Complex Circuits in Synthetic Biology
Our Projects

- Bacterial Etch-a-Sketch
- Bacterial Full Adder
- MYS!S BioBrick Optimizer
BACTERIAL
ETCH-A-SKETCH
Goal

[Image of a magnifying glass with the word "Rutgers" inside]
Challenges

- Light
- Color
- Sensitivity
- Selectivity
- Speed
- Noise
Challenges

- **Light** – need bacteria to respond to light
- Color
- Sensitivity
- Selectivity
- Speed
- Noise
Challenges

• Light – need bacteria to respond to light

• **Color** – need it to make color

• Sensitivity

• Selectivity

• Speed

• Noise
Challenges

- **Light** – need bacteria to respond to light
- **Color** – need it to make color
- **Sensitivity** – need it to respond to a *short* pulse
- **Selectivity**
- **Speed**
- **Noise**
Challenges

- Light – need bacteria to respond to light
- Color – need it to make color
- Sensitivity – need it to respond to a short pulse
- **Selectivity** – don’t want it to respond to ambient light
- Speed
- Noise
Challenges

• Light – need bacteria to respond to light
• Color – need it to make color
• Sensitivity – need it to respond to a short pulse
• Selectivity – don’t want it to respond to ambient light
• **Speed** – want to see the color quickly
• Noise
Challenges

• Light – need bacteria to respond to light
• Color – need it to make color
• Sensitivity – need it to respond to a short pulse
• Selectivity – don’t want it to respond to ambient light
• Speed – want to see the color quickly
• Noise – don’t want to see random splotches of color
Conceptual Idea

• Emphasize light, color, and sensitivity
  • Old-fashioned camera: instant image capture, but overnight development

• Modular circuit: interchangeable parts
Light input $\rightarrow$ Signal amplification $\rightarrow$ Color output
LovTAP – Light-inducible repressor

Image from Strickland et al. PNAS 2008
Locking switch – Signal amplifier

- Based on Peking 2009 bistable switch

![Diagram](image)
Locking switch – Signal amplifier

- By default, cl434 levels are high
Locking switch – Signal amplifier

• Dropping cl434 levels flips the switch
Locking switch – Signal amplifier

- Dropping cl434 levels flips the switch
Locking switch – Signal amplifier

• Dropping cl434 levels flips the switch
Locking switch – Signal amplifier

• Once switch is flipped, it is stuck
Locking switch – Signal amplifier

- Once switch is flipped, it is stuck
mRFP1 – Strong color output

- mRFP1
- Fluoresces red but visible in ambient lighting
- Expressed from T7 promoter
Final circuit

In the light

- pTRP L
- cl434
- pRM
- cl
- TRP R
- T7 Polymerase
- mRFP1
- Final circuit

Legend:
- Red arrow: Represses
- Green arrow: Activates
BioBricks created

- ptrpL BBa_K588001
  - gctgtgacaattaatcataactagttactagtacgcaagttcag
  - Contains **SpeI** sites in **function critical locations**
  - Took UNAM’s idea, changed SpeI to Nhel

- trpR BBa_K588000
  - Cloned with standard RFC10 prefix and suffix
Results

- Two-plasmid circuit
- Fourteen-ligation plan
Results

Plasmid 1

Key
Green # = Completed
Red # = Not completed
Results

Plasmid 2

Key
Green # = Completed
Red # = Not completed
Application – Biosensor

Input

- Light
- Chemicals
- Radiation

Output

- Color
- Enzymatic activity
- Cell growth/Development
- Signaling

Locking switch
Adding binary numbers

\[
\begin{align*}
0 & \quad 0 & \quad 1 & \quad 1 & \quad \leftarrow A \\
+ 0 & \quad + 1 & \quad + 0 & \quad + 1 & \quad \leftarrow B \\
\hline
0 & \quad 1 & \quad 1 & \quad ? & \quad \leftarrow \text{Sum}
\end{align*}
\]
Adding binary numbers

\[
\begin{array}{cccccc}
0 & 0 & 1 & 1 & \leftarrow A \\
\hline
+ 0 & + 1 & + 0 & + 1 & \leftarrow B \\
\hline
0 & 1 & 1 & 0 & \leftarrow \text{Sum} \\
0 & 0 & 0 & 1 & \leftarrow \text{Carry}
\end{array}
\]

If Sum = 1, Carry = 0
then input = 0 + 1 or 1 + 0

If Sum = 0, Carry = 1
then input = 1 + 1
1-bit adder

\[ \text{Sum} = A + B \]

\[ \text{Sum} \quad \text{Carry} \]
3-bit adder

Adder 1
A
B
Carry
Sum

Adder 2
A
B
Carry
Sum

Adder 3
A
B
Carry
Sum
Adding binary numbers

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & + & 1 & + \\
0 & + & 0 & + \\
0 & + & 1 & + \\
0 & + & 0 & + \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{Carry In} & \text{A} & \text{B} & \text{Sum} & \text{Carry Out}
\end{array}
\]

Full adder

Sum = A + B + Cin

Sum Co
There are numerous ways to implement the truth table.
### Condensed truth table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Digital</th>
<th>Bacterial</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputs</strong></td>
<td><strong>Outputs</strong></td>
</tr>
<tr>
<td></td>
<td>Number of input signals</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Biological logic gates
Biological logic gates

AND

AND
SupD allows translation of T7ptag to yield functional T7 polymerase
Leucine zipper domains form dimers
Leucine zipper dimerization drives split protein reformation
Leucine zipper dimerization drives split protein reformation
Leucine zipper dimerization drives split protein reformation
Leucine zipper dimerization drives split protein reformation.
<table>
<thead>
<tr>
<th>Number of Input Signals</th>
<th>Output: Binary</th>
<th>Output: Bacteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>No Output</td>
</tr>
<tr>
<td>Number of Input Signals</td>
<td>Output: Binary</td>
<td>Output: Bacteria</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>No Output</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>RFP</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Input A**
  - Promoter A
  - RFP
  - czGFP
  - T7ptag

- **Input B**
  - Promoter B
  - RFP
  - czGFP
  - SupD

- **Input C**
  - Promoter C
  - RFP
  - nzGFP
  - czCFP
1 Input $\rightarrow$ RFP
1 Input $\rightarrow$ RFP

Promoter $A$ → RFP → czGFP → T7ptag
<table>
<thead>
<tr>
<th>Number of Input Signals</th>
<th>Output: Binary</th>
<th>Output: Bacteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>GFP and RFP</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Input A** + **Input C**
  - **Promoter A**
  - RFP
  - czGFP
  - T7ptag

- **Input B** + **Input C**
  - **Promoter B**
  - RFP
  - czGFP
  - SupD

**OR**
2 Inputs $\rightarrow$ RFP + GFP (czGFP + nzGFP)
2 Inputs $\rightarrow$ RFP + GFP (czGFP + nzGFP)
2 Inputs $\rightarrow$ RFP + GFP (czGFP + nzGFP)
<table>
<thead>
<tr>
<th>Number of Input Signals</th>
<th>Output: Binary</th>
<th>Output: Bacteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>GFP and RFP</td>
</tr>
</tbody>
</table>

Input A

Input B
2 Inputs $\rightarrow$ RFP + GFP (via T7 promoter)
2 Inputs $\rightarrow$ RFP + GFP (via T7 promoter)
2 Inputs $\rightarrow$ RFP + GFP (via T7 promoter)
2 Inputs $\rightarrow$ RFP + GFP (via T7 promoter)
<table>
<thead>
<tr>
<th>Number of Input Signals</th>
<th>Output: Binary</th>
<th>Output: Bacteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>11</td>
<td>CFP, RFP and GFP</td>
</tr>
</tbody>
</table>
3 Inputs $\rightarrow$ RFP + GFP + CFP
3 Inputs $\rightarrow$ RFP + GFP + CFP
3 Inputs $\rightarrow$ RFP + GFP + CFP
3 Inputs $\rightarrow$ RFP + GFP + CFP
3 Inputs → RFP + GFP + CFP
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Digital</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>Digital</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bacterial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
</tr>
<tr>
<td>Number of input signals</td>
</tr>
<tr>
<td>Outputs</td>
</tr>
<tr>
<td>Output signal</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
Inputs and outputs in our circuit

• Inputs
  – Promoters — plac, pbad, ptrpL

• Outputs
  – Reporters — RFP, GFP, CFP
Importance of logic and interchangeability of parts

Bacteria Adder

Bacteria Adder

Bacteria Adder

Bacteria Adder
Significance

• Complex reporter system
• Basic full adder $\rightarrow$ complex logic circuits
• Zipper domains $\rightarrow$ AND gates

BioBricks created

• N-terminal leucine zipper domain- BBa_K588100
• C-terminal leucine zipper domain- BBa_K588101
• Both with RFC25 fusion standard
Our Software Tool
MYS!S goals

• Check BioBricks for violations of assembly standards

• Perform codon usage optimization

• Output as user friendly lab protocol, using BioCoder
Assembly standard check

Sequence Name: BBa_K191006
Organism: Escherichia coli K12
Standard: Assembly Standard 10
Plasmid: pSB1A3

Original DNA Sequence
```
atgttgctactacatttgaacglatggagagaacttggccagataaccccattatatctgccgctccg
gatttctttgcagtgaagaatatagccgtgaaagaaattttggaagaaccctcggggttttctacaaggtcctgaatgatcgccgcagctgagaaattagagatgcctcagatgaatattatataaatgctgcaggtgtaaa
```

Primers

Modified DNA Sequence

Go!
Assembly standard check

Original DNA Sequence

Modified DNA Sequence

Primers

3' taaaacccttttggacatccaaagatgttccagg 5'
5' attttggaagaacagtgtagttttctacaaggtcc 3'
3' cgactaatcttttgagcggttttaaactactccgc 5'
5' gcctgatatagaagaaactgcggaaatattgatgagccg 3'
# Lab protocol

Protocol For Site Directed Mutagenesis

```c
#include "biocoder.h"

void main()
{
    start_protocol("Quik Change Site Directed Mutagenesis");

    // Materials needed
    Fluid forward Primer = new_fluid("Forward primer", "10uM");
    Fluid reverse Primer = new_fluid("Reverse primer", "10uM");
    Fluid biobrick Part = new_fluid("dsDNA template", "5 - 50ng or a 10X dilution from a miniprep");
    Fluid buffer = new_fluid("10x PCR Reaction Buffer");
    Fluid dNTP = new_fluid("dNTP mix");
    Fluid polymerase = new_fluid("PfuTurbo DNA polymerase");
    Fluid ddH2O = new_fluid("Distilled Water");
    Fluid DpnI = new_fluid("DpnI restriction enzyme", vol(1, UL));
    Fluid competent cells = new_fluid("Chemically competent cells", vol(50, UL));
    Plate plate = new_container("Plate made with Ampicillin antibiotic");

    // Containers needed
    Container pcr_reaction tube = new_container(STERILE PCR TUBE);
    Container comp cell tube = new_container(SCREW CAP TUBE, competent cells);

    // Steps needed to complete protocol
    first_step("PCR reaction mix");
    char* tubes[1] = {"PCR Reaction for Quik Change"};
    Volume volume[6] = {vol(1, UL), vol(1, UL), vol(1, UL), vol(5, UL), vol(1, UL), vol(XVAL, UL)};
    mixing_table[2, 7, fluid array, tubes, volume, vol(50, UL), pcr reaction tube];

    next_step();
    measure_fluid(polymerase, vol(1, UL), pcr reaction tube);

    next_step("PCR conditions");
    pcr init semiphase(pcr reaction tube, 95, time(30, SECS));
    thermocycle(pcr reaction tube, 12, 95, time(30, SECS), 53, time(60, SECS), 68, time(120, SECS), NORMAL);
```
BioCoder
A programming language for biology protocols

C++ code

```cpp
// 95% ethanol precipitation of samples for purification
next_step();
measure_fluid(sdw, vol(16, UL), rxn_tube);
measure_fluid(eth95, vol(2, ML), rxn_tube);
tap(rxn_tube);
store_for(rxn_tube, RT, time(15, MINS));
```

BioCoder

5. Add 16 µl of sterile distilled water. Add 2 ml of 95% ethanol. Gently tap the mixture for a few secs. Store at room temperature for 15 mins.
BioCoder

- **MYS!S** automatically creates **C++ code**
- **BioCoder** uses code to **generate web pages** with lab procedures
Future features of MYS!S

• More powerful algorithms to optimize protein output

• Capability to model RNA structure both in 2D and 3D

• Expand the laboratory procedures MYS!S supports to include all that are required for BioBrick assembly

• Be able to give a wet lab team a plan for the construction of an entire genetic circuit
RU iGEM Team

Team Members:
Anish Vaghela, Kevin Lu, Apexa Modi, Sana Wajid, Akanksha Arya, Stephanie Fairclough, Boris Joffe, Illja Melentijevic, Ethan Stern, Neil Patel, Vatsal Bhatt, and Michael Verano

Our Advisors:
Dr. Andrew Vershon, Dr. James Stapleton, and Dr. Ryan Golhar

Our Sponsors:
Questions?