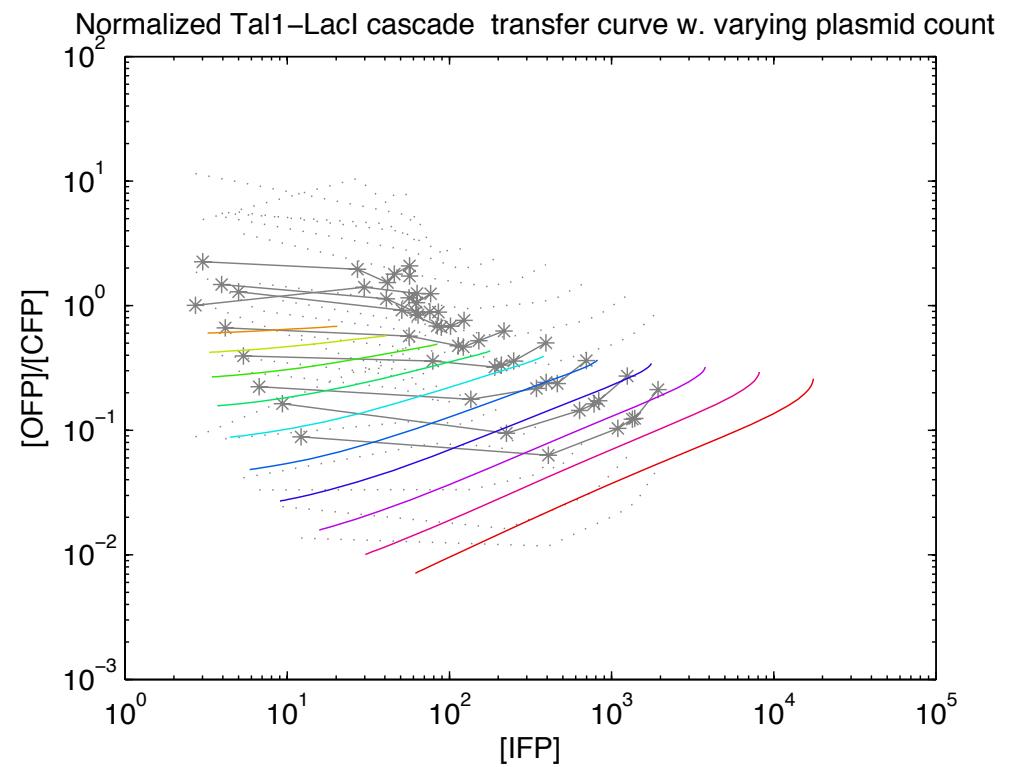
**(red (not (not (yellow (Dox))))))**

- Model prediction:

- Low copy: no effect
- High copy: 30x
- Gradual transition

- Experimental result:

- Low copy: no effect
- High copy: 10x
- Gradual transition
- 3x higher

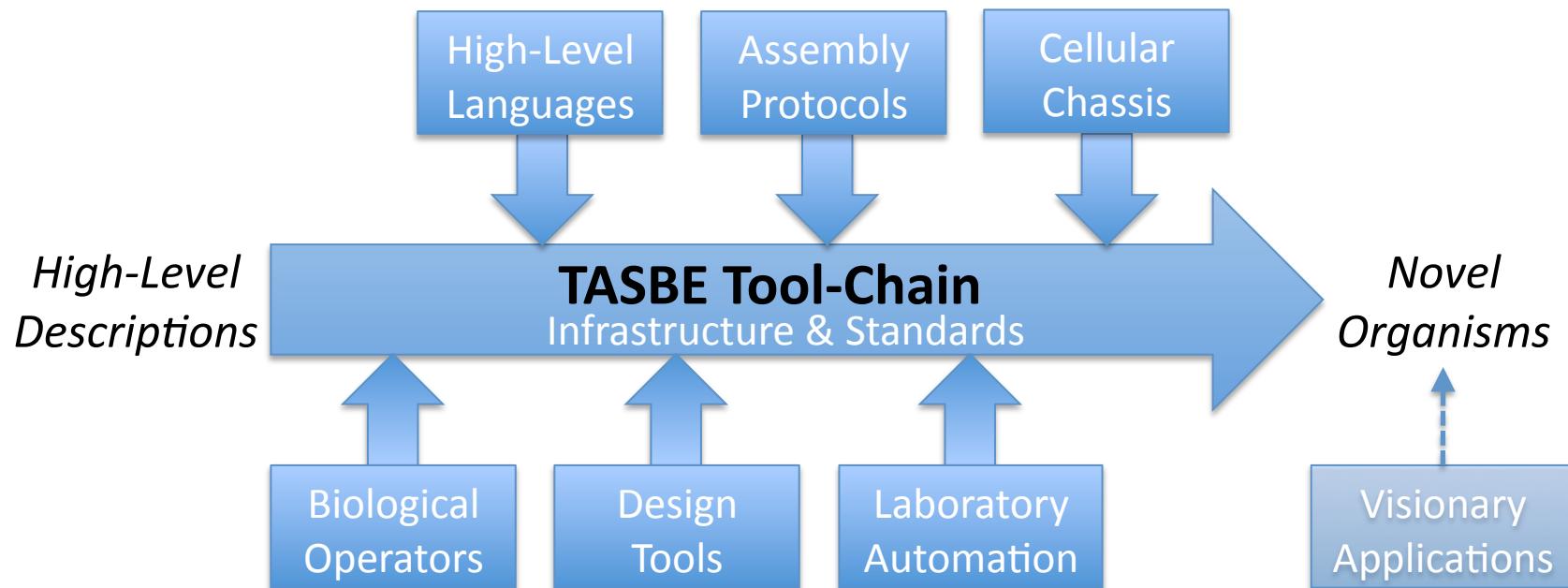


## Contributions:

---

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

# Toward a community platform...



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