Title

Kinetic filtering: Enumerating biochemical circuits that distinguish transient and sustained stimulation

Author

Gerardin, Jaline

Publication Date

2013

Kinetic filtering: Enumerating biochemical circuits that distinguish transient and sustained stimulation

by

Jaline Gerardin

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Biophysics

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, SAN FRANCISCO

[redacted]

To my family, friends, and mentors.

Acknowledgments

As with any major endeavor, the list of people to thank is long. Three and a half years ago, Wendell Lim rushed over to my desk waving a paper on T cell receptors. I was in the midst getting mired in a previous project that was quickly heading nowhere, so I was excited to try something new. Wendell was just as excited, and his belief in the importance of this project has propelled us into some really interesting work on how cells can measure time. So thanks to Wendell for keeping his eyes on the long-term vision of the project and pushing me to answer the big questions.

A number of other faculty have also provided many insights that have shaped the work in this thesis. Chao Tang participated in a number of early discussions and pointed us in the right direction in quantifying the behavior we were studying. My thesis committee members Wallace Marshall and Hana El-Samad engaged me in some lively back-and-forth and generated interesting avenues to explore. Matt Jacobson provided the clear-eyed perspective of a scientist outside the field.

When I first joined the Lim lab, Angi Chau showed me the ropes and taught me everything I needed to know about running thousands and thousands of simulations. Even after I stopped working on her polarization study to begin my own projects, she was always happy to chat about new data and simulation snags, no matter how minor. Her insights, enthusiasm, and encouragement were invaluable in the early stages of this project.

Other members of the Lim lab have been similarly generous with their time and brainpower. Russell Gordley and I have spent many hours discussing everything from rate-balance plots to possible experimental designs. Talking with Jared Toettcher generated helpful theoretical ideas as well as pie-in-the-sky simulation ideas that were ultimately outside the scope of this project but would have been an interesting alternative way to address the problem. Jessica Walter, Hyun Youk, Jesse Zalatan, Reid Williams, Amir Mitchell, Wilson Wong, Caleb Bashor, and Andy Wei have also contributed critical feedback in one form or another, and I am very grateful to them. The Lim lab is composed of wonderful people and I'm very glad to have met and worked beside all of you.

Many people in the wider UCSF community have provided their insights and comments over the years. The UCSF Center for Systems and Synthetic Biology has been a great forum for meeting interested scientists, and I'm particularly glad to have had the opportunity to exchange ideas with David Soloveichik and Matt Thomson. Over the years people in the Tang lab have also been very helpful, especially Wenzhe Ma and Xiaojing Yang.

Matt Jacobson, Rebecca Brown, and the Biophysics Graduate Program have been essential in helping me navigate toward graduation. A big thanks to both of you for going above and beyond in practical and emotional support. I'd also like to thank Joe DeRisi and Jonathan Weissman, the two members of my orals committee not already mentioned, for passing me even though I ditched my orals project the first chance I could.

Two more mentors deserve special mention here. Dana Pe'er, your firm belief in me has been more encouraging than I can say. Cindy Stokes, thank you for taking a chance on me and giving me the opportunity to expand my horizons.

Lastly, I'd like to thank all the people who provided vast amounts of intangible support. I really couldn't have done it without you. My classmates, including Jenny Hsiao, Dan Lu, David Booth, Monica Tremont, Brittany Belin, David Burkhardt, and Monica Guo, are great people and scientists, and I feel fortunate to have gotten to know you. Heartfelt thanks to Angi Chau and Jess Walter, you know why. Thanks also to Andrea Spillmann, Kapil Amarnath, Monisha Santamaria, Vania Cao, and Jack Wang for being there for me and providing distractions when I needed them. Gabriel Rocklin has been a reliable source of jokes, scientific insights, and cheerleading, and I'll leave it at that.

Thank you to my family for a constant source of inspiration: Hadia Gerardin for showing that it can be done, Ylaine Gerardin for leading by good example, and my mother for pushing me to be the best I can be.

Kinetic filtering: Enumerating biochemical circuits that distinguish transient and sustained stimulation

Jaline Gerardin

Abstract

Many cellular processes require telling time. Cells may use kinetic filtering, the ability to respond to sustained but not transient stimulation, to decode information stored in dynamical profiles and exert control over relative timing of events. Previous work has found only one circuit architecture, the AND-gated coherent feed forward loop, capable of such behavior. We enumerate the space of 1-, 2-, and 3-node networks to ask what additional architectures can drive kinetic filtering, which is quantified by measuring temporal dose response steepness and trigger time. We find two types of positive feedback architectures, two types of buffering architectures, and the coherent feed forward loop with AND logic are the only kinetic filters of 3 or fewer components. The buffering double inhibition circuit architecture is modular, reversible, capable of the widest range of trigger times, and most suitable as a template for future engineering efforts.

Table of Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

OVERVIEW

The relationship between structure and function of cellular signaling circuits remains an intriguing problem. In this chapter, we introduce the concept of kinetic filtering, the ability of a signaling circuit to distinguish between transient and sustained stimuli. We describe our approach to finding design principles behind circuits capable of kinetic filtering behavior and briefly outline our findings.

1.1 CIRCUIT STRUCTURE AND CIRCUIT FUNCTION

Cells experience a multitude of internal and external stimuli, and signal transduction networks process sensed information into the appropriate response (Figure 1.1)[1]. Receptors on the cell surface, for example, can be activated by the presence of specific molecules. Decision-making biochemical circuits transduce the signal from the receptor to response machinery that modulates gene expression, changes cell shape, or triggers other behavior. For many years, scientists have sought to fully understand how cells use biochemical circuits to carry information and make decisions.

By studying extremely simplified systems, theoreticians have shown that network elements such as positive feedback and biochemical mechanisms such as cooperativity are responsible for observed behavior like bistability and all-or-none switching[2-4]. Signaling systems with the same connectivity but different dynamical behavior, such as the mitogen-activated protein kinase (MAPK) cascades found in a variety of eukaryotic response pathways, have been observed to generate vastly different responses[5-7].

The relationship between signaling circuit architecture and behavior is of great interest in a variety of biological contexts. In addition to contributing towards basic science goals of understanding biology, studying signal transduction networks has many beneficial applications. In many cancer systems, cells proliferate because their signaling circuits are constitutively on instead of responding only when proliferation cues are present; drug targeting of overactive signaling components has been remarkably successful as treatment[8]. Artificially turning on certain signaling networks may aid in regenerative medicine. In synthetic biology efforts, engineers need blueprints upon which to design and construct circuits that respond to a specific signal in a specific way[9,10].

The usual approach to studying signaling circuits consists of investigating a particular biological circuit in great detail and dissecting its biochemical features in their entirety. This “top down” approach is suitable for understanding specific circuits, but it is difficult to single out core behavioral machinery amidst all the details. In addition, even after data for many systems have been gathered, identifying unifying features across similarly behaving systems each with many complex components remains a challenge.

Instead of studying one particular biological circuit, we take the “bottom up” approach, beginning with simple, generalized circuits and seeking to discover the core circuit elements that drive a specific behavior. The past ten years have seen an explosion of theoretical and computational work on simple circuits linking biochemical circuit structure to biological function, including studies of oscillations[11,12], noise filtration[13,14], and temporal expression programs[15]. These investigations have typically focused on a single circuit structure or small family of related circuits and described their behavior in quantitative detail. We look at the problem from the opposite end: begin with a biological behavior and characterize the signaling circuits that can give rise to it.

In order to definitively identify all simple circuits capable of a biological behavior, we exhaustively search the space of low-complexity circuits in an unbiased manner. By studying simpler circuits of fewer components, we can more easily identify the key circuit elements responsible for that particular behavior. Circuit elements that add robustness or redundancy can be eliminated. Parameters can be individually tuned and their effects measured.

Several studies in recent years have used enumeration with great success to search for circuits with specific behaviors in an unbiased manner. Previous work has described circuits capable of switching[16], segmentation[17], perfect adaptation[18], and spatial polarization[10].

Systematically searching circuit space has two main benefits. First, we can be confident that we have identified the complete set of circuits with our behavior of interest, and circuits outside that set are guaranteed not to exhibit the behavior. Second, we can potentially uncover architectures that can perform a biological function but have not been found in actual biological systems. While these novel architectures may be previously unobserved because they fail at some level in vivo, it is also possible that evolution has overlooked them by chance and they would still be functional in an engineered context.

Enumerative search is not the only way to computationally identify circuits capable of biologically relevant behavior. Genetic algorithms have also been used to find networks that oscillate[19], adapt to stimuli[20], and drive segmentation[21]. While computational resources limit enumerative search to small networks, genetic algorithms have the freedom to evolve larger, more complex networks that may be capable of more sophisticated behavior. Where genetic algorithms simulate an evolutionary path to networks of greater and greater fitness, in enumerative search all networks are considered independently, making fitness comparisons between networks more challenging. However, because we aim to exhaustively identify core circuit elements that drive behavior, for our purposes enumerative search is preferred to genetic algorithms.

In this work we seek to identify circuits capable of distinguishing transient and sustained stimuli. We envision a "periodic table" of signaling circuits, grouping simple circuit architectures together by the type of functionality they can generate (Figure 1.2).

1.2 KINETIC FILTERING ALLOWS CELLS TO MEASURE TIME

How do cells tell time and measure the duration of events? From coordination of developmental processes to filtering of spurious signals, it is often critical for cells to be able to measure time (Figure 1.3). Signaling components immediately downstream of receptors must filter noisy, transient environmental fluctuations from real, sustained signals[22]. In addition, recent work has highlighted the role that activation duration can play in maintaining input-output fidelity in circuits that reuse the same components[23,24]. Cells can encode information in response dynamics, where decoding modules translate transient or sustained dynamics into cell fate decisions. Measuring stimulation time also allows cells to respond to signals only after a certain delay and coordinate events to happen in a specific sequence, a critical ability in processes such as the cell cycle.

Several biological systems have been observed to use duration of signaling events in determining cell fate. In rat neuronal precursor cells (PC-12 cells), kinetic filtering allows cells to reuse the same signaling components in response to different inputs (Figure 1.4 left). Growth factors EGF and NGF both stimulate mitogen-activated protein kinase (MAPK) pathways that lead to ERK activation[25]. However, EGF stimulation results in transient ERK activation and eventual cell proliferation, while NGF stimulation leads to prolonged ERK activation and cell differentiation. If EGF stimulation is re-engineered to cause prolonged ERK activation and NGF to cause transient ERK activation, then outcomes are switched: EGF stimulation results in differentiation and NGF stimulation results in proliferation[23]. The circuitry thought to measure the duration of ERK activation consists of a feed-forward motif, where activated ERK induces expression of a transcription factor whose activity is dependent on the continued presence of active ERK[26].

In the eukaryotic DNA damage response pathway, p53 is a transcription factor that activates a suite of stress response genes[27]. Recent studies of p53 dynamics has found that gamma irradiation, which induces double-stranded breaks in DNA, results in a pulsatile activation of p53, while UV irradiation, which exposes single-strand DNA, results in sustained activation of p53[28,29] (Figure 1.4 right). Ultimately, pulses of p53 induce cell cycle arrest while sustained p53 induces apoptosis; however, the mechanism by which the cell "reads" p53 dynamics remains unknown.

Recent work on sporulation in *Bacillus subtilis* has shown that cells may also measure time by initiating an oscillator upon stimulation, "counting" the number of oscillations, and setting the timer off after a threshold number has been reached[30]. In most other systems, however, there is little evidence for oscillations, leading us to suspect that a different mechanism is responsible for measuring time.

1.3 ENUMERATING CIRCUITS TO SEARCH FOR KINETIC FILTERING BEHAVIOR

To date there has been no systematic study of which signaling circuit architectures can kinetically filter stimuli and measure time. Even in biological circuits discovered to measure duration of certain signaling events, the precise circuitry responsible for the measurement is often not known.

Multi-staged cascades of reversible reactions have been shown to be capable of kinetic filtering in the limit of very long cascades[31,32]. The only non-cascading circuit with known kinetic filtering ability is the coherent feed forward loop with AND logic, found by a comprehensive study of feed forward architectures in transcriptional regulation[33,34]. In this architecture, input activates expression of two output-activating transcription factors, one of which also activates expression of the other. Presence of both transcription factors is necessary for output expression (AND logic); thus, the coherent feed forward loop only responds to inputs with duration long enough for signal to reach the output node via both arms of output node activation.

We ask what non-oscillatory circuit architectures, with complexity on par with or simpler than the coherent feed forward loop, are also kinetic filters that allow cells to time the duration of signaling events. As discussed above, enumerative search is an excellent method for characterizing the entire space of small circuits capable of kinetic filtering.

We introduce the concept of "temporal ultrasensitivity" to quantify a signaling network's kinetic filtering behavior. Analogous to concentration-based ultrasensitivity, temporal ultrasensitivity implies a steep temporal dose response curve where inputs with duration shorter than the trigger time result in minimal or no activation of the network and inputs longer than the trigger time result in maximal activation. A temporally ultrasensitive circuit reports on input duration by responding all-or-none to stimuli with duration longer or shorter than the trigger time, respectively. Given that many biological responses happen at timescales faster than transcription, we have chosen to focus on enzymatic circuits as a model system for studying kinetic filtering (details discussed in Chapter 2).

We apply the enumeration method to signaling networks of 1, 2, and 3 nodes, searching for the simplest core architectures capable of driving temporally ultrasensitive behavior. While previous enumeration efforts have tended to ignore the effect of node logic on circuit behavior, here we have allowed each node to combine regulations of the same sign using OR or AND logic (see Chapter 2), which model independent and cooperative interactions respectively.

The enumerative search identified thousands of circuits of 1, 2, or 3 nodes as robust kinetic filters (Chapter 3). By clustering circuits by phenotype, we sorted the kinetic filters into five groups with common structural features within each group: two types of positive feedback architectures, two types of buffering architectures, and the previously uncovered coherent feed forward loop with AND logic (Chapter 3). In addition to differing by phenotype, the five types of kinetic filters differ mechanistically and with respect to their trigger time (Chapter 4).

While the coherent feed forward loop with AND logic is the only circuit with experimentally derived biological examples, other types of kinetic filters are also found in biological settings where telling time may be critical (Chapter 5). The buffering double inhibition topology is particularly attractive as a blueprint for constructing synthetic kinetic filters (Chapter 5).

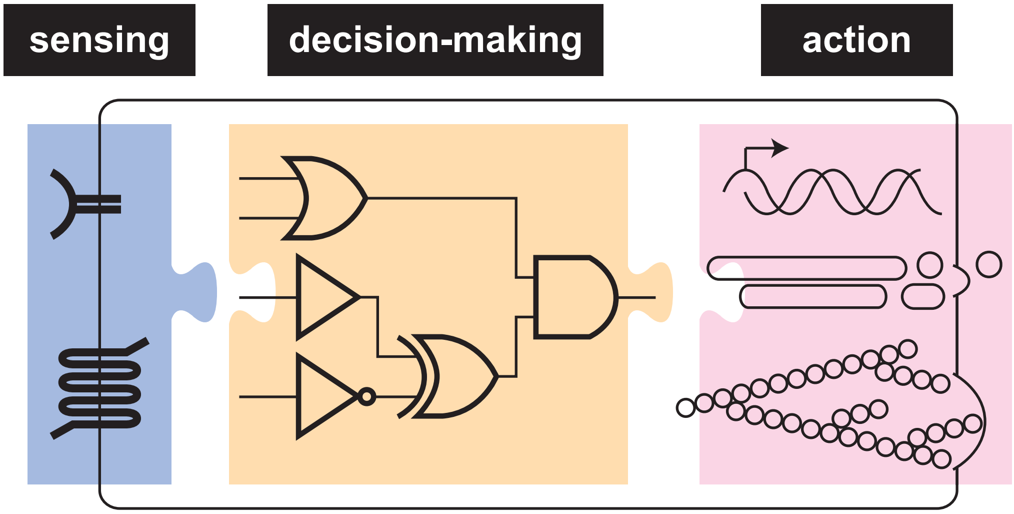


Figure 1.1 . Signal transduction networks translate sensed information into cell action.

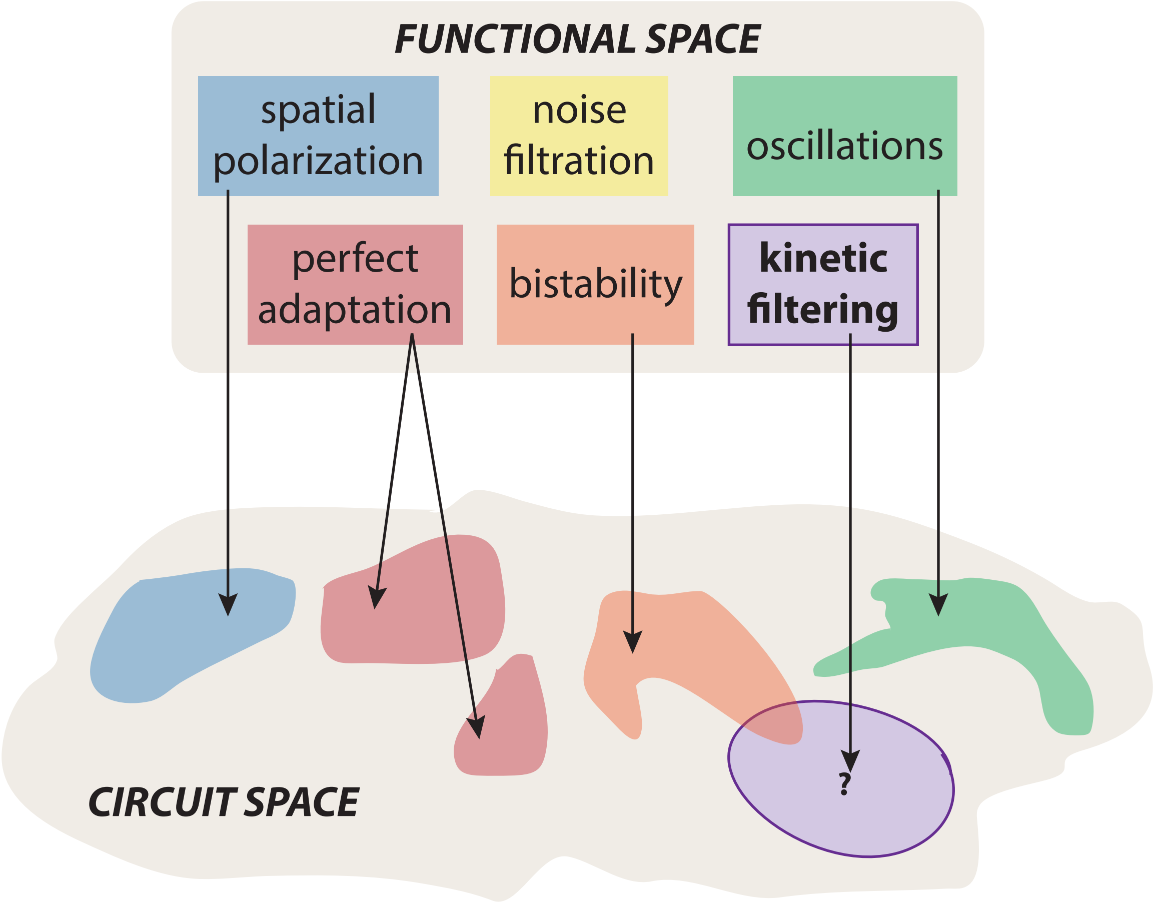


Figure 1.2. Mapping between the space of biological circuit behaviors and network architectures.

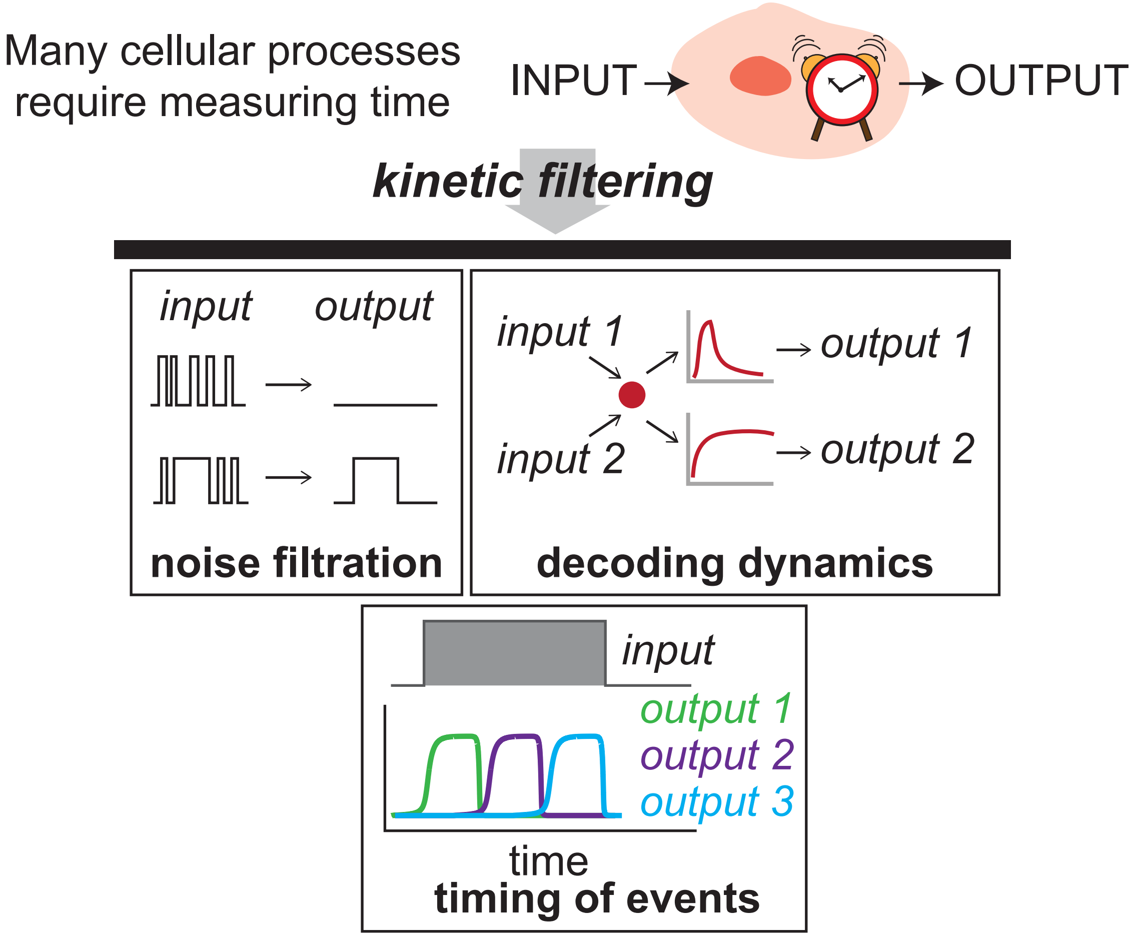


Figure 1.3 . Kinetic filtering allows cells to distinguish between transient and sustained stimuli and can drive noise filtration, dynamics decoding, and coordination of events in time.

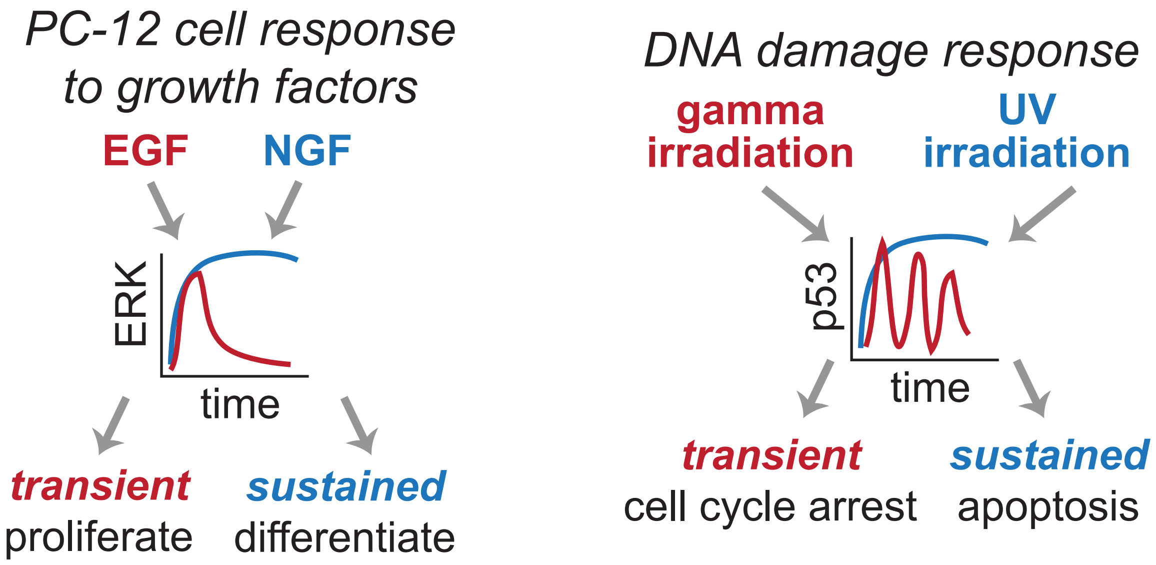


Figure 1.4. Kinetic filtering can be used to maintain input-output fidelity.

Chapter 2

SIMULATION METHODS

OVERVIEW

To identify the simplest kinetic filtering circuits, we enumerated the space of 1-, 2-, and 3-node enzymatic circuits, allowing each node to operate with OR or AND logic, and quantified each topology's performance by measuring its temporal ultrasensitivity score and trigger time. Kinetic filters are circuits with temporal ultrasensitivity score ≥ 0.5 and trigger time ≥ 1, and kinetic filtering topologies are robust when at least 10 out of 10,000 sampled parameter sets result in kinetic filtering behavior. Regulatory interactions were simulated under enzymatic conditions using total quasi-steady state Michaelis-Menten equations.

2.1 ENUMERATION

Our approach to finding kinetic filtering circuits involves systematic screening of a library of circuit topologies with 1, 2, or 3 nodes and sampling 10,000 parameter sets for each topology (Figure 2.1). Each circuit's kinetic filtering behavior was quantified by subjecting the circuit to inputs of varying duration.

To enumerate circuit topologies, we allowed each node to use OR or AND logic to combine regulations and each link to be positive, negative, or absent (Figure 2.2). We discarded topologies where the input signal does not reach the output node. Circuits with regulations on a non-input, non-output node that does not in turn regulate another node were also discarded. For AND logic topologies, we discarded all circuits where the node with AND logic does not have two regulatory links of the same sign, counting input as a positive regulation. A total of 68,705 topologies were included in our enumerated circuit space.

Due to computational limitations, we only enumerated systems with one, two, or three nodes. A four-node system contains on the order of 43 million topologies before considering node logic, making an unbiased search of 4-node circuit space prohibitively expensive.

2.2 PARAMETER SAMPLING

Up to 26 parameters were sampled for each circuit: k_{cat} and K_m for each of the up to 9 possible circuit links, 3 possible constitutive activators and deactivators, and the input link. All parameter samplings used the Latin hypercube method[35] with range 0.1 to 10 for k_{cat} and 0.001 to 100 for K_m; this range is roughly physiological with units of seconds and μM. 10,000 parameter sets were sampled for the enumerative search and 100,000 for determining parameter regime restrictions.

2.3 MODELING ENZYMATIC CIRCUITS

Reactions were modeled with total quasi-steady-state Michaelis-Menten kinetics (tQSS-MM)[36-38]. While the usual Michaelis-Menten steady-state approximation assumes excess of substrate compared to small enzyme concentrations, in our system this assumption fails because every node can act as both an enzyme and a substrate. Compared to the Michaelis-Menten model, the tQSS model treats enzyme and substrate on more equal footing, tracking the total concentration of substrate rather than the concentration of free substrate, and is appropriate in a wider range of biochemical situations while still allowing rapid simulation compared to mass action. See Figure 2.3 for example equations.

Nodes were converted between active and inactive states according to network linkages, where positive regulations catalyze activations and negative regulations catalyze deactivations (Figure 2.3). The total concentration of each node was held constant at 1, and only the active fraction of each node could catalyze other reactions. For nodes operating under OR logic, tQSS-MM expressions for incoming links were added. For nodes operating under AND logic, tQSS-MM expressions for incoming links of the same sign were multiplied, and expressions for incoming links of opposite signs are added. In systems of up to three nodes, no node can have two or more activators and two or more deactivators, so if a node was acting under AND logic, either the activators or the deactivators used AND logic but not both.

In addition to the regulations between nodes A, B, and/or C, a circuit had additional constitutive activators and deactivators as needed such that no node had only activators or only deactivators. Constitutive activators and deactivators had constant concentration of 0.1.

Each circuit was numerically integrated using a fifth-order embedded Runge-Kutta formula with an adaptive stepsize controller[39]. Active concentrations of each node were initialized to 0.1 and allowed to come to steady state before the application of input. If the circuit could not initialize to a steady state, for example by showing oscillating behavior, that circuit was discarded and additional simulations were not performed. After input was removed, the system was again allowed to reach steady state before output characteristics such as final value and maximum amplitude were measured. See Table 2.1 for a description of simulation software; code and additional details are contained in Appendix A.

Code	Description
simulate.cc	Given a set of parameters and input characteristics, simulates a circuit's response to a single pulse of input
doseResponseMetrics.cc	Given a set of inputs and responses, calculates steepness and EC50 of that dose response curve
runSimulations.py	Simulation manager that reads in a list of parameters, calls simulate.cc for various input durations, and calls doseResponseMetrics.cc to calculate metrics of kinetic filtering

Table 2.1. Key software for simulating large numbers of circuits at many input durations and calculating metrics for kinetic filtering. Code is contained in Appendix A.

2.4 QUANTITATIVE DEFINITION OF KINETIC FILTERING

Kinetic filtering circuits are those that respond maximally to long inputs and minimally to short inputs. We quantified a circuit's kinetic filtering ability by subjecting the circuit to inputs of different duration but identical constant amplitude, measuring the resulting output amplitudes and constructing a temporal dose response curve, and using that curve to infer the circuit's degree of temporal ultrasensitivity (Figure 2.4).

Input pulses of duration 0.25, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 800, 1000, 2000, 3000, 5000, 6000, 8000, 10000, 20000, and 50000 seconds were applied separately to each parameter set of each topology. Input amplitude was always 0.1. Maximum output amplitude was measured over the period covering both the duration of the input pulse as well as a post-pulse recovery period lasting until the system came to steady state. Circuits that failed to reach steady state within 86,400 simulation seconds of recovery time were rejected from consideration. Inverting circuits whose output decreased upon application of input were also discarded to save overall computation time.

Kinetic filtering ability was quantified by plotting the circuit's maximum output amplitude in response to each duration of input and measuring the temporal ultrasensitivity score (TU score) of the resulting curve (Figure 2.4). TU score was defined as the ratio of input duration yielding 10% of maximum response to input duration yielding 90% of maximum response, analogous to cooperativity score in classical dose response curves[3]. The 10% and 90% input durations were determined by interpolating a linear fit between the simulated input durations bracketing the 10% and 90% response amplitudes respectively.

A kinetic filter's trigger time was defined as the input duration that yielded 50% maximum output amplitude and qualitatively delineates what the circuit sees as "long" versus "short" input. The trigger time was calculated by linear fit between input durations bracketing 50% maximum output amplitude (Figure 2.4).

Maximal response (Rmax) and difference between maximal and minimal response (ΔR) were also measured for each circuit (Figure 2.4). Circuits with Rmax < 0.001 or ΔR/Rmax < 0.5 were rejected prior to analysis due to showing respectively too little response to input or too little difference in output with respect to input duration.

Kinetic filters were defined as those circuits with TU score ≥ 0.5 and trigger time ≥ 1. A topology's robustness was defined as the fraction of the topology's sampled parameter sets that were kinetic filters. For a topology to be a robust kinetic filter, we required a minimum robustness of 0.0010 or 10 out of 10,000 sampled parameter sets to exhibit kinetic filtering behavior. Robustness measurements across independent parameter samplings correlated well but not well enough to directly compare robustness between two topologies with robustness within 0.0020 of each other (Figure 2.5).

2.5. MEASUREMENT OF OTHER CIRCUIT BEHAVIORS

In addition to the basic measurements of kinetic filtering behavior described above, we also measured other behaviors of circuits already found to be kinetic filters. Rationale, results, and conclusions are presented in Chapter 4, but we describe the simulation details below.

2.5.1 Clustering kinetic filters by phenotype

We used only the kinetic filtering parameter sets of minimal kinetic filtering topologies of 1-, 2-, or 3-nodes found in the large enumeration study for this analysis. For each of these parameter sets, we applied one pulse of duration 50,000 to measure 5 phenotypic metrics: long-term memory, on timing, on steepness, off timing, and off steepness (Figure 2.6). Trigger time, described above, was the sixth phenotypic metric.

Long-term memory was quantified by taking the ratio of final output level after input pulse has been removed to the maximum output level achieved. Circuits with long-term memory turn on and stay on after input has been removed, and this ratio is close to 1. In contrast, circuits that turn off once input has been removed have this ratio close to 0.

The on timing phenotypic metric was defined simply by the time for output to reach 50% maximum amplitude after input has been turned on. On steepness was the ratio between the time needed for output to reach 10% maximum amplitude to the time to reach 90% maximum amplitude, analogous to the quantification of temporal ultrasensitivity (temporal dose response steepness) described above.

The off timing and off steepness metrics were not measured for circuits that do not turn off after input is removed. Off timing was the time needed for output to decrease to 50% maximum amplitude after input has been removed. Off steepness was the ratio between the time needed for output to reach 90% amplitude to the time to reach 10% maximum amplitude.

Trigger time, on timing, and off timing were transformed by taking base 10 logarithms on the calculated value. All metrics were subsequently normalized to range [-1, 1]. Phenotypic clustering was performed using the Cluster software program by clustering genes using hierarchical clustering, euclidean distances, and centroid linkage. Principal components were calculated using the SciPy linalg.svd singular value decomposition package. For data sets without off timing and off steepness metrics, those metrics were set to the mean value of off timing and off steepness respectively.

2.5.2 Quantification of tunability

For this analysis, we used kinetic filtering parameter sets identified in the 100,000 parameter set sampling of the 5 kinetic filtering prototypes. Every parameter of each parameter set, other than k_{cat} and K_m of the input acting on node A, was scaled by a factor of 0.1 and 10 in turn. For each parameter scaling, TU score and trigger time were measured. Circuits with TU score < 0.4 were rejected as no longer exhibiting kinetic filtering behavior. Tunability of each parameter was calculated by averaging the resulting trigger time for tuning that parameter and multiplying by the fraction of tunings that resulted in a circuit the TU score ≥ 0.5. Tunable parameters were those with tunability ≥ 1.5.

2.5.3 Quantification of reset time

Only the kinetic filtering parameter sets of 1-, 2- and 3-node minimal kinetic filtering topologies found in the large enumeration study were used in this analysis. For each parameter set, the input duration resulting in 90% maximum output amplitude was determined as described above. This input duration was delivered in two pulses of equal duration with separation 0.25, 0.5, 1, 2.5, 5, 10, 25, 50, 100, 250, 500, 1000, 5000, 10000, or 50000 seconds. Maximum output amplitude was measured for each separation duration. The reset time, the pulse separation yielding 50% maximum output amplitude, was measured with the same method described above for trigger time.

2.5.4 Quantification of integration ability

Only the kinetic filtering parameter sets of 1-, 2- and 3-node minimal kinetic filtering topologies found in the large enumeration study were used in this analysis. Temporal dose response curves for input amplitudes of 0.01, 0.03, 0.05, 0.07, 0.1, 0.3, 0.5, 0.7, 1, 3, 5, 7, and 10 were constructed for kinetic filtering parameter sets of minimal kinetic filtering topologies, and trigger times for each amplitude were measured as described above. Simulations resulting in TU score < 0.4, Rmax < 0.001, and/or ΔR/Rmax < 0.3 were discarded. Circuits with fewer than 3 remaining threshold measurements were also discarded. For the remaining circuits, slopes of linear fit of log threshold and log input amplitude were calculated with the polyfit function of the Numpy Python package.

2.6 MODELING TRANSCRIPTIONAL AND MIXED REGULATION CIRCUITS

Transcriptionally regulated links were modeled as two steps, transcription and translation, with regulation occurring at the transcriptional level. Activators and repressors modified mRNA synthesis levels in the standard manner[40,41] with sampled cooperativity, and basal synthesis was set to 0. Translation was modeled with first-order dependence on mRNA levels; both mRNA and protein levels decayed as first-order processes. Input remained an enzymatic process, phosphorylating node A; phosphorylated node A could then continue to act as an activator or repressor of downstream transcription.

Parameters were sampled using Latin hypercube with the following ranges. Enzymatic parameter ranges were identical to those described above for purely enzymatic circuits. Maximum transcription rate and mRNA decay rate each ranged from 0.001 to 10; decay rate was the same for all mRNAs in the same circuit. Km's of activator and repressor binding ranged from 0.001 to 100. Hill coefficients were sampled at 1, 2, and 8. Translation rate constant and protein decay rate ranged from 0.01 to 10, and all proteins in the same circuit decayed at the same rate.

Equations were integrated using the odeint function of the integrator function of the Scipy Python package. Input durations used were the same as for enzymatic circuits above, and temporal ultrasensitivity score and trigger time were calculated as for enzymatic circuits.

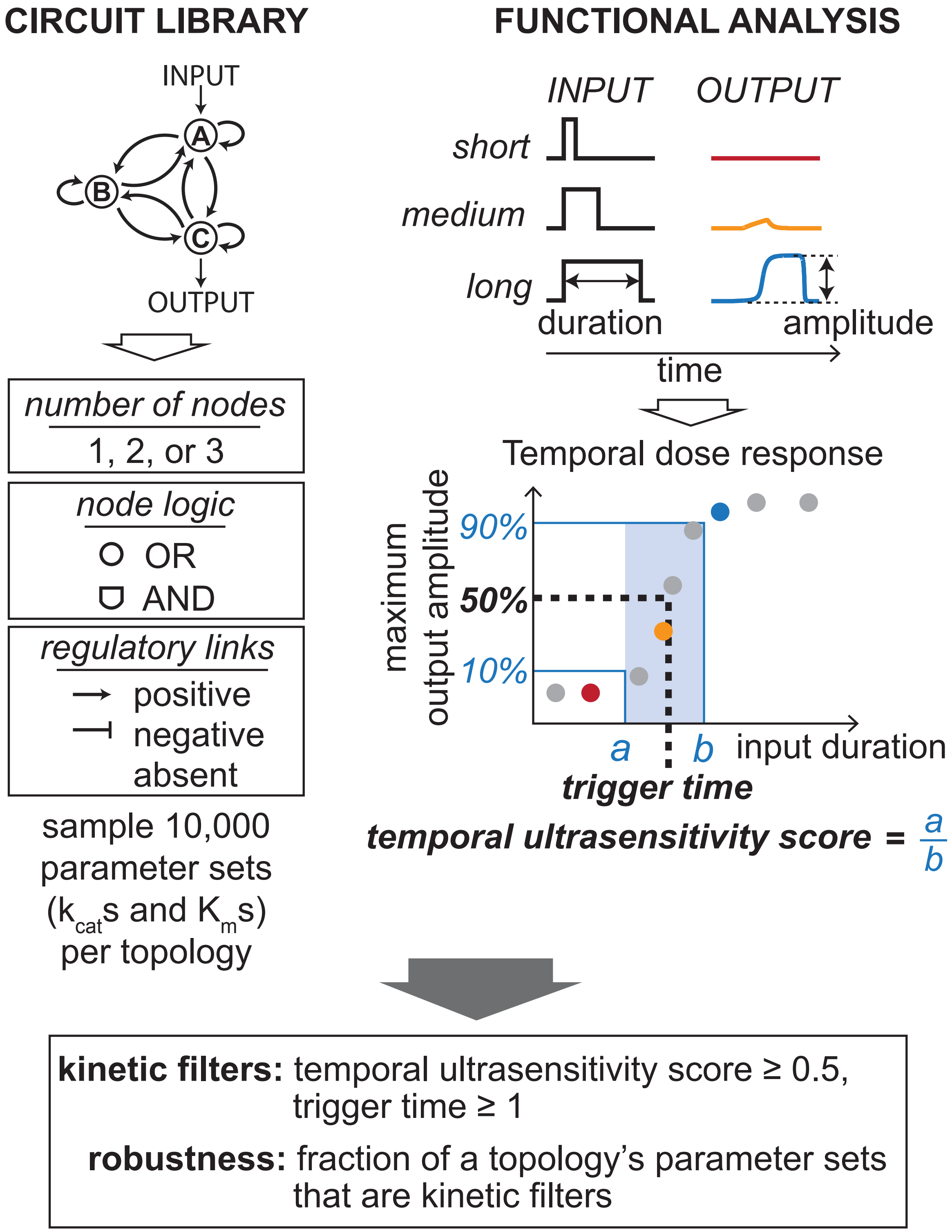

CIRCUIT LIBRARY
INPUT
A
B
C
OUTPUT
number of nodes
1, 2, or 3
node logic
OR
AND
regulatory links
positive
negative
absent
sample 10,000 parameter sets (k_{cat}s and K_ms) per topology
FUNCTIONAL ANALYSIS
INPUT
OUTPUT
short
medium
long
duration
amplitude
time
Temporal dose response
maximum output amplitude
90%
50%
10%
a
b
input duration
trigger time
temporal ultrasensitivity score = $\frac{a}{b}$
kinetic filters: temporal ultrasensitivity score ≥ 0.5, trigger time ≥ 1
robustness: fraction of a topology's parameter sets that are kinetic filters

Figure 2.1. Enumerate a library of circuits and measure temporal dose response curves to identify kinetic filters.

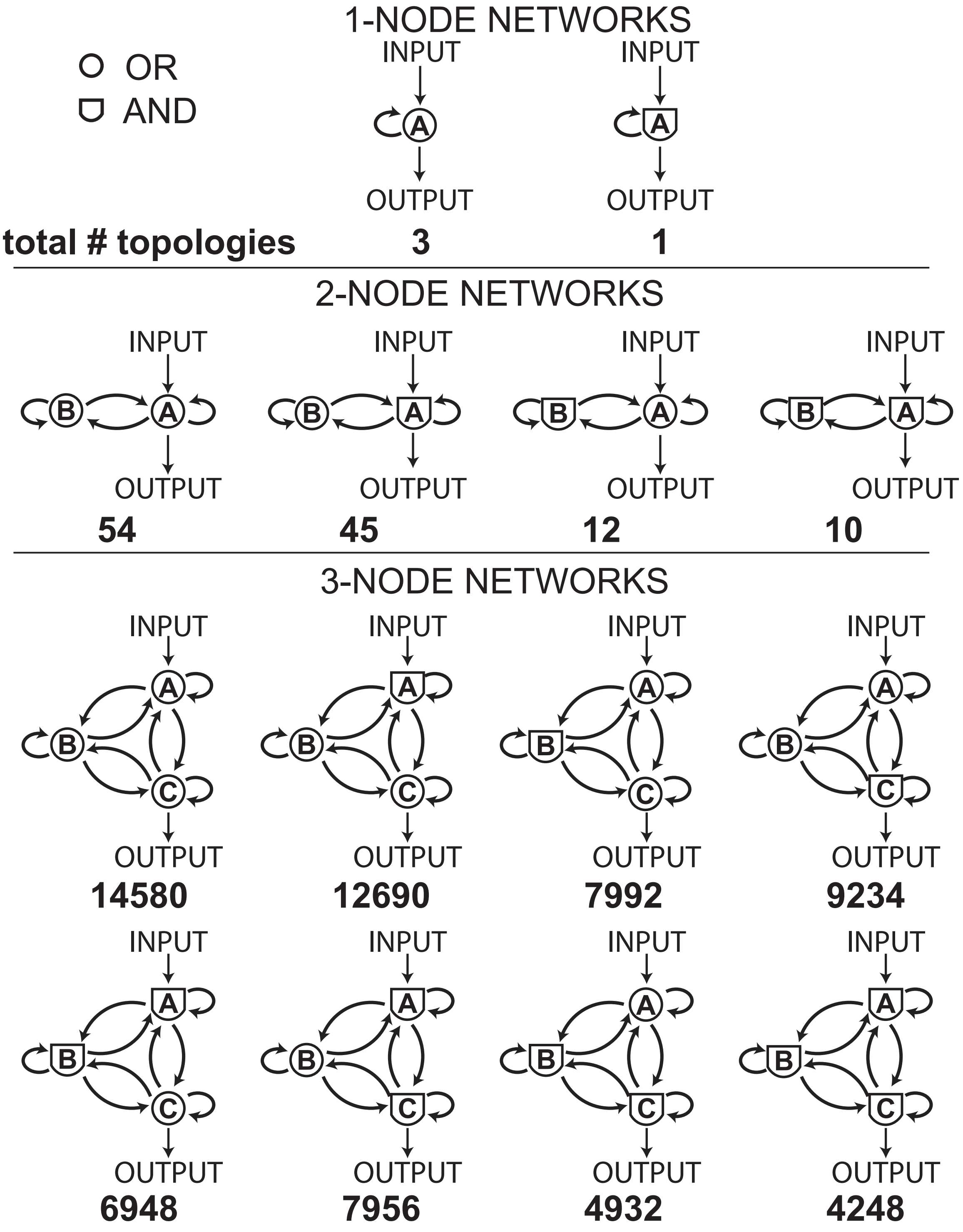
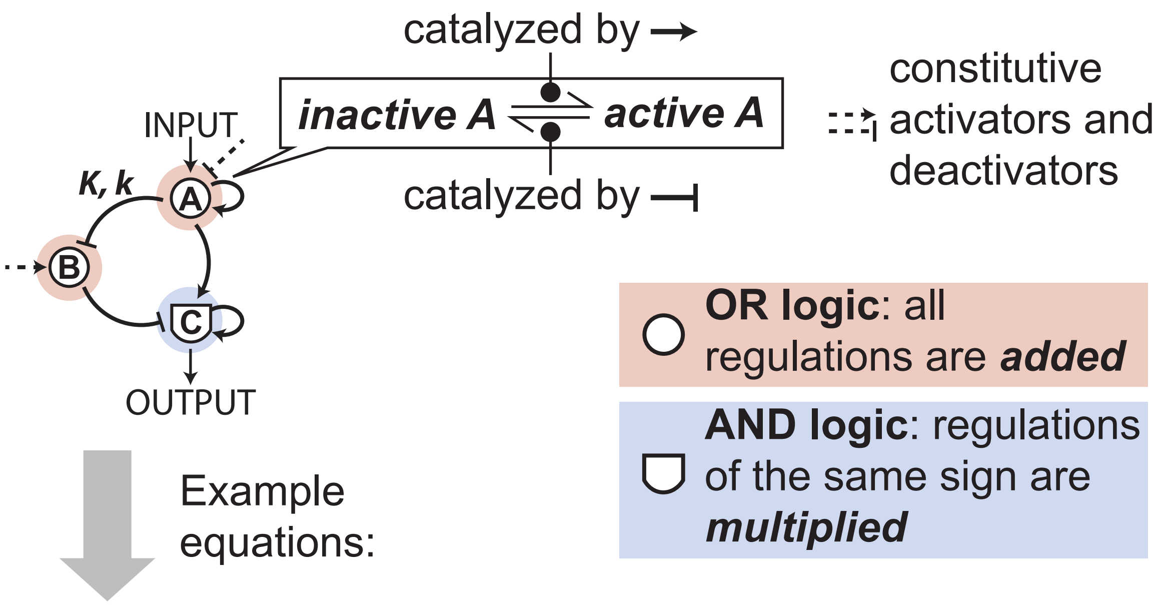


Figure 2.2. Enumerating circuits with 1, 2, or 3 nodes and sampling OR or AND logic on each node.


catalyzed by →
inactive A
active A
catalyzed by ⊣
constitutive activators and deactivators
INPUT
K, k
A
B
C
OUTPUT
Example equations:
○ OR logic: all regulations are added
AND logic: regulations of the same sign are multiplied

$$\frac{dA}{dt} = \frac{k_{IA}\, I\, (1-A)}{K_{IA} + I + (1-A)} + \frac{k_{AA}\, A\, (1-A)}{K_{AA} + A + (1-A)} - \frac{k_{FA}\, F\, A}{K_{FA} + F + A}$$

$$\frac{dB}{dt} = \frac{k_{FB}\, F\, (1-B)}{K_{FB} + F + (1-B)} - \frac{k_{AB}\, A\, B}{K_{AB} + A + B}$$

$$\frac{dC}{dt} = \frac{k_{AC}\, A}{K_{AC} + A + (1-C)} * \frac{k_{CC}\, C}{K_{CC} + C + (1-C)}\,(1-C) - \frac{k_{BC}\, B\, C}{K_{BC} + B + C}$$

X : concentration of active node X
F : constitutive enzyme concentration
k_{XY} : k_{cat} of enzyme X acting on substrate Y
K_{XY} : K_m of enzyme X acting on substrate Y
all nodes have total concentration = 1

Figure 2.3. Modeling signaling circuits. See Chapter 2.3: Modeling enzymatic circuits for details.

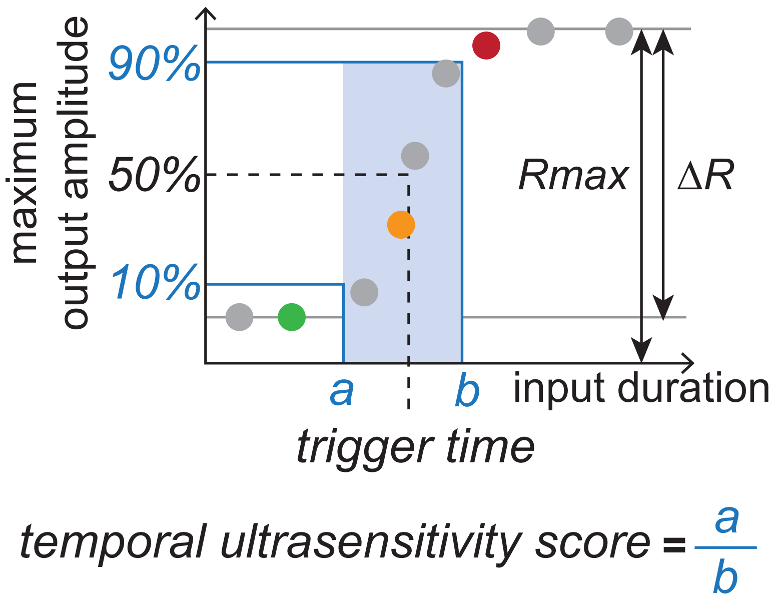


Figure 2.4. Defining temporal ultrasensitivity score, trigger time, Rmax, and ΔR with a temporal dose response curve.

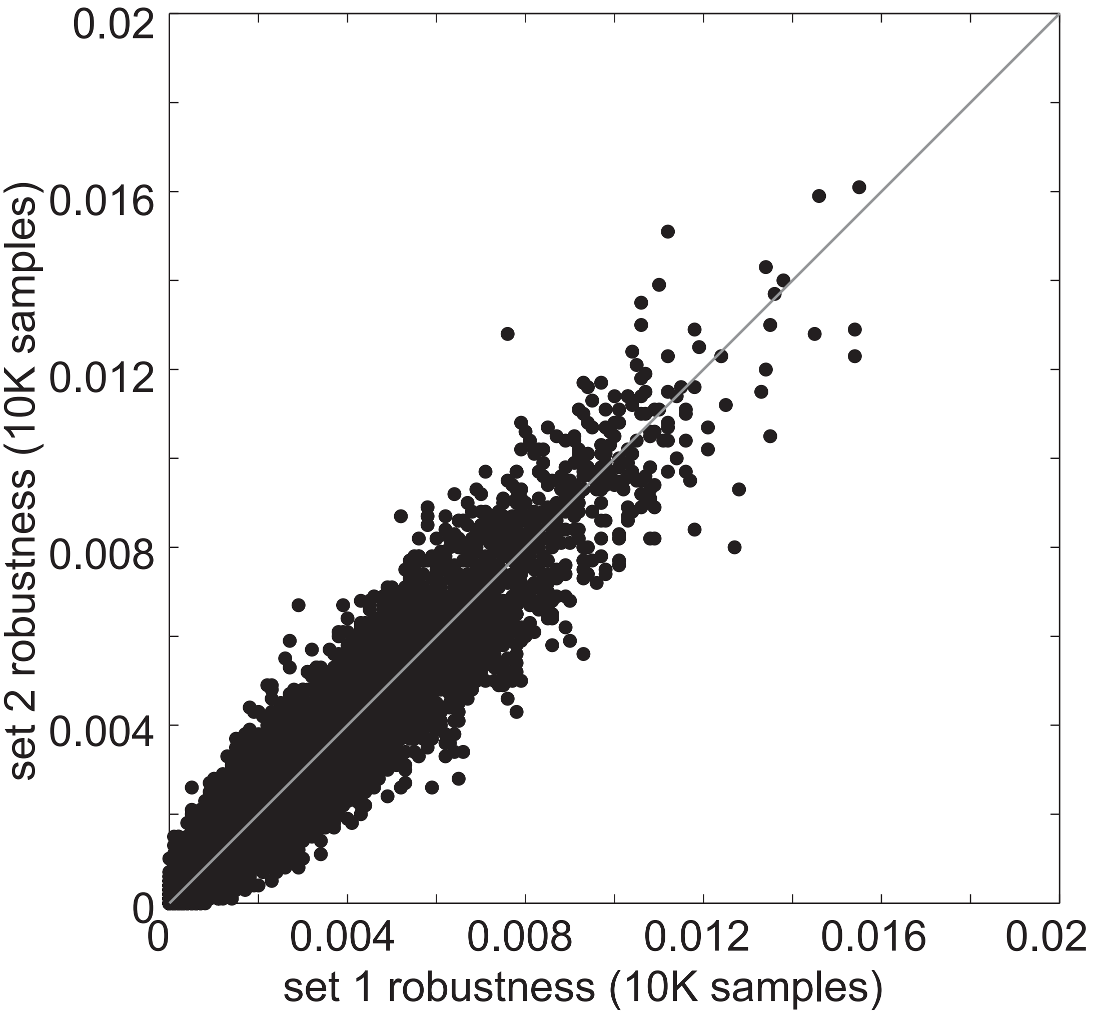


Figure 2.5 Robustness of independent parameter samplings correlate well.

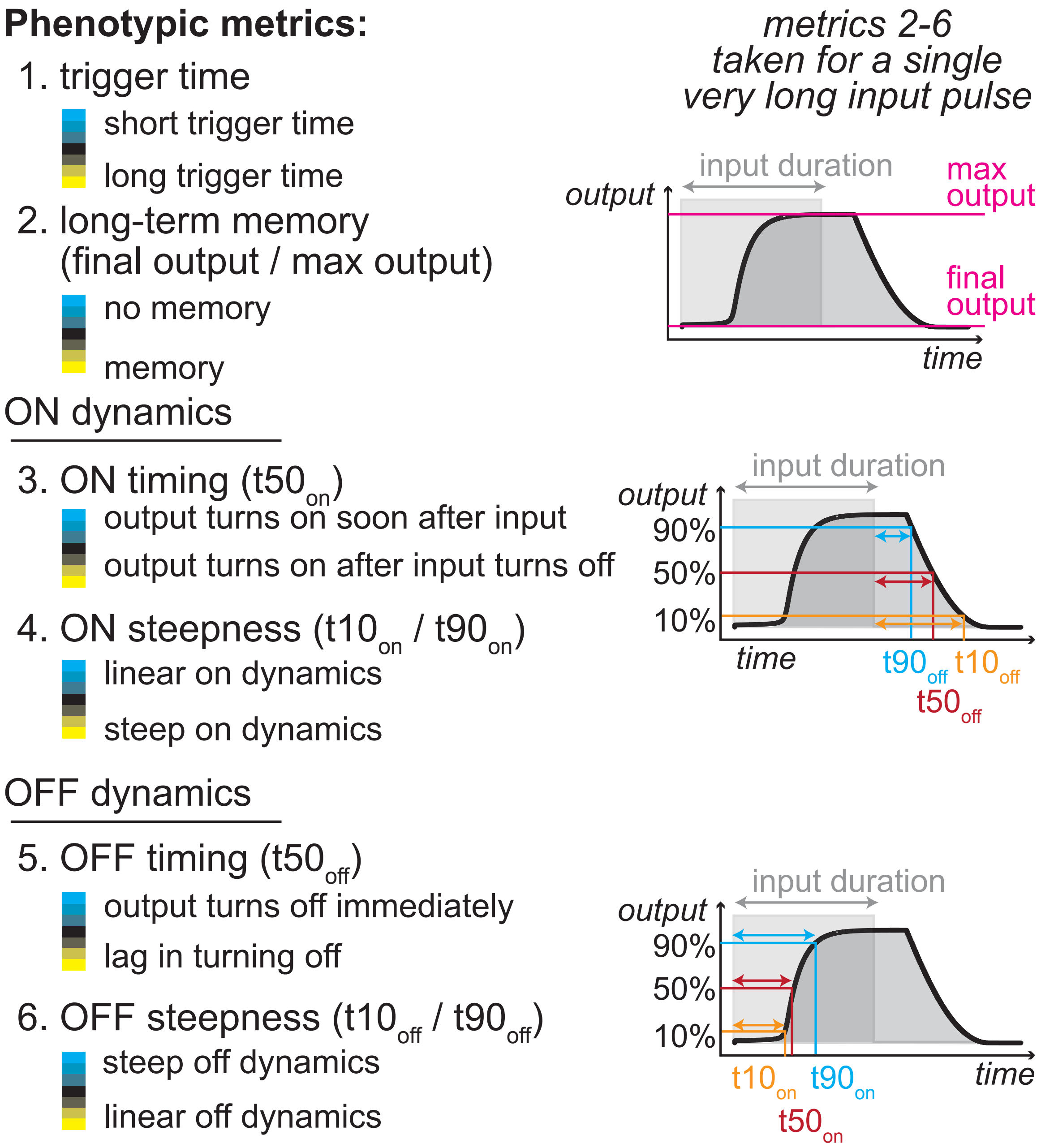


Figure 2.6. Metrics used for phenotypic clustering. log10 is taken on metrics 1, 3, and 5 before clustering. Colors correspond to scale in Figure 3.11.

Chapter 3

ENUMERATION RESULTS AND CLASSIFICATION

OVERVIEW

To identify the simplest kinetic filtering circuits, we enumerated the space of 1-, 2-, and 3-node enzymatic circuits and quantified each topology's performance by measuring its temporal ultrasensitivity score and trigger time (see Chapter 2 for details). We found thousands of 1-, 2-, and 3-node topologies to be robust kinetic filters. In order to focus only on the architectural features that drive kinetic filtering behavior, we identified the minimal topologies capable of kinetic filtering. We subsequently used phenotypic clustering and principal component analysis to sort the minimal topologies into five classes of kinetic filters.

3.1 ONE-NODE TOPOLOGIES

Four topologies comprise the space of 1-node circuits (Figure 3.1). For 1-node networks, there is one possible regulatory link: feedback from A to itself. Under OR logic, where all regulatory effects are summed, this link can be positive, negative, or absent, yielding a total of 3 topologies. Under AND logic, where regulatory effects of the same sign are multiplied, the link can only be positive, because only the positive feedback link results in two regulatory effects of the same sign, the other being the input. For all but the negative feedback case, a constitutive deactivator is added so that node A has both activators and deactivators. Enumerations for 2- and 3-node circuits follow the same pattern.

For each of the four circuit topologies, we sampled 10,000 sets of parameters describing k_{cat} and K_m of every regulation; for each parameter set, we measured the circuit's temporal ultrasensitivity score (TU score) and trigger time, the duration of input that achieves 50% response amplitude (Figure 2.1, and see Chapter 2 for details). Kinetic filtering circuits were defined as follows. We required TU score ≥ 0.5 for kinetic filtering in order to consider only the most "all-or-none" kinetic filters. We required trigger time ≥ 1s in order to exclude the circuit with no regulations, which behaves as a kinetic filter only for trigger times < 1s.

We quantified the robustness of each topology's kinetic filtering by measuring the fraction of a topology's 10,000 parameter sets with TU score ≥ 0.5 and trigger time ≥ 1s. A higher robustness implies that the topology's performance is more robust to changes in parameter values. We required a minimum robustness of 0.001, or 10/10,000 parameter sets, for a topology to be counted as a kinetic filter.

Of the three 1-node networks with OR logic and the one 1-node network with AND logic, only the positive feedback topologies were kinetic filters with robustness ≥ 0.001 (Figure 3.1). Neither the negative feedback nor the no feedback circuit had a single parameter set that showed kinetic filtering behavior. The AND-based positive feedback circuit was over an order of magnitude more robust than the OR-based positive feedback circuit. Details of positive feedback circuit phenotypes and mechanisms are discussed below and in Chapter 4.

3.2 TWO-NODE TOPOLOGIES

In two-node circuit space, each node can use OR or AND logic, yielding four groups of circuits: OR on both nodes (54 topologies), AND on the input/output node and OR on the regulator node (45 topologies), OR on the input/output node and AND on the regulator node (12 topologies), and AND on both nodes (10 topologies), a total of 121 topologies. Compared to using OR logic, using AND logic on a node reduces the number of topologies available because we require an AND node to be subjected to at least two regulations of the same sign.

Of the 2-node networks, the following were kinetic filters: 15 circuits using OR on both nodes, 24 circuits with AND on input/output node A and OR on regulator node B, 3 circuits with AND on node B and OR on node A, and 6 circuits with AND on both nodes (Figure 3.2). As in the 1-node case, using AND logic allows some topologies to be kinetic filters with much higher robustness than any topology relying only on OR logic. The most robust 2-node kinetic filters were about twice as robust as the most robust 1-node kinetic filter; increasing complexity allows increased robustness. All robust 2-node kinetic filtering topologies are shown in Figures 3.3 – 3.6.

3.4 THREE-NODE TOPOLOGIES

Three-node circuit space is comprised of a total of 68,580 topologies, of which 15,330 were kinetic filters with robustness ≥ 0.001 (Figure 3.7). Unlike in the 1- and 2-node systems, in the 3-node system some OR-only circuits are capable of robustness comparable to the most robust AND-containing circuits. In terms of maximum observed robustness, there is only a small gain when increasing complexity from 2 nodes to 3 nodes. Like the 1- and 2-node systems, however, AND-containing circuit space contains more topologies that are highly robust kinetic filters (Figure 3.8).

3.4 MINIMAL KINETIC FILTERING TOPOLOGIES

Our enumerative search identified 15,380 circuits of 1, 2, or 3 nodes to be robust kinetic filters. While this is a very large number of circuits, most of the enumerated circuit topologies did not exhibit kinetic filtering behavior (Figure 3.8). We noted above that using AND logic on the input node allows circuits much greater robustness than when using OR logic, and that increased robustness also allows a higher fraction of topologies using AND logic on the input node to be robust kinetic filters than topologies using OR logic.

To focus only on the core network features that drive kinetic filtering, we identified minimal kinetic filtering topologies where removal of any link of combination of links destroys robust kinetic filtering ability. In the 1-node system, both kinetic filtering topologies are also minimal kinetic filters. In the 2-node system, four circuits were minimal kinetic filters, but two were identical to 1-node positive feedback circuits with node B taking the role of the constitutive deactivator, so they have been ignored in subsequent analysis (Figure 3.9). Adding a second node of complexity thus gains two new kinetic filtering architectures.

In the 3-node system, the 15,330 robust kinetic filters collapse into 21 minimal kinetic filtering topologies, vastly expanding the architectural space available for kinetic filtering (Figure 3.10). Thus in 1-, 2-, and 3-node circuit space, a total of 25 distinct minimal topologies form the basis set of all robust kinetic filtering circuits. All the remaining non-minimal kinetic filtering topologies contain at least one of the minimal kinetic filtering topologies as a substructure.

3.5 CLASSIFYING KINETIC FILTERING TOPOLOGIES

3.5.1 Phenotypic clustering

To further classify the 25 minimal topologies, we clustered them by phenotype. We measured 6 phenotypic metrics for each parameter set of the minimal topologies that resulted in a kinetic filtering circuit, a total of 2896 parameter sets. The phenotypic metrics were as follows: 1) trigger time; 2) long-term memory, defined as final output concentration divided by maximum output concentration; 3) timing of the circuit turning on, defined as how long the output took to reach 50% maximum amplitude; 4) steepness of the circuit turning on, defined as how long the output took to reach 10% maximum amplitude divided by how long the output took to reach 90% amplitude; 5) timing of the circuit turning off, defined as how long the output took to decrease to 50% maximum amplitude after the input was turned off; and 6) steepness of the circuit turning off, defined as how long the output took to decrease to 10% maximum amplitude after the input was turned off divided by how long the output took to decrease to 90% of maximum amplitude after the input was turned off. See Chapter 2 and Figure 2.6 for more details.

Results of clustering the 2896 individual circuits, making up 25 minimal kinetic filtering topologies, by phenotype are shown in Figure 3.11. Hierarchical clustering was performed using Cluster (Cluster 3.0 for Mac OS X using the C Clustering Library version 1.36) with euclidean distances and centroid linkage.

Underneath the clustered phenotypic metrics we have marked the location in the clustering of each parameter set of the 25 minimal topologies. Topologies are colored according to classification determined in subsequent analysis.

The clustergram in Figure 3.11 clearly shows that certain topologies have very similar phenotypes. Topologies 5-8, for example, exhibit a phenotype not accessible by any other topology, as do topologies 1 and 9-14. Topologies 15-17 form a closely related trio; so do topologies 18-20, and topologies 21-24 form a quartet that also overlaps with topology 4. Topology 25 shows some but not all the phenotypes available to topologies 21-24. Additionally, topologies 2, 3, and 15-20 can all access a phenotypic space on the far right of the clustergram that is not populated by the other topologies. Based solely on qualitative observations, we can use the clustergram to classify the 25 minimal kinetic filters into 5 groups: 1) topologies 1 and 9-14; 2) topologies 2, 3, and 15-20; 3) topologies 4 and 21-24; 4) topologies 5-8; and 5) topology 25.

3.5.2 Principal components of phenotypic space

To classify the 25 minimal kinetic filters more quantitatively, we conducted a principal component analysis using the 6 phenotypic metrics. Composition and singular values of the 6 principal components are shown in Table 3.1. Squaring the singular values gives the proportional variance explained by each principal component.

	PC#1	PC#2	PC#3	PC#4	PC#5	PC#6
Singular value	29	26	22	16	15	7
Trigger time	0.169	-0.564	0.298	-0.433	-0.276	0.549
Long-term memory	-0.938	-0.165	-0.082	0.117	0.025	0.268
ON timing	-0.155	-0.546	0.340	-0.106	0.088	-0.737
ON steepness	0.242	-0.437	0.031	0.703	0.445	0.239
OFF timing	-0.017	0.056	-0.067	-0.525	0.840	0.107
OFF steepness	0.093	-0.403	-0.885	-0.133	-0.109	-0.122

Table 3.1. Principal components of phenotypic space. Row 1 shows the singular value of each of the 6 components, while remaining rows contain relative weights of each of the phenotypic metrics that compose principal components.

Principal component 1 is composed nearly entirely of the long-term memory phenotypic metric. Principal component 2 is mostly trigger time and on timing, but also includes significant contributions from on steepness and off steepness. Principal component 3 has its largest contribution from off steepness. Principal components 4, 5, and 6 have smaller singular values and explain less of the variance in the phenotypic metrics. Thus the 25 minimal kinetic filters phenotypically differ from each other primarily in terms of long-term memory, trigger time and on timing, and off steepness.

To group the minimal kinetic filtering topologies by phenotype, we found the mean value of each principal component for all the parameter sets in each topology, then plotted the means in 3D principal component space (Figure 3.12). The topologies fall into three separate clusters: a group of 4 (green, topologies 5-8), a group of 7 (red, topologies 1 and 9-14), and the remaining 14 topologies. The large group of 14 can be subdivided into three groups: a singlet (orange, topology 25), a group of 5 (blue, topologies 4 and 21-24) that are higher in principal component 3, and a group of 8 (purple, topologies 2, 3, and 15-20) that is lower in principal component 3. These five groups correspond exactly to the qualitative groupings seen in the clustergram in Figure 3.11.

3.5.3 Five classes of kinetic filters

The five classes of minimal kinetic filters are shown in Figure 3.13. Although these groups were identified by clustering similar phenotypes, we find that topologies within each group also share key structural features. All topologies in the red group contain a positive feedback loop that activates a node using OR logic. The positive feedback loop may be self-self (topologies 1, 10, 13, 14) or be encoded through one (topologies 11, 12) or two (topology 9) regulator nodes. Regardless of the size and location of the positive feedback loop, topologies in this group are all bistable and capable of long-term memory (Figure 3.12).

Topologies in the purple group also contain positive feedback loops, but in contrast to the red group, positive feedback now feeds into a node using AND logic. Positive feedback loops can encompass one (topologies 1, 15, 18, 20), two (topologies 3, 18), or three (topology 17) nodes but must feed back onto an AND node. Unlike the positive feedback OR circuits in the red group, circuits with positive feedback AND do not exhibit long-term memory (Figure 3.12).

In the blue group, all topologies contain a double inhibition motif, where a node that acts as an inhibitor is itself inhibited. The double inhibition is presented as a net positive feedback (topologies 4, 21, 22, 23) or in series (topology 24), but phenotypes of kinetic filters are similar in the feedback and series cases (see topologies 21-24 in Figure 3.11 as well as Figures 4.6-4.9). Although the double inhibition in feedback form is often categorized as a positive feedback loop with exclusive bistability (A on and B off or vice versa), these minimal double inhibition circuits are not bistable without additional positive feedback loops. Like the positive feedback AND circuits, double inhibition circuits are incapable of long- term memory, but show much higher off steepness than positive feedback AND circuits (Figure 3.12). Unlike positive feedback circuits, double inhibition circuits cannot be implemented using only one node.

Topologies in the last two groups require at least 3 nodes to implement their shared structural features. In the green group, whose phenotype consists of long-term memory and very long on timing (Figure 3.12), shared structural features include a positive feedback loop through one node, a negative feedback loop through at least two nodes, and an inverter on the output node. The mechanism for kinetic filtering in this unusual class is explained in more detail in the next chapter.

Finally, the last group of kinetic filters consists of one circuit, the coherent feed forward loop with AND logic, which is the only previously known kinetic filter. In the coherent feed forward loop, signal from the input reaches the output node through two parallel paths: directly from the node receiving input and from a regulatory node that is also activated by the node receiving input. Circuits using the coherent feed forward loop to drive kinetic filtering show no long-term memory, short trigger time, and high off steepness (Figure 3.12).

More details of the principal component analysis are shown in Figures 3.14 and 3.15. In Figure 3.14, each data point represents one kinetic filtering parameter set of a minimal kinetic filtering topology colored by the classification discussed above. In Figure 3.15, we show the distribution of principal component values as stacked histograms, where each element in the stack is a class of kinetic filter. The projections and histograms clearly show which principal components separate which classes of kinetic filters.

Principal component 1, which corresponds to the long-term memory phenotypic metric, separates the positive feedback OR and bistable buffering inverter circuits from the other three classes. Principal component 2, a conglomeration of trigger time and on timing, distinguishes positive feedback OR and bistable buffering inverter circuits from each other. Principal component 2 also separates coherent feed forward loop circuits from the positive feedback AND and buffering double inhibition circuits, which are distinguished from each other via principal component 3, which describes the steepness of the circuit turning off.

3.6 COMPARISON OF KINETIC FILTER PHENOTYPES

After enumerating all enzymatic circuits with 1, 2, or 3 nodes and phenotypically clustering the minimal robust kinetic filtering topologies, we identified five behavioral and structural classes of kinetic filters. A comparison of phenotypic features of the five classes of kinetic filters is presented in Figure 3.16 and recapitulates many observations discussed above.

Positive feedback OR and bistable buffering inverters are both bistable topologies that stay on after responding to input. These topologies are "fire-once" systems that cannot respond again to new input. Positive feedback OR and bistable buffering inverters differ in that bistable buffering inverters can only turn on after input has turned off---their mechanism is discussed in more detail in Chapter 4.

The remaining three classes of kinetic filters all turn off after responding to input of any duration. Positive feedback AND circuits, however, turn off more gradually than buffering double inhibition circuits and coherent feed forward circuits. Buffering double inhibition circuits are the only topologies capable of turning off quickly and immediately after a long input has turned off; they are the only kinetic filtering architectures that can faithfully follow the shape of the input. Coherent feed forward loop circuits, and some buffering double inhibition circuits, show a delay-off phenotype where output stays elevated for a short while after input has turned off but then turns off sharply.

We discuss mechanism and more details about kinetic filtering performance of the five classes of small-network kinetic filters in Chapter 4.

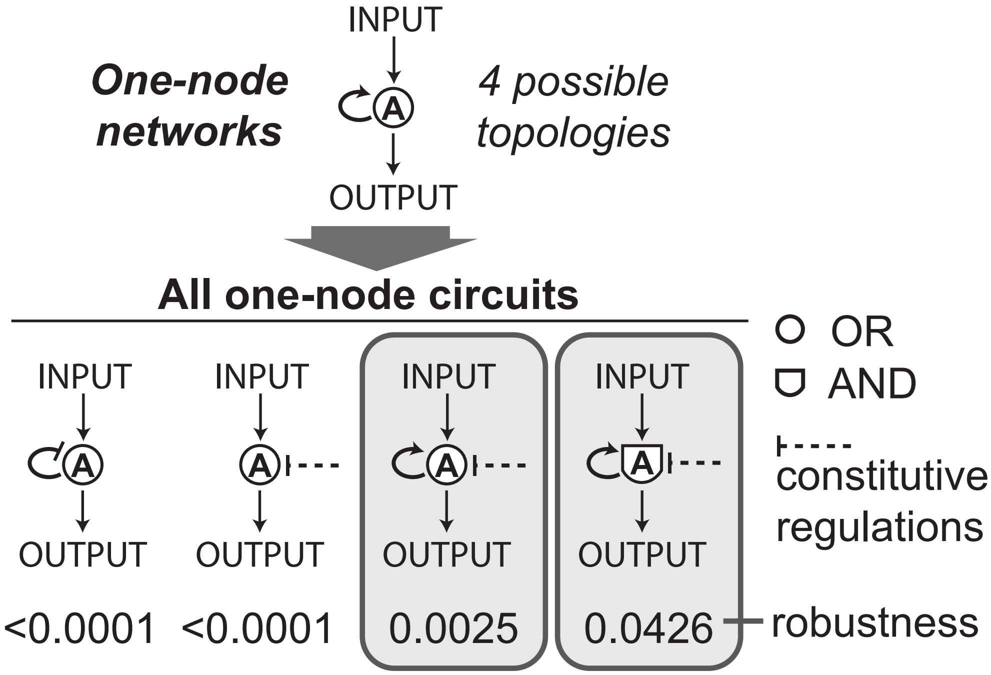


Figure 3.1. Two out of four one-node topologies are robust kinetic filters.

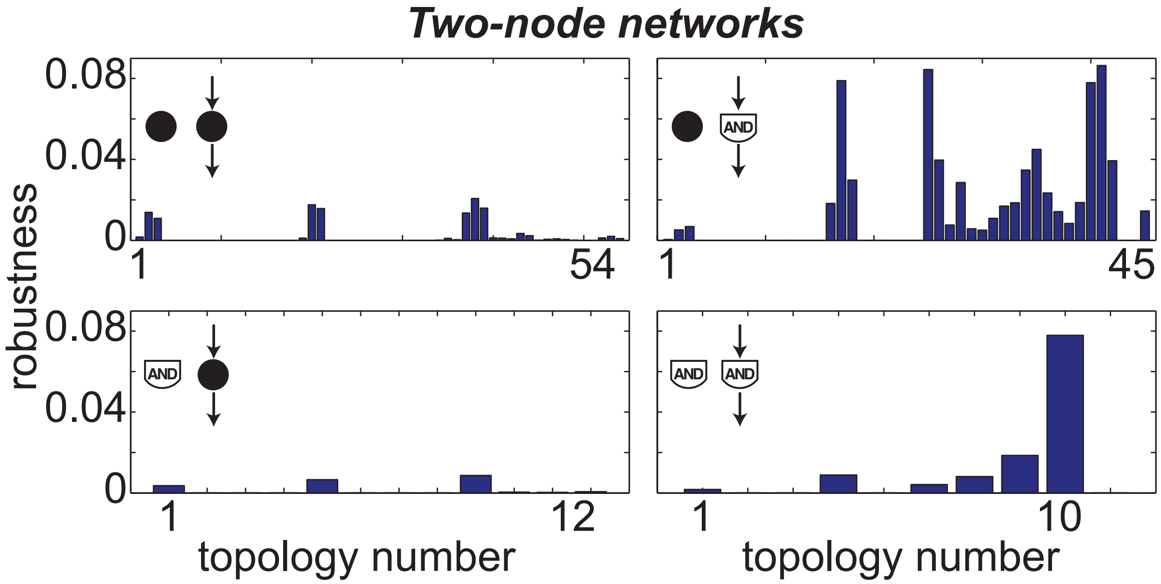


Figure 3.2. Robustness of all two-node topologies.

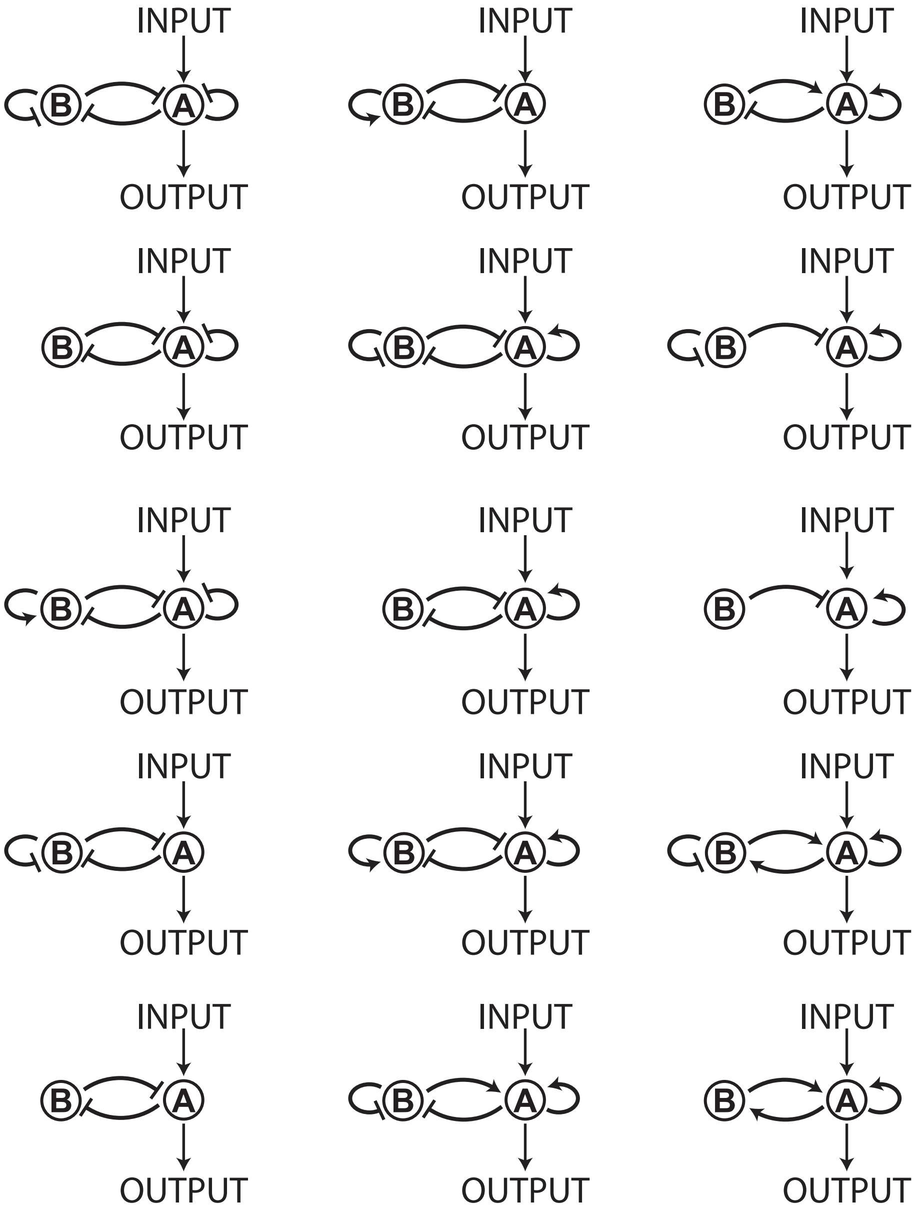


Figure 3.3. 15 2-node OR circuits have robustness ≥ 0.001.

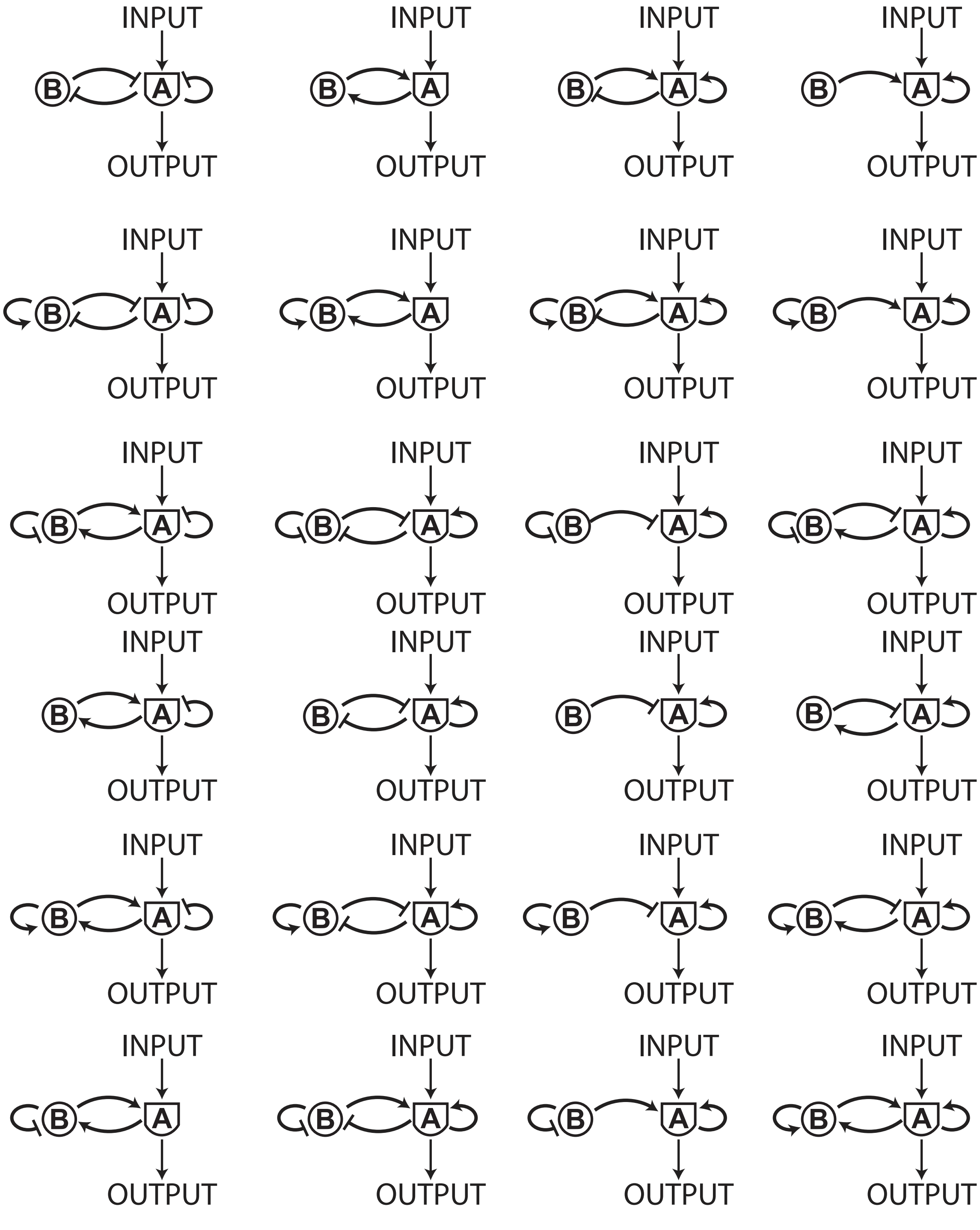


Figure 3.4. 24 2-node circuits with AND logic on the input/output node only have robustness ≥ 0.001.

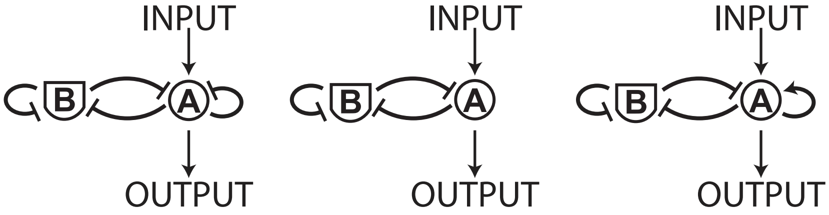


Figure 3.5. 3 2-node circuits with AND logic on the regulator node only have robustness ≥ 0.001.

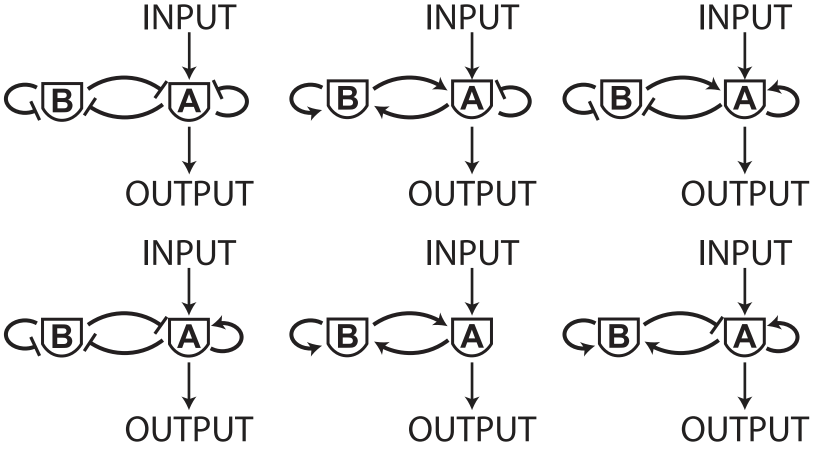


Figure 3.6. 6 2-node circuits with AND logic on both nodes have robustness ≥ 0.001.

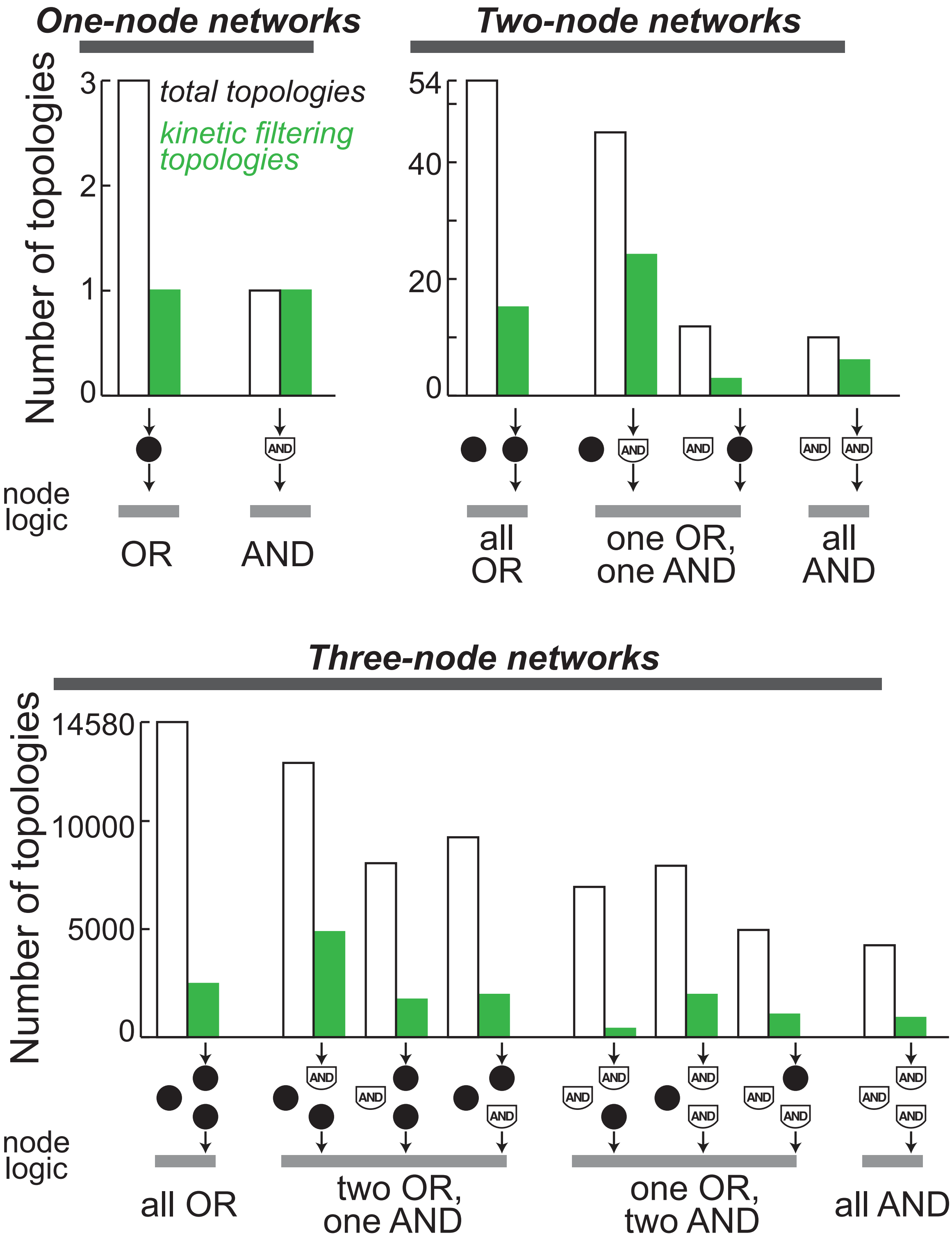


Figure 3.8. Most topologies are not robust kinetic filters.

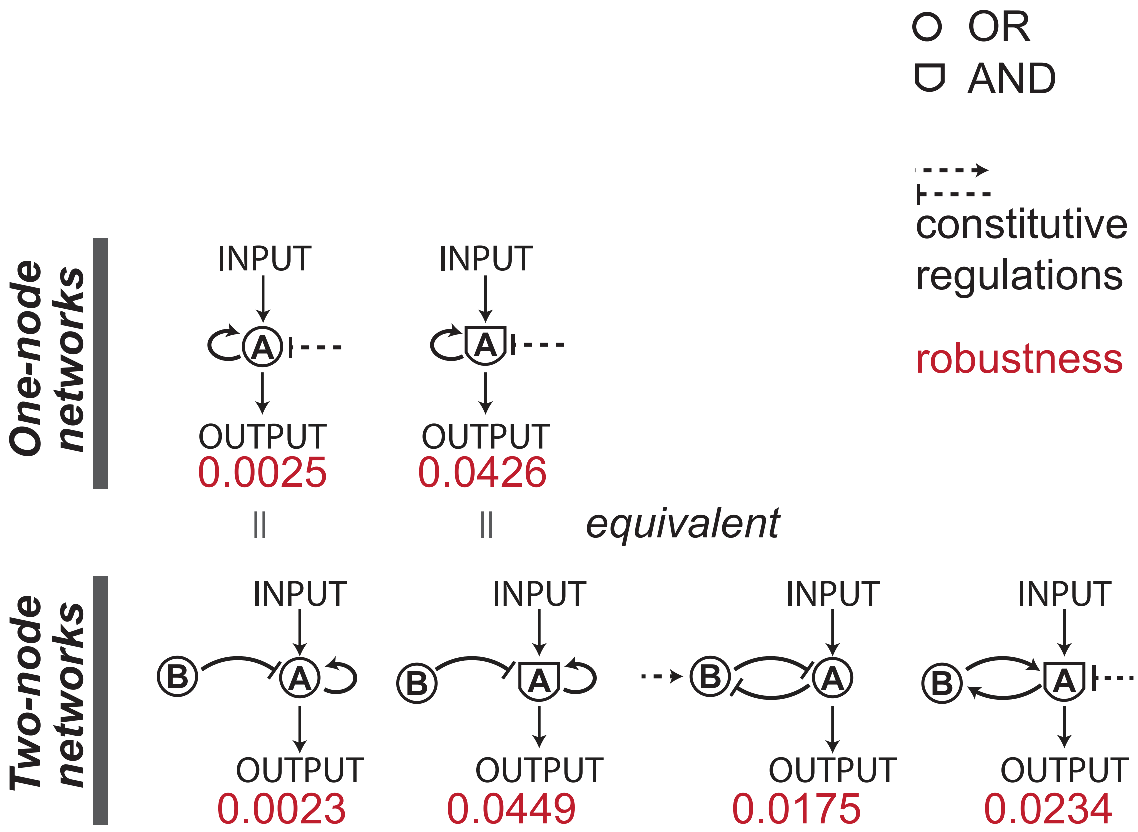


Figure 3.9. Minimal kinetic filters of 1 and 2 nodes.

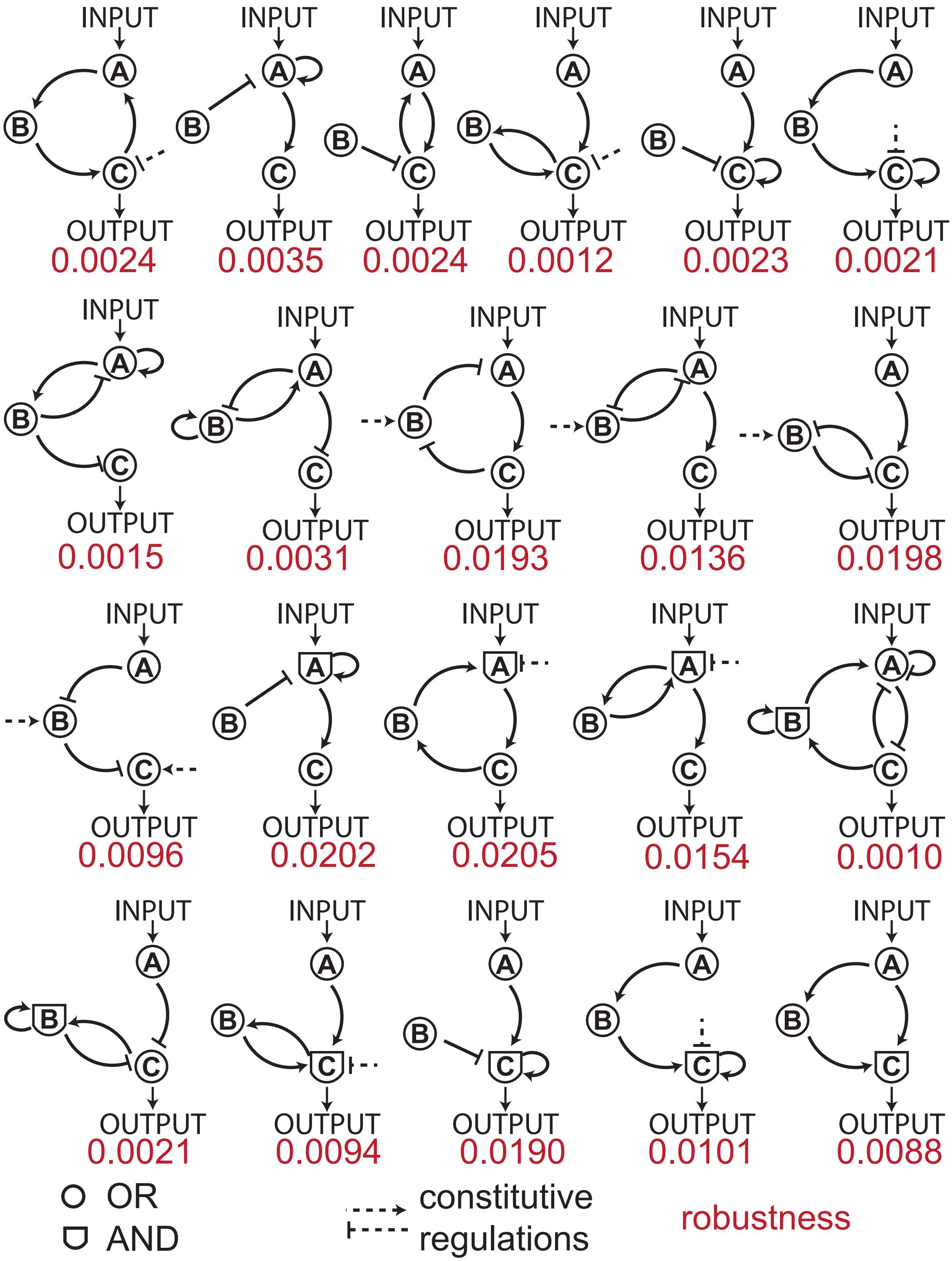


Figure 3.10. Minimal kinetic filters of 3 nodes.

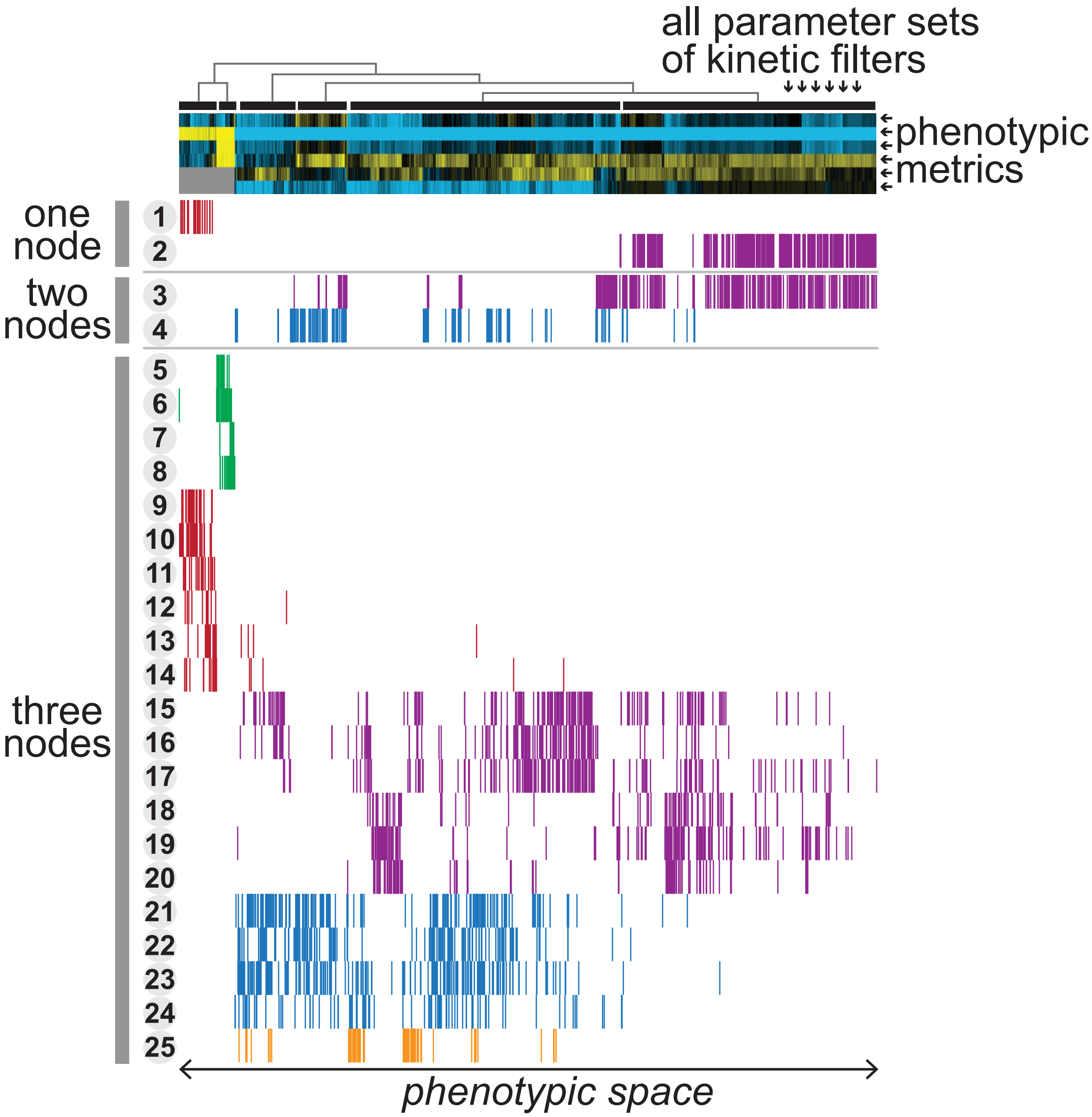


Figure 3.11. Clustergram of kinetic filtering parameter sets of minimal kinetic filtering topologies. See Figure 2.6 for description of phenotypic metrics and blue-yellow scaling.

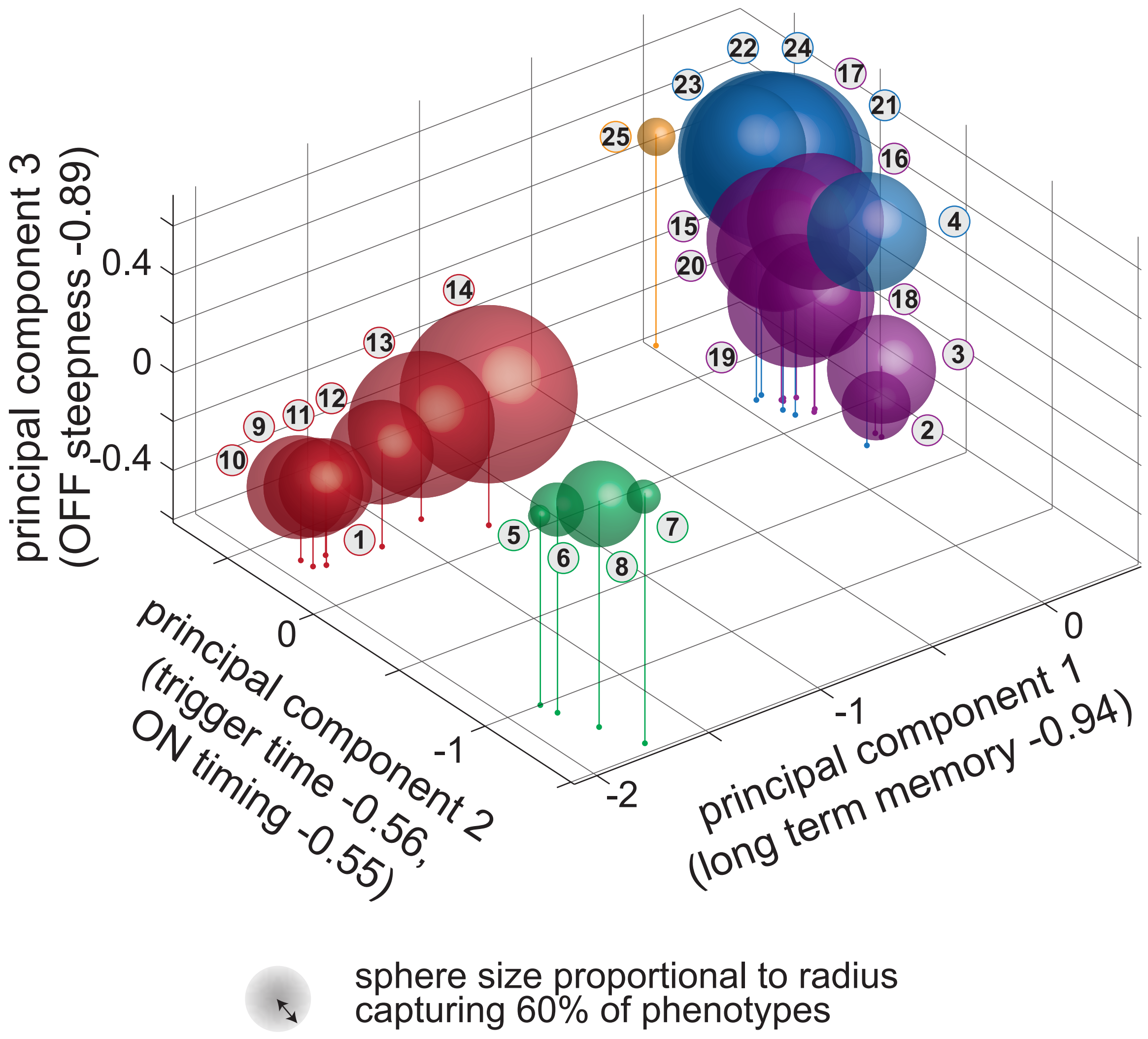


Figure 3.12. Location of 25 minimal kinetic filtering topologies in principal component space.

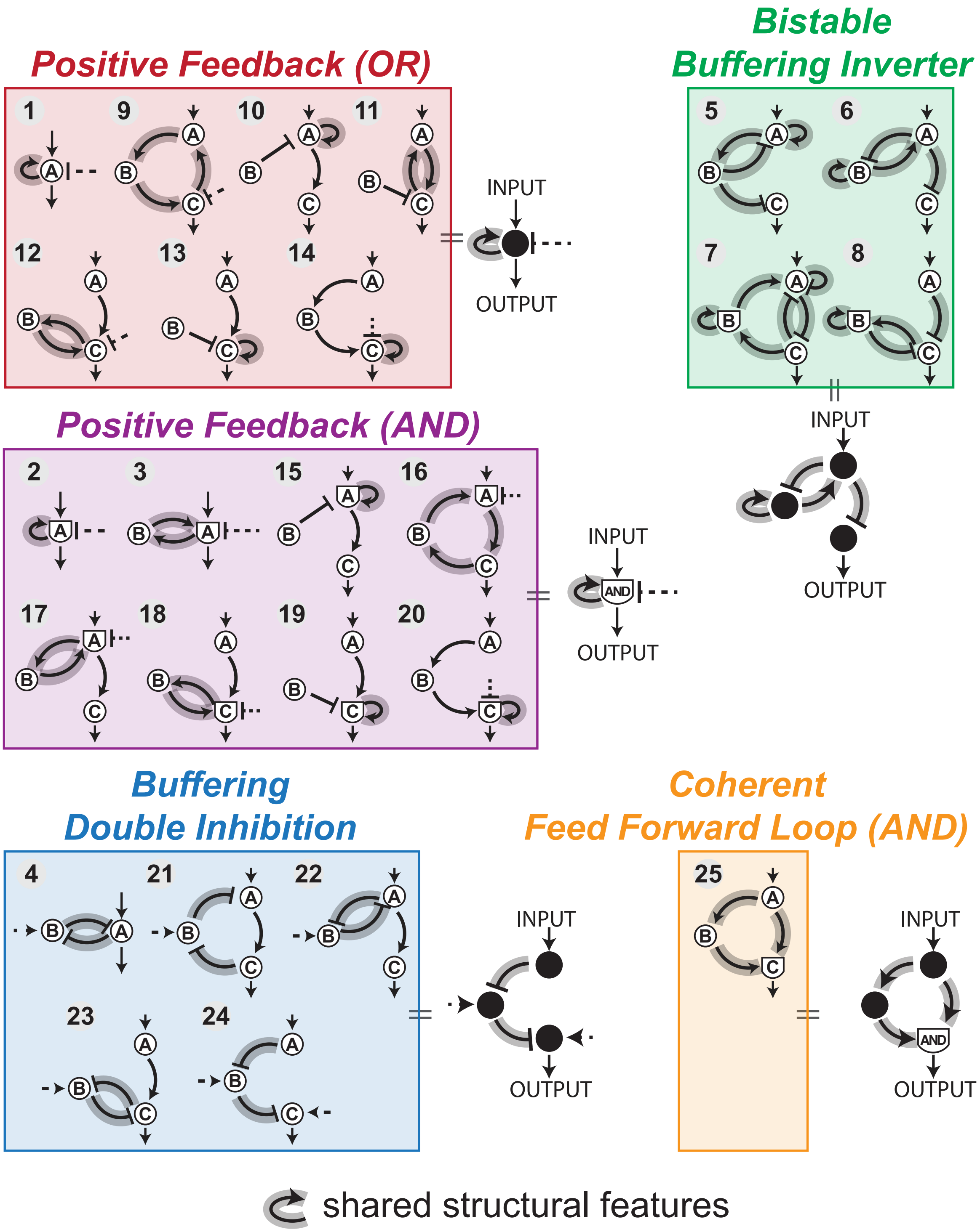


Figure 3.13. 25 minimal kinetic filters are phenotypically classified into 5 groups that also each share structural features.

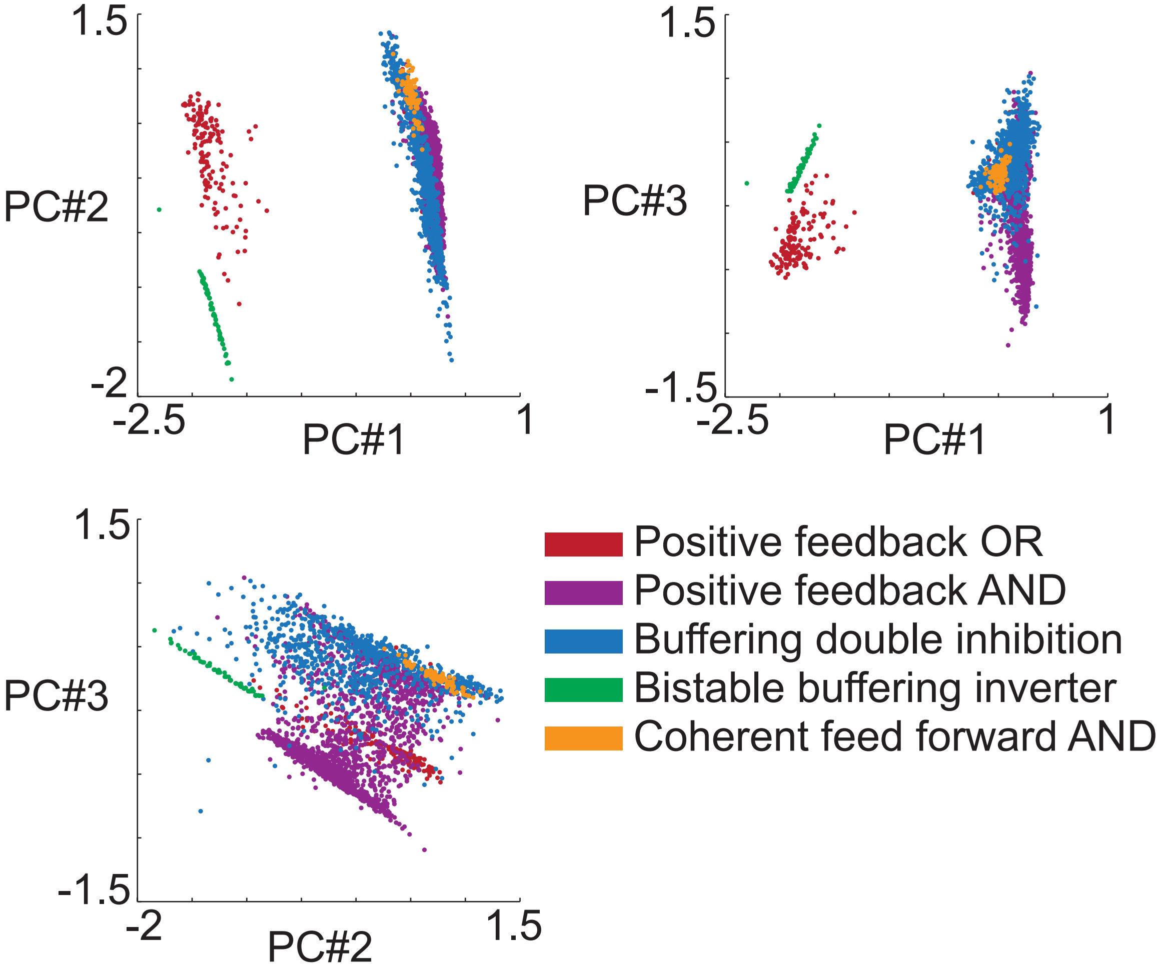


Figure 3.14. Projections in principal component space of individual kinetic filtering parameter sets, colored by classification.

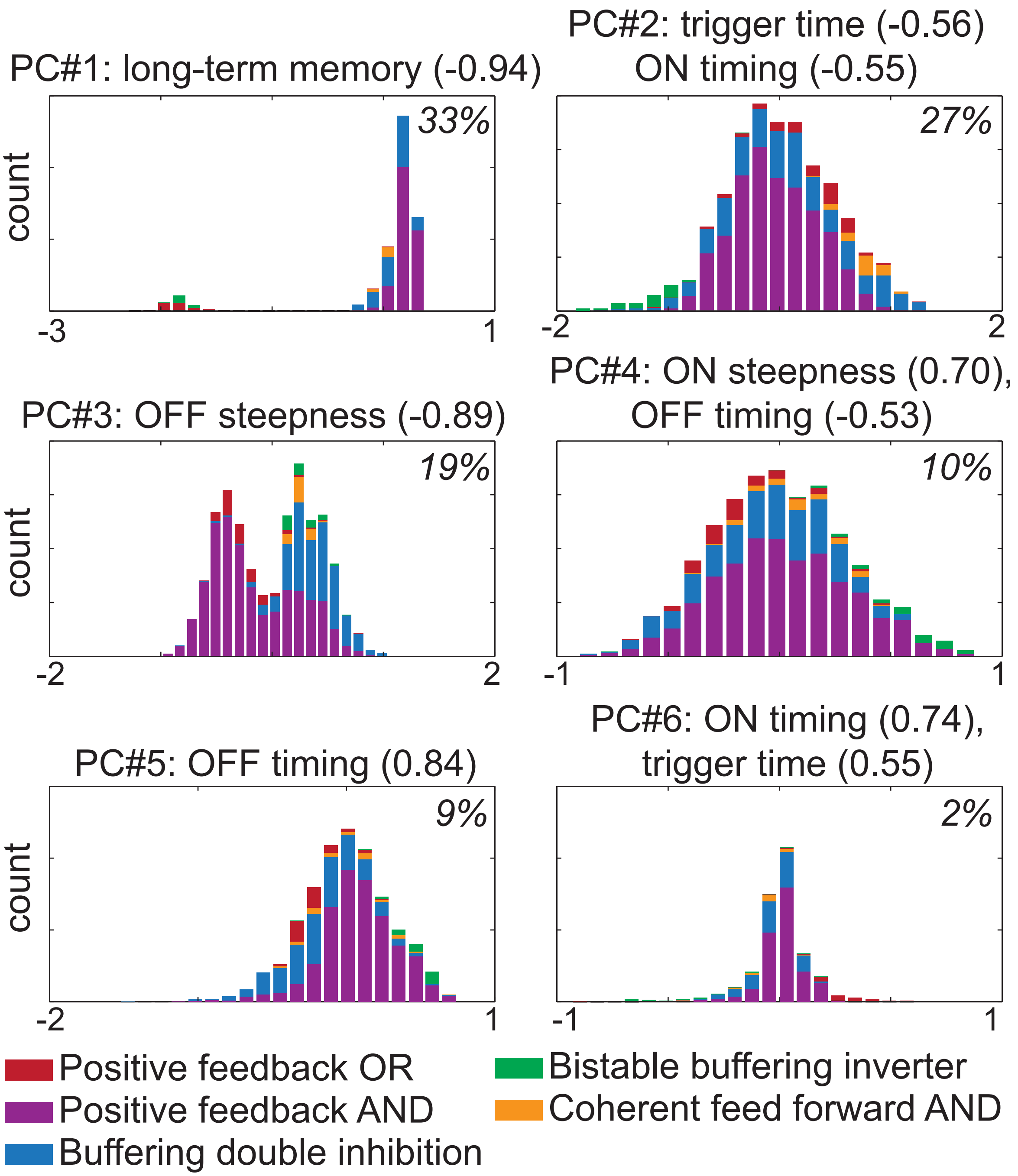


Figure 3.15. Distributions of principal components of individual kinetic filtering parameter sets, colored by classification. Inset number is the proportional variance of each principal component.

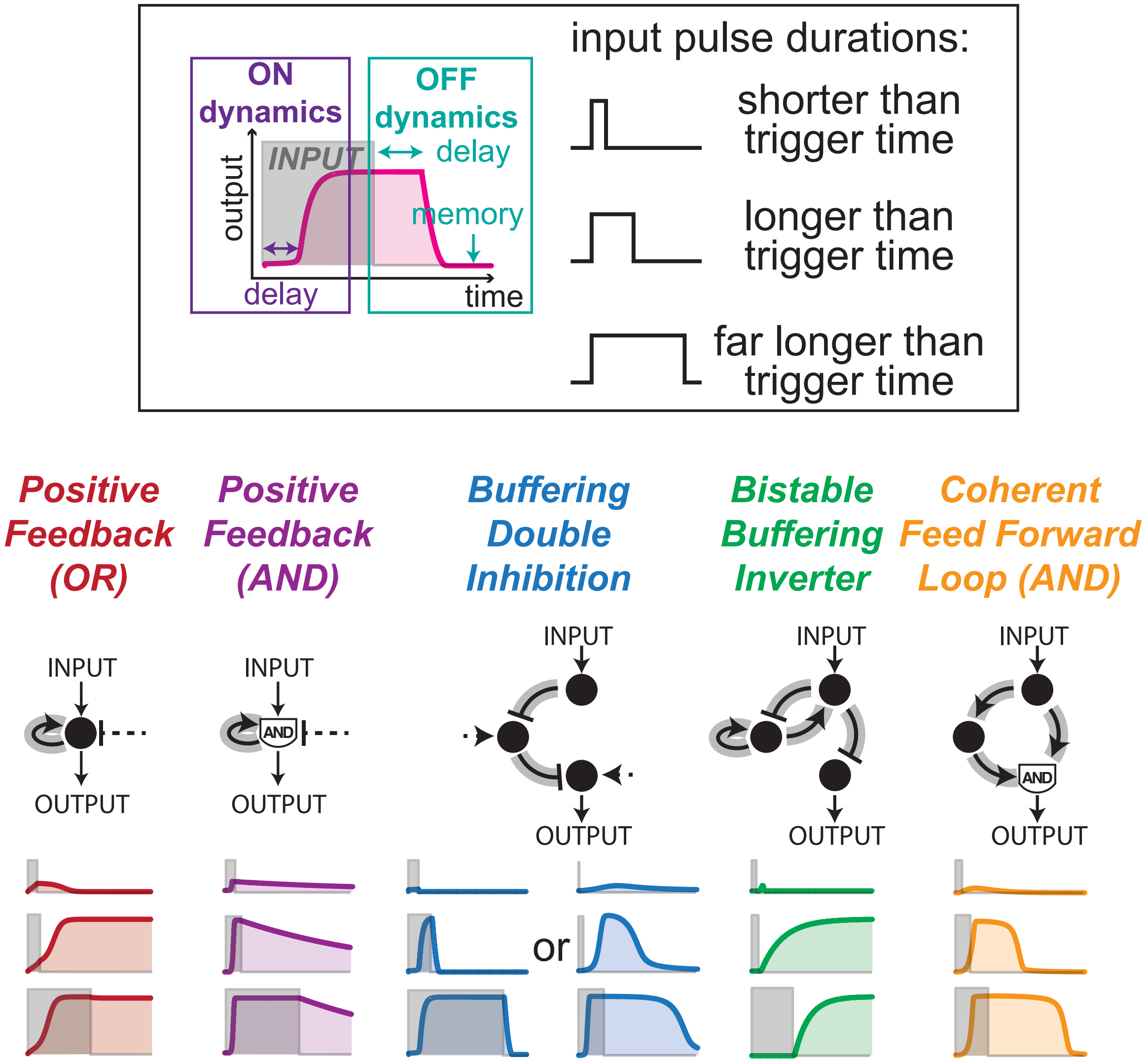


Figure 3.16. Phenotypic features of 5 classes of kinetic filters.

Chapter 4

FEATURES OF SIMPLE KINETIC FILTERS

OVERVIEW

In this chapter we conduct a more detailed investigation into the properties of the five types of kinetic filters of 1, 2, or 3 nodes uncovered in Chapter 3. Taking a detailed look at representative timecourses allows us to elucidate the mechanisms by which the five kinetic filter classes distinguish between transient and sustained stimuli. We show that different classes can access different regimes of trigger time and demonstrate why a simple activation cascade is incapable of the long trigger times we observe in double inhibition topologies. We also show that a kinetic filter's reset time, the length of time necessary for the circuit to forget a short input, depends on architecture type. Preferred parameter regimes are identified for a representative topology in each class of kinetic filter. Lastly, we show that most kinetic filters are integrating input, but some can act as absolute timers insensitive to input height.

4.1 EXAMPLE TIMECOURSES AND MECHANISMS OF FIVE TYPES OF KINETIC FILTERS

4.1.1 Positive feedback OR

Positive feedback OR circuits are bistable switches. Temporal dose response and timecourses of a representative circuit are shown in Figure 4.1, with additional examples in Figure 4.2 (parameters used to plot timecourses and other data are listed in Appendix B). Long inputs raise levels of active A above a threshold concentration, indicated by the dotted lines in the timecourses. When active A surpasses the threshold, positive feedback strength exceeds deactivation strength, activating A and bringing levels of active A up to a high steady state. Short inputs leave concentration of active A below the threshold, where activity of the constitutive deactivator dominates, and node A returns to its initial deactivated state.

Once a positive feedback OR circuit has turned on, removal of input does not result in output returning to the off state, as the positive feedback is strong enough to keep the circuit in a stable on state. Positive feedback OR circuits thus exhibit long-term memory and irreversibility; resetting the circuit requires degradation and new synthesis of node A. See section 4.4 for a detailed examination on parameter requirements for bistability in positive feedback OR circuits.

4.1.2 Positive feedback AND

Unlike positive feedback OR circuits, positive feedback AND circuits are not bistable and instead slowly turn off after input has been removed (Figure 4.3, with additional examples in Figure 4.4). Because of the AND gate at node A, both input and positive feedback must be present to activate A, and thus A cannot remain activated after input has turned off.

Given that both input and positive feedback must be present to activate A, how do positive feedback AND circuits ever turn on? Positive feedback AND circuits that are kinetic filters have very weak constitutive deactivators (see section 4.4 on preferred parameter regimes), allowing input to activate a small amount of A on its own. The circuit turns on if and only if the input was on long enough to activate enough A to trigger positive feedback. The weakness of the constitutive deactivator also explains why output of an activated circuit turns off much more slowly than it turns on, leading to the slow and flat off dynamics common in positive feedback AND circuits. In the 2-node implementation of the positive feedback AND circuit, both output and regulator concentrations decay very slowly to pre-input levels (Figure 4.5).

4.1.3 Buffering double inhibition

Representative timecourses of double inhibition circuits are presented in Figures 4.6-4.9. When input turns on, an increase in active A levels leads to a linear decrease in active B levels. For long inputs, active B concentration drops to the point where active node C can quickly accumulate. For short inputs, node B buffers the input signal and concentration of active B never falls enough for active node C to rise very much. When the double inhibition links are arranged in feedback (Figures 4.8 and 4.9), node A acts as both the input and the output node. Node A is initially activated only a small amount by the input, and this small amount of active A is enough to begin turning off B; when levels of active B have fallen low enough, A can then fully activate.

Like positive feedback AND circuits, buffering double inhibition circuits show a distinct delay between input application and rise in output. As we noted in the phenotypic classification discussed in Chapter 3, buffering double inhibition circuits have identical phenotype when the double inhibition links are arranged in series and in feedback (compare timecourses Figures 4.6 and 4.8).

Where positive feedback AND circuits delay while slowly accumulating enough active A to overcome constitutive deactivation, buffering double inhibition circuits delay while node B buffers the input signal. Unlike positive feedback AND circuits, buffering double inhibition circuits can quickly turn off once input has been removed, but can also show a lag in turning off for some parameter combinations (Figure 4.7 bottom).

4.1.4 Bistable buffering inverter

In the bistable buffering inverter group of kinetic filters, a representative timecourse shows that short inputs lead to oscillations between A and B that eventually die out, and C is not activated (Figures 4.10 and 4.11). Long input gives A enough time to completely deactivate B; because only active B can catalyze the activation of B, B's deactivation is permanent. When the input turns off, only deactivators of A remain, allowing A to be deactivated and C to be activated by its constitutive activator. The elements necessary to the bistable buffering inverter are threefold: the regulator (node B) is turned off by input; the regulator can only be activated via self positive feedback, with the consequence that once completely deactivated, the regulator cannot be reactivated; and an inverter turns the depletion of regulator into activation of the output node. These elements can be arranged in multiple ways, as we saw in Chapter 3, without changing the output phenotype (Figure 4.12).

Unlike any other class of kinetic filters, bistable buffering circuits only turn on after input has been turned off as long as the input was long enough to activate the circuit. Longer inputs further delay output onset. Like positive feedback OR circuits, bistable buffering inverter circuits show long-term memory.

4.1.5 Coherent feed forward loop AND

A representative coherent feed forward loop timecourse shows that output node C is only activated when both active A and active B rise above a threshold concentration, and C remains active until active A and active B drop back below that threshold (Figures 4.13 and 4.14). Coherent feed forward circuits can remain activated long after input has been turned off, but will eventually return to a deactivated steady state once active A or active B falls below its threshold.

The coherent feed forward loop with AND logic has been previously described as a delay-on circuit. In our hands, where we have used enzymatic instead of transcriptional regulation, no Hill coefficients, and application of input to one node instead of two, we observe both a delayed on and a delayed off. The delayed off is due to the decay time of nodes A and B.

4.2 TRIGGER TIME

4.2.1 Distribution of trigger times by kinetic filter type

Kinetic filters of the five architecture types we found can have trigger times spanning up to 5 orders of magnitude in time, but trigger time varies between architecture types. Coherent feed forward loop kinetic filters are limited to short trigger times, unlike circuits in the other four classes. Positive feedback OR and bistable buffering inverter circuits also favor short trigger times but can access longer trigger times for some parameter combinations. Positive feedback AND kinetic filters tend to have longer trigger times than coherent feed forward, positive feedback OR, and bistable buffering inverter kinetic filters, but not as long as buffering double inhibition kinetic filters. The longest trigger times and the widest range of trigger times are accessible only to the buffering double inhibition architecture.

Of the five classes of kinetic filters, the coherent feed forward loop and buffering double inhibition architectures are the only classes where kinetic filters are reversible and modular. These architectures have no long-term memory and can quickly return to the off state after input has turned off; in a biological context, further circuitry can be added downstream of coherent feed forward loop or buffering double inhibition architectures to make cell fate decisions based on the activity of the kinetic filter. For these reasons, we will focus on the coherent feed forward loop and buffering double inhibition architectures in the next few sections.

4.2.2 Coherent feed forward loop trigger time is governed by longer trigger time of its two arms

We asked why buffering double inhibition circuits can have much longer trigger time than coherent feed forward loop circuits, which are limited to relatively short trigger times. We resampled the coherent feed forward loop more finely, at 100,000 parameter sets, to gather better statistics. Taking only the kinetic filtering parameter sets, we knocked out each arm of the coherent feed forward loop in turn and measured the resulting circuit's trigger time (Figure 4.16). Most of the knockout circuits did not meet the TU score (steepness) requirement for kinetic filtering, and of those that did, none had trigger time above 1s.

Correlations of feed forward loop trigger time to trigger time of each of the knockouts are shown in Figure 4.17 along with correlation of feed forward loop trigger time to the shorter of the knockout trigger times (Figure 4.17) and longer of the knockout trigger times (Figure 4.16). Trigger times of the two knockouts do not correlate with each other. Slopes and correlation values are bootstrapped values.

We find that coherent feed forward loop trigger time correlates best with the maximum trigger time of the two knockout circuits. In our set of coherent feed forward loop kinetic filters, 45% had longer trigger times in the right arm knockout (3-node cascade) and 55% had longer trigger times in the left arm knockout (2-node cascade) (Figure 4.18). Timecourses of coherent feed forward loop circuits show that circuits where the right arm knockout has longer trigger time also have concentrations of node A that rise quickly to high levels after input turns on. Similarly, when the left arm knockout has longer trigger time, node B rises more quickly and to higher levels than node A.

While one might expect that the left arm of the coherent feed forward loop is the slower arm because it contains an additional node and additional step compared to the right arm, we find that the left arm is slower only in 45% of cases. Over half of our kinetic filtering parameter sets had slow dynamics in turning on node A and faster and more sensitive response at node B, thus making the shorter arm of the coherent feed forward loop also the slower arm.

4.2.3 Double activation circuits show faster dynamics than double inhibition circuits even under identical parameters

After showing that coherent feed forward loop trigger time is determined by the trigger time of its slower arm, we directly compared response dynamics of a double inhibition circuit and a double activation circuit, which is a coherent feed forward loop with the right arm knocked out (Figure 4.19). We used a kinetic filtering parameter set of the double inhibition topology and also simulated the double activation circuit using those parameters. We gave each circuit a single step input and followed timecourses of output node C and regulator node B.

Dynamics of node B were the same in both cases, differing only in sign. In the double inhibition circuit, node B begins in the fully activated state and is deactivated upon input; in the double activation circuit node B begins in the fully deactivated state and is activated upon input. As expected, the shape of deactivation and activation are identical due to their underlying identical parameters.

Surprisingly, output node C dynamics are not identical. In the double inhibition circuit, output activation exhibits a long delay; in the double activation circuit, output is activated almost immediately (Figure 4.19). Tuning up the strength of the constitutive deactivator can delay output activation in the double activator circuit, but only by a small amount, and at the cost of flattening the rate of rising active output levels (Figure 4.19, pink dotted lines). In the double inhibition circuit, tuning up the strength of the constitutive activator has the opposite effect: output activates sooner but also more gradually.

The difference in output dynamics between double inhibition and double activation circuits drives the difference in temporal dose response and hence kinetic filtering behavior. Because output rises sooner after input application in double activation circuits than in double inhibition circuits, temporal dose response curves are shifted to the left for double activation circuits (Figure 4.19 bottom). We also see the tradeoffs in steepness and delay noted above for double activation circuits: increasing constitutive deactivator strength prolongs trigger time but decreases temporal dose response steepness. For double inhibition circuits, longer trigger time and steeper temporal dose response go hand in hand.

4.2.4 Differences in dynamics are driven by steady state behavior

To investigate the origin of asymmetry in the output dynamics, we examined a simplified system consisting of only the output node, an activator, and a deactivator (Figure 4.20). If the deactivator is slowly turned off while the activator is held constant, the simple system approximates the double inhibition circuit. If the activator is slowly turned on while the deactivator is held constant, the simple system approximates the double inhibition circuit. To maximize symmetry, k_{cat} = 1 and K_m = 0.5 for both the activator and the deactivator. We solved for the steady state concentration of active output C_{ss} as a function of activator concentration A, deactivator concentration D, and K_m of K:

$$C_{ss} = \frac{-1}{2a}\left(b + \sqrt{b^2 - 4ac}\right)$$

where

$$a = D - A$$

$$b = -KA - 2DA - D + A - KD$$

$$c = KA + DA$$

If k_{cat}'s and K_m's are not equal for the activator and deactivator but instead have values k_A, k_D, K_A, and K_D,

$$a = D - \frac{k_A}{k_D}A$$

$$b = \frac{k_A}{k_D}A(1 - K_D - D) - (K_AD + DA + D)$$

$$c = \frac{k_A}{k_D}A(K_D + D)$$

In Figure 4.20, we show the steady state amount of active output for different concentrations of activator and deactivator. The steady state output diagram shows the source of asymmetric dynamical behavior in double inhibition vs double activation circuits. The system is most sensitive to changes in activator or deactivator concentrations when those concentrations are small (near zero). In the double activation circuit, the activator is being turned on by the input, and thus the activator concentration regime is exactly where steady state output is most sensitive to changes. In contrast, in double inhibition circuits, deactivator is being turned off by the input. The deactivator concentration is initially outside the regime where output is most sensitive to changes in deactivator concentration. Thus output remains fairly constant until deactivator levels sink low enough to enter that regime.

Other observations from comparing double inhibition and double activation timecourses and temporal dose responses are recapitulated in the steady state diagram. In the increasing activator system, a larger concentration of constitutive deactivator pushes the curve to the right, yielding a larger EC50 but losing steepness. This leads to the tradeoff between temporal dose response steepness and trigger time discussed above. In the decreasing deactivator system, a larger concentration of constitutive activator also flattens the deactivator-output curve and increases EC50. However, because the deactivator is being decreased, a larger EC50 results in a shorter time needed to activate output. Thus in the double inhibition circuit, there is no tradeoff between temporal dose response steepness and trigger time.

The steady state analysis suggests a few ways to manipulate the double inhibition and double activation systems to bring their trigger times into the same range. One could cap the concentration of deactivator to a small amount so that the double inhibition system starts in the regime where output is sensitive to small changes in deactivator concentration. Conversely, a simple way of further increasing the delay of a double inhibition circuit is to increase the starting concentration of deactivator. Adding cooperativity and Hill functions could change the shape of the steady state curves and move the regime of high sensitivity to the right, allowing the double activation circuits to show some delay before output is activated.

In our enumerative search, we ignored all circuits where output turned off in response to input turning on. However, the steady state curves suggest that a system where an activator is turned off should have longer trigger time than a system where a deactivator is turned on. We tested an “activator on” circuit and a “deactivator off” circuit and found that this was indeed the case (Figure 4.21).

4.2.5 Trigger time tunability in response to individual parameters

We measured tunability by quantifying the effect of perturbing each parameter in a kinetic filtering circuit. Beginning with the five canonical kinetic filtering topologies, we took all parameter sets that resulted in kinetic filtering behavior. Each parameter other than input k_{cat} and K_m of each parameter set was perturbed by a factor of 10 in each direction, and trigger times of the resulting temporal dose response curves were measured (Figure 4.22). Two example perturbations in coherent feed forward loop circuits are shown in Figure 4.23A. Each parameter's sensitivity was quantified by the fold change in trigger time upon parameter perturbation. A tunable parameter was one where a 10-fold change in parameter strength led to at least a 1.5-fold change in trigger time in either direction.

The most tunable architectures, the buffering double inhibition and coherent feed forward loop topologies, each had 3 tunable parameters, while the positive feedback AND circuit had one and the positive feedback OR and bistable buffering inverter circuits had none (Figure 4.23B, Table 4.1).

Parameter	/10 mean	/10 fraction	*10 mean	*10 fraction	Max change
Positive feedback OR					
kcat AA	0.15	0.01	0.11	0.40	0.04
Km AA	0.90	0.49	1.30	0.26	0.44
kcat FA		0.00	0.09	0.48	0.04
Km FA	1.39	0.71	0.69	0.15	0.98
Positive feedback AND					
kcat AA	10.22	0.21	0.10	0.94	2.10

Km AA	0.83	1.00	1.77	0.71	1.25
kcat FA	0.79	0.43	1.33	0.18	0.34
Km FA	1.32	0.18	0.79	0.44	0.35
Buffering double inhibition					
kcat AB	13.09	0.29	0.10	0.56	3.85
Km AB	0.90	0.97	1.70	0.64	1.09
kcat FA	0.18	0.60	10.25	0.47	4.86
Km FA	2.61	0.83	0.49	0.72	2.18
kcat BC	0.84	0.13	1.35	0.55	0.74
Km BC	1.04	0.99	0.95	0.76	1.02
kcat FB	0.68	0.42	1.92	0.18	0.34
Km FB	1.36	0.48	0.82	0.46	0.65
kcat FC	1.10	0.39	0.90	0.23	0.43
Km FC	0.98	0.94	1.07	0.78	0.92
Bistable buffering inverter					
kcat AB		0.00	0.08	0.01	0.00
Km AB	0.41	0.17	5.93	0.08	0.46
kcat AC	0.98	0.36	1.01	0.60	0.61
Km AC	1.01	0.88	0.98	0.61	0.89
kcat FA	0.20	0.04	8.93	0.01	0.10
Km FA	1.41	0.55	0.71	0.35	0.78
kcat BA	2.09	0.17	0.66	0.31	0.35
Km BA	0.95	0.74	1.16	0.51	0.71
kcat BB	0.30	0.05		0.00	0.01
Km BB	2.05	0.57	0.72	0.39	1.16
kcat FC	1.03	0.61	0.94	0.24	0.63
Km FC	0.97	0.52	1.02	0.62	0.63
Coherent feed forward loop AND					
kcat AB	3.73	0.61	0.39	0.57	2.28
Km AB	0.77	0.92	2.15	0.80	1.72
kcat AC	3.96	0.20	0.29	0.88	0.78
Km AC	0.96	1.00	1.18	0.71	0.96
kcat FA	0.63	0.69	3.17	0.46	1.44
Km FA	1.73	0.76	0.72	0.71	1.31
kcat BC	3.96	0.20	0.29	0.88	0.78
Km BC	0.95	0.99	1.21	0.76	0.95
kcat FB	0.66	0.71	2.76	0.63	1.74
Km FB	1.37	0.92	0.80	0.84	1.26

kcat FC	0.45	0.78	3.43	0.33	1.12
Km FC	1.08	0.99	0.87	0.86	1.07

Table 4.1 Tunability of k_{cat}'s and K_m's of canonical kinetic filtering architectures. Columns: "/10 mean" = mean fold change in trigger time after dividing parameter by 10; "/10 fraction" = fraction of circuits where dividing parameter by 10 results in a kinetic filter; "*10 mean" = mean fold change in trigger time after multiplying parameter by 10; "*10 fraction" = fraction of circuits where multiplying parameter by 10 results in a kinetic filter; "max change" = Maximum of (/10 mean * /10 fraction) and (*10 mean * *10 fraction). "FA" denotes the constitutive regulator of A, and similarly for nodes B and C. Deeper blue color indicates larger fold changes in trigger time; deeper orange coloring indicates brittle parameters where tuning destroying kinetic filtering behavior; deeper green coloring indicates tunable parameters.

Most of the tunable parameters were k_{cat}'s, though in a few cases K_m's were also observed to tune trigger time. For positive feedback AND circuits, decreasing k_{cat} of the positive feedback link increases trigger time. For buffering double inhibition circuits, trigger time can be tuned by manipulating k_{cat} of the first inhibition link as well as k_{cat} or K_m of the constitutive deactivator of the node where input is applied. For coherent feed forward loop circuits, trigger time can be tuned by changing k_{cat}'s and K_m's of the first link on the left arm of the circuit.

4.2.6 Effects of chaining motifs

We asked how chaining multiple copies of the same kinetic filtering motif affects trigger time. We took the buffering double inhibition and coherent feed forward loop as starting architectures. For the buffering double inhibition, we simply added additional copies of the second two nodes, keeping the same parameters as in the original kinetic filtering topology (Figure 4.24). For the coherent feed forward loop, we added additional nodes into the left (longer) arm of the feed forward loop, keeping the same parameters as in the first step of the left arm (highlighted in red, Figure 4.24).

We measured fold change in trigger time of many kinetic filtering parameter sets of each of the two architecture types when chaining multiple copies of each basic motif (Figure 4.24). Buffering double inhibition circuits typically lost kinetic filtering ability upon chaining because the later motifs were not being subjected to the correct amplitude of input to which they are temporally sensitive. In contrast, adding additional nodes into the longer left arm of the coherent feed forward loop can increase trigger time by more than one order of magnitude. However, not all coherent feed forward loop parameter sets showed increases in trigger time; in some parameter sets, adding more nodes to the left arm actually decreased trigger time.

Representative timecourses of each of these two coherent feed forward loop behaviors are shown in Figure 4.25. Circuits where trigger time increases with more nodes in the longer arm are those where the longer arm has slower dynamics than the shorter arm. If the longer arm has fast dynamics, then adding additional nodes can actually steepen the activation from that arm that the output node sees, resulting in slightly faster dynamics in the output node and hence shorter trigger time.

4.2.7 Transcriptional and mixed regulation circuits

The coherent feed forward loop is the only kinetic filtering topology identified as such in real biological systems (see Chapters 1 and 5). Unlike our system, however, in biology the coherent feed forward loop is often encoded with one transcriptional arm and one enzymatic arm. Because we observe only short trigger times in our enzymatic-only coherent feed forward loops, we also simulated transcription-only and mixed regulation coherent feed forward loop circuits. A buffering double inhibition circuit was also simulated as a comparison. Both simulations sampled 100,000 parameter sets. See Chapter 2 for simulation details.

Using transcriptional or mixed regulation increases trigger time for both buffering double inhibition and coherent feed forward loop circuits (Figure 4.26). While coherent feed forward loop circuits continue to have shorter trigger times than buffering double inhibition circuits under all regulatory conditions, using transcription does alleviate the differences to some extent.

4.3 RESET TIME

Because of the difference in off dynamics across the five types of kinetic filters, we suspected that the five types might display different behavior when presented with multiple short pulses. We measured the reset time of the 3-node circuits in Figure 4.27 (simulation details in Chapter 2).

By definition, all kinetic filtering circuits have temporal ultrasensitivity score ≥ 0.5, which means that the ratio between input duration needed to reach 10% maximum output amplitude and input duration needed to reach 90% amplitude is ≥ 0.5. This further means that the input duration needed to reach 10% amplitude is at least half the duration needed to reach 90% amplitude. Thus if an input pulse long enough to activate to 90% is divided into two pulses of equal duration, each of those pulses should activate output to at most 10% of maximum. To measure reset time, we took those two pulses and separated them by varying amounts of time, measuring maximum output amplitude for each separation length. We then measured the separation duration necessary to achieve 50% of maximum output amplitude; this is the reset time (Figure 4.27 and Chapter 2).

Unsurprisingly, positive feedback AND circuits had the longest reset times. Since positive feedback AND circuits require a poor constitutive deactivator in order to activate at all (see below on preferred parameter regimes), they show a very slow, linear decrease in active output after input has been removed (Figures 4.3-4.5). This means that positive feedback AND circuits require a long time to reset active output levels back to initial levels; additionally, they may be good frequency detectors if they can integrate effects of many short inputs but ignore short inputs separated by a longer off time.

Buffering double inhibition circuits had the largest range of reset times, reflecting their ability to exhibit short-term memory or turn off immediately when input has turned off (Figure 4.7). Positive feedback OR and bistable buffering inverter circuits had the shortest reset times.

4.4 PREFERRED PARAMETER REGIMES

Although the five architecture classes share kinetic filtering behavior, they present a diverse array of parameter requirements and behaviors (Figures 4.28-4.32). Kinetic filtering restricts the allowable parameter regimes of some but not all regulatory links each in circuit type, including many of the constitutive activators and deactivators.

Positive feedback OR circuits require the constitutive deactivator to operate in the saturated regime, and k_{cat}'s of the positive feedback and deactivator must be balanced to allow bistability (Figure 4.28). Positive feedback AND circuits require the constitutive deactivator to operate in the linear regime with very slow kinetics, which is essential for allowing input signal to "leak" through (Figure 4.29). Among buffering double inhibition circuits, both inhibition links prefer the saturated regime, while the constitutive activator of the buffering node prefers the linear regime and slow kinetics, allowing even a small amount of node A to begin turning off the buffering node (Figure 4.30). The bistable buffering inverter circuit prefers input and deactivating links to be in the saturated regime and the constitutive activator of node C to be in the linear regime (Figure 4.31). In coherent feed forward loop circuits, all regulations on the output node prefer the saturated regime, conferring zero-th order ultrasensitivity on the output node such that output responds all-or-none to the presence of its activators (Figure 4.32).

For the positive feedback OR circuit, we were able to analytically describe the parameter constraints on positive feedback and constitutive deactivator k_{cat}'s to ensure bistability and hence kinetic filtering behavior. Consider a one-node positive feedback system where node A catalyzes its own activation and a constitutive deactivator catalyzes its deactivation. The rate of change of A for a simple positive feedback circuit with constitutive deactivator F is described using total quasi-steady-state Michaelis-Menten equations as

$$\frac{dA}{dt} = \frac{k_A A(1-A)}{K_A + A + (1-A)} - \frac{k_F F A}{K_F + F + A}$$

where total concentration of A is 1; k_A and K_A and k_F and K_F are k_{cat} and K_m of positive feedback and constitutive deactivation respectively. We simplify to

$$\frac{dA}{dt} = \frac{k_A A(1-A)}{K_A + 1} - \frac{k_F F A}{K_F + F + A}$$

At steady state,

$$\left.\frac{dA}{dt}\right|_{A=A_{ss}} = 0$$

Let $\phi = k_F/k_A$. Solving for steady state concentration of A, we find 3 roots:

$$A_{ss0} = 0$$

$$A_{ss+,-} = \frac{1}{2}\left(1 - K_F - F \pm \sqrt{(1-K_F-F)^2 - 4((\phi F(K_A+1) - (K_F+F))}\right)$$

When the positive feedback system is bistable, there are 3 positive real values for A_{ss}. We can solve for the lower and upper bounds of ϕ as follows. At the lower bound of ϕ, $A_{ss-} \to 0$, implying

$$4(\phi F(K_A+1) - (K_F+F)) \to 0$$

and the lower bound of ϕ to be

$$\phi > \frac{K_F + F}{F(K_A+1)}$$

The upper bound of ϕ is found by requiring the discriminant for the expression of $A_{ss+,-}$ to be positive:

$$(1-K_F-F)^2 - 4(\big(\phi F(K_A+1) - (K_F+F)\big) > 0$$

yielding

$$\phi < \frac{(1 - K_F - F)^2 + 4(K_F + F)}{4F(K_A + 1)}$$

So the region of bistability of this simple positive feedback circuit with constitutive deactivator is defined by the following requirement on the ratio of k_{cat}'s of the deactivator and positive feedback:

$$\frac{K_F + F}{F(K_A + 1)} < \frac{k_F}{k_A} < \frac{(1 - K_F - F)^2 + 4(K_F + F)}{4F(K_A + 1)}$$

Steady state concentration of A is plotted as a function of k_F/k_A for a positive feedback OR kinetic filter in the bifurcation diagram in Figure 4.33 (top). For ease of plotting, we have normalized k_F/k_A to its value in the starting parameter set. The bifurcation diagram shows a region of one stable steady state (purple) where deactivation strength is too high and the circuit is always off and a region of one stable and one unstable steady state (green) where positive feedback strength is high enough to always fully turn on the circuit if it strays from the fully off state. Between the two regions lies the regime of one low and one high stable steady state separated by an unstable steady state. In this regime, circuits are bistable and exhibit kinetic filtering behavior.

The rate balance plots in Figure 4.33 (bottom) show how adding input to the system destroys the low and intermediate steady states, kicking the system into the high steady state. When input turns off, the system then falls into the high steady state and remains there. Varying the deactivation k_{cat} shows how only intermediate values of k_{cat} allow activation and deactivation rate curves to cross each other three times and yield two stable steady states.

A bifurcation diagram and rate balance plots for a representative positive feedback AND circuit are shown in Figure 4.34. This type of kinetic filter is not bistable and can only achieve the on state when input is on. Figure 4.34 shows that a huge range of constitutive deactivator k_{cat}'s allow the circuit to turn on; only extremely high values of k_{cat} prevent the circuit from turning on.

4.5 INTEGRATORS AND ABSOLUTE TIMERS

We asked whether the circuits we identified as kinetic filters are integrating the input signal. Taking the set of minimal kinetic filtering topologies, we tested each kinetic filtering parameter set for integration behavior. We measured each parameter set's trigger time for a set of input durations greater and lesser amplitude (Figure 4.35 and see Chapter 2 for simulation details). Kinetic filters that are integrators should have shorter trigger time for higher amplitude inputs and longer trigger time for lower amplitude inputs. Kinetic filters that are absolute timers should have the same trigger time for all amplitudes of input. To quantify integration behavior, we simply measure the slope of trigger time versus input amplitude.

Kinetic filtering circuits are usually integrators (Figure 4.36). Positive feedback AND and bistable buffering inverter circuits are also able to act as absolute timers, depending on specific topology and parameters.

Circuits that act as absolute timers have parameter sets that activate node A maximally for even low input amplitudes, eliminating the effect of further increasing input strength. Thus if a topology has node A as the output node, that topology can never be an absolute timer. Tuning parameters of topologies that can be absolute timers can allow the same topology to act as an integrator or absolute timer in different regimes of input amplitude.

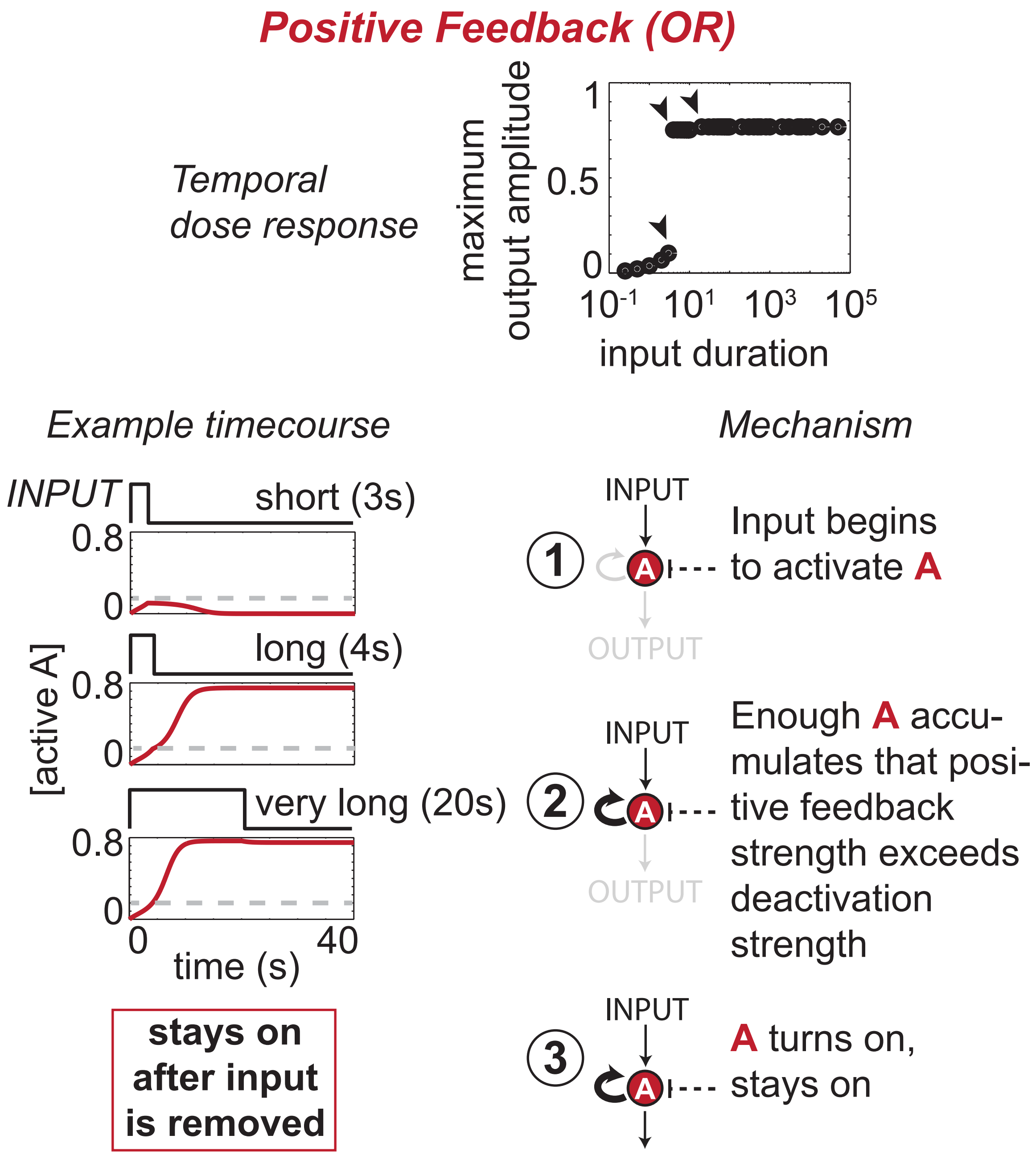


Figure 4.1. Representative temporal dose response, timecourse, and mechanism of a positive feedback OR circuit. Arrows on the temporal dose response indicate the input durations shown in the example timecourse (same is true in subsequent figures).

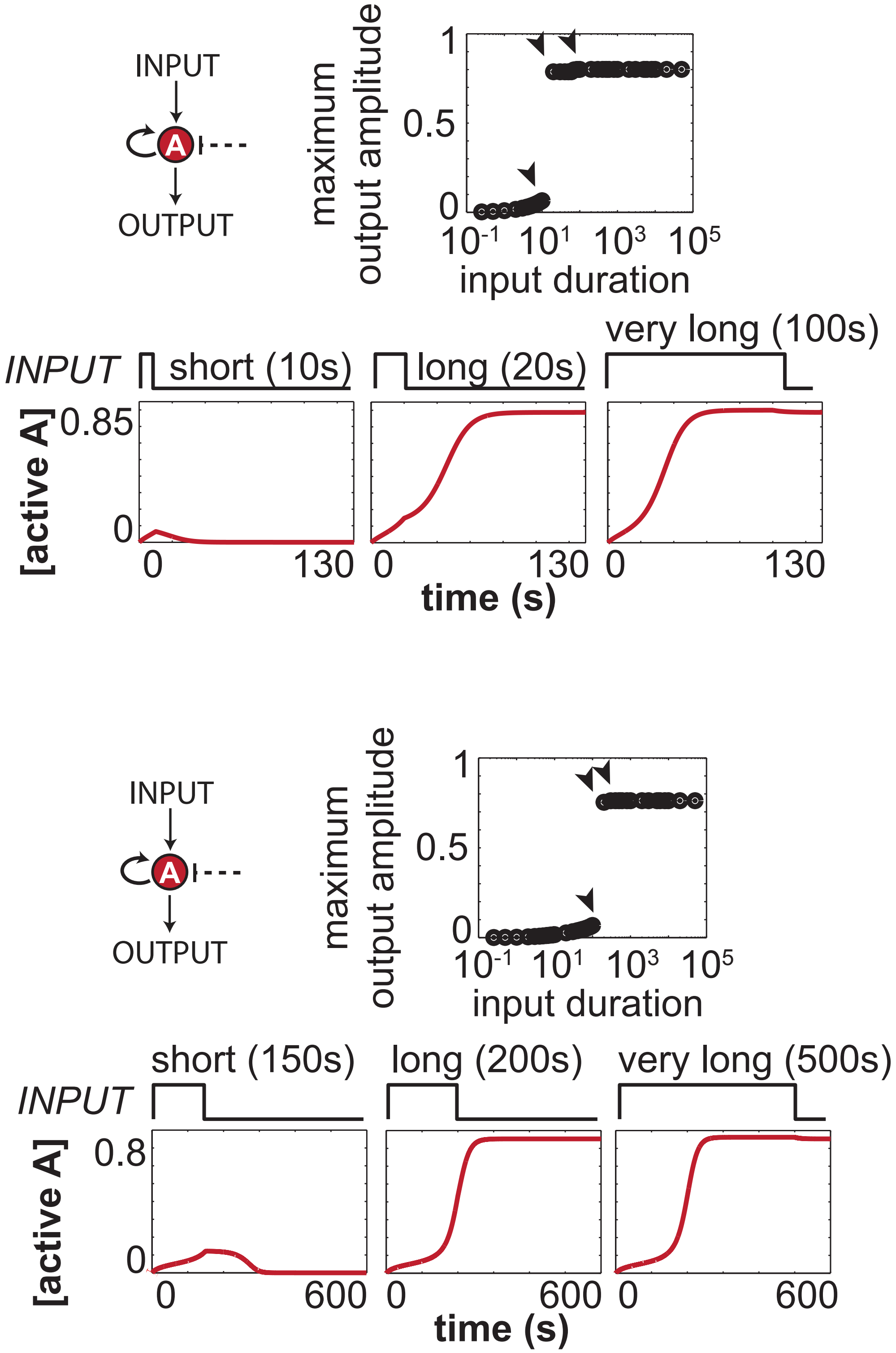


Figure 4.2. Additional temporal dose response curves and timecourses for 1-node positive feedback (OR) circuits.

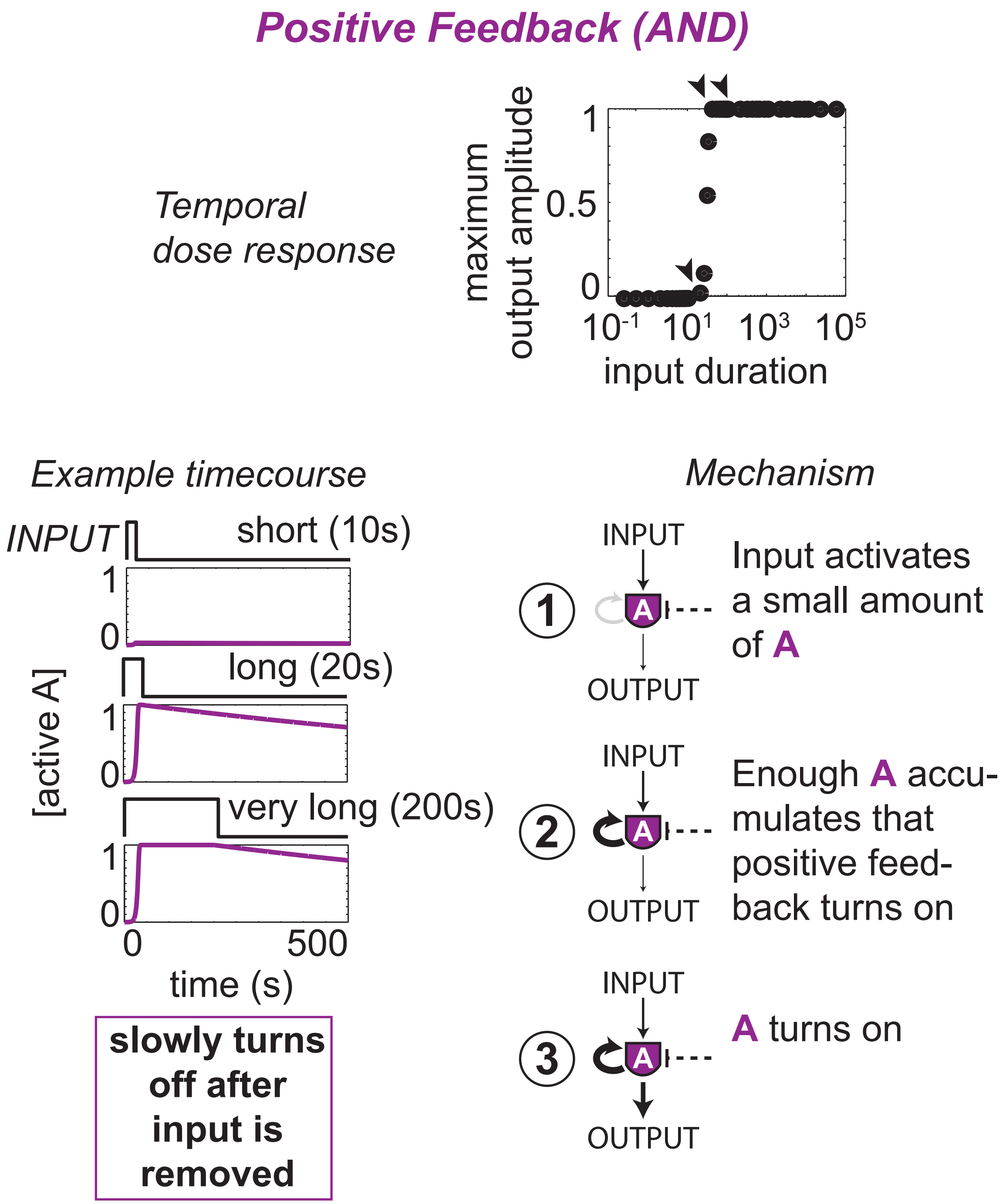


Figure 4.3. Representative temporal dose response, timecourse, and mechanism of a positive feedback AND circuit.

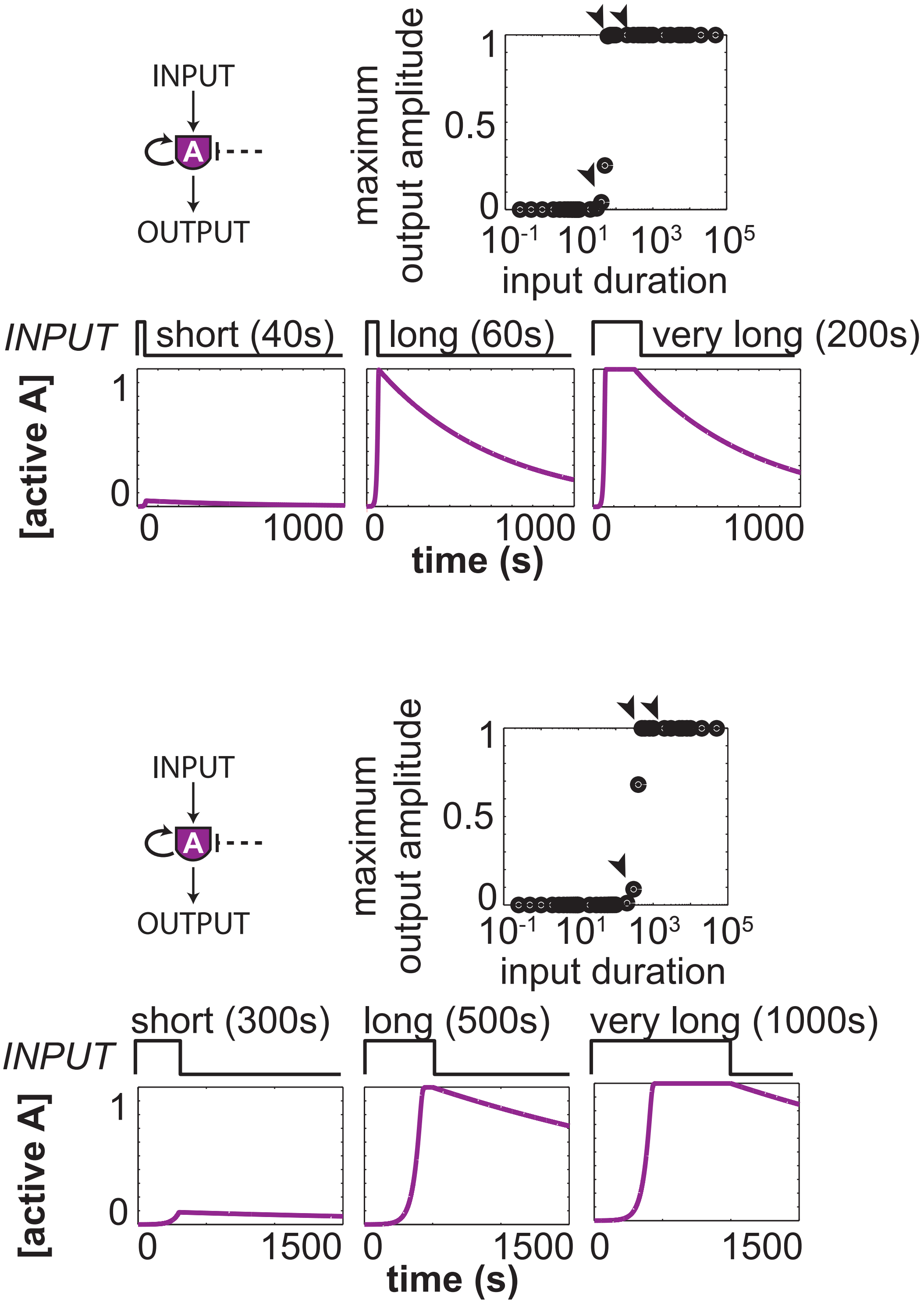


Figure 4.4. Additional temporal dose response curves and timecourses for 1-node positive feedback (AND) circuits.

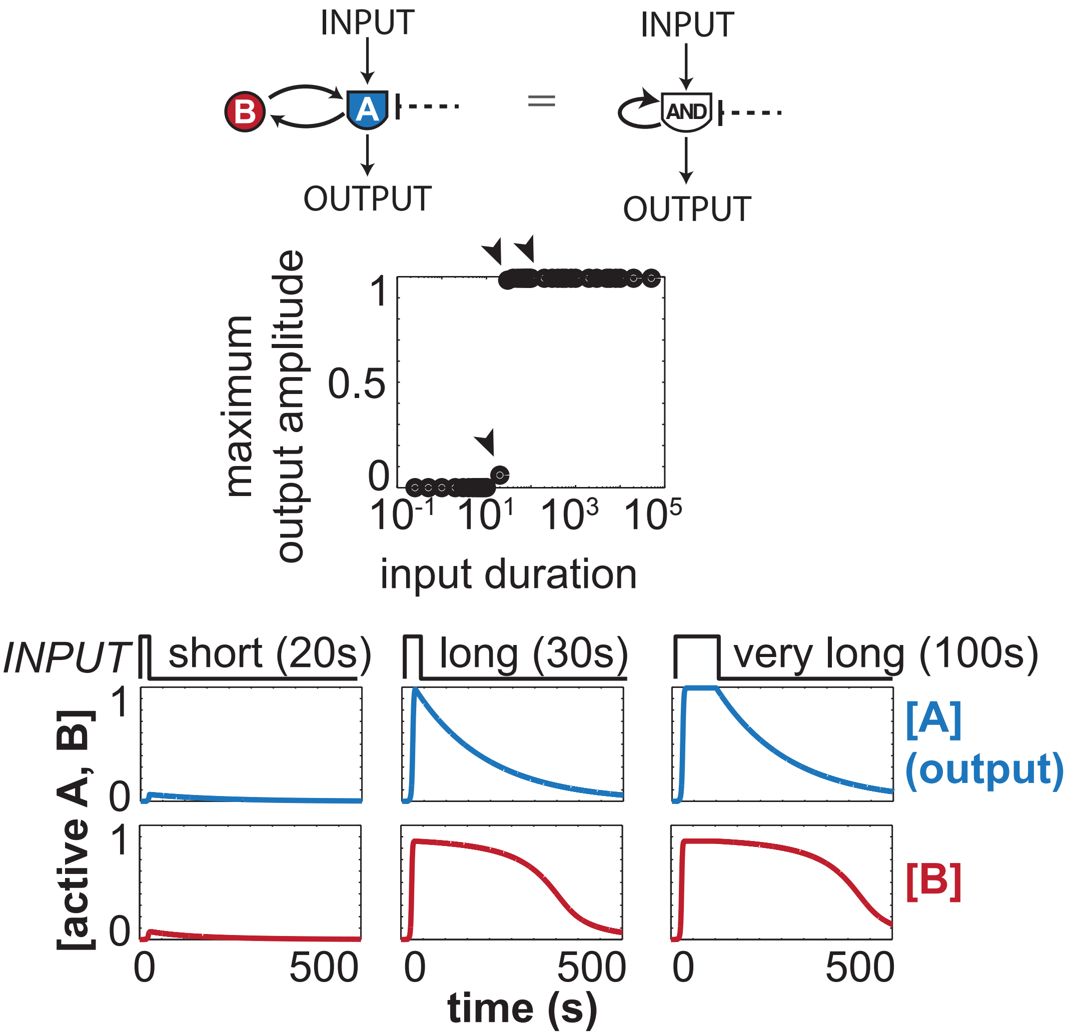


Figure 4.5. 2-node positive feedback AND circuit behaves like 1-node version.

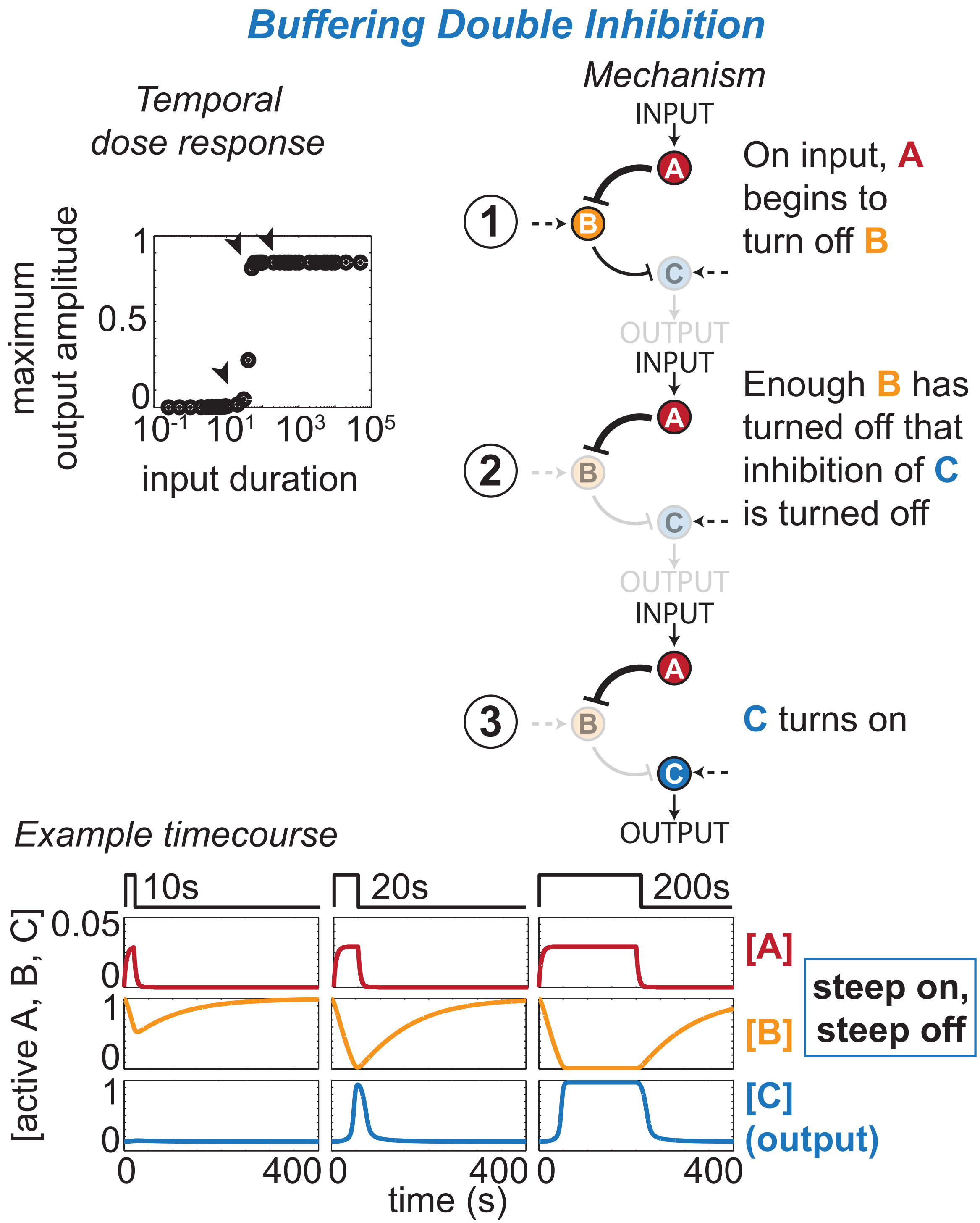


Figure 4.6. Representative temporal dose response, timecourse, and mechanism of a buffering double inhibition circuit with the inhibitors arranged in series.

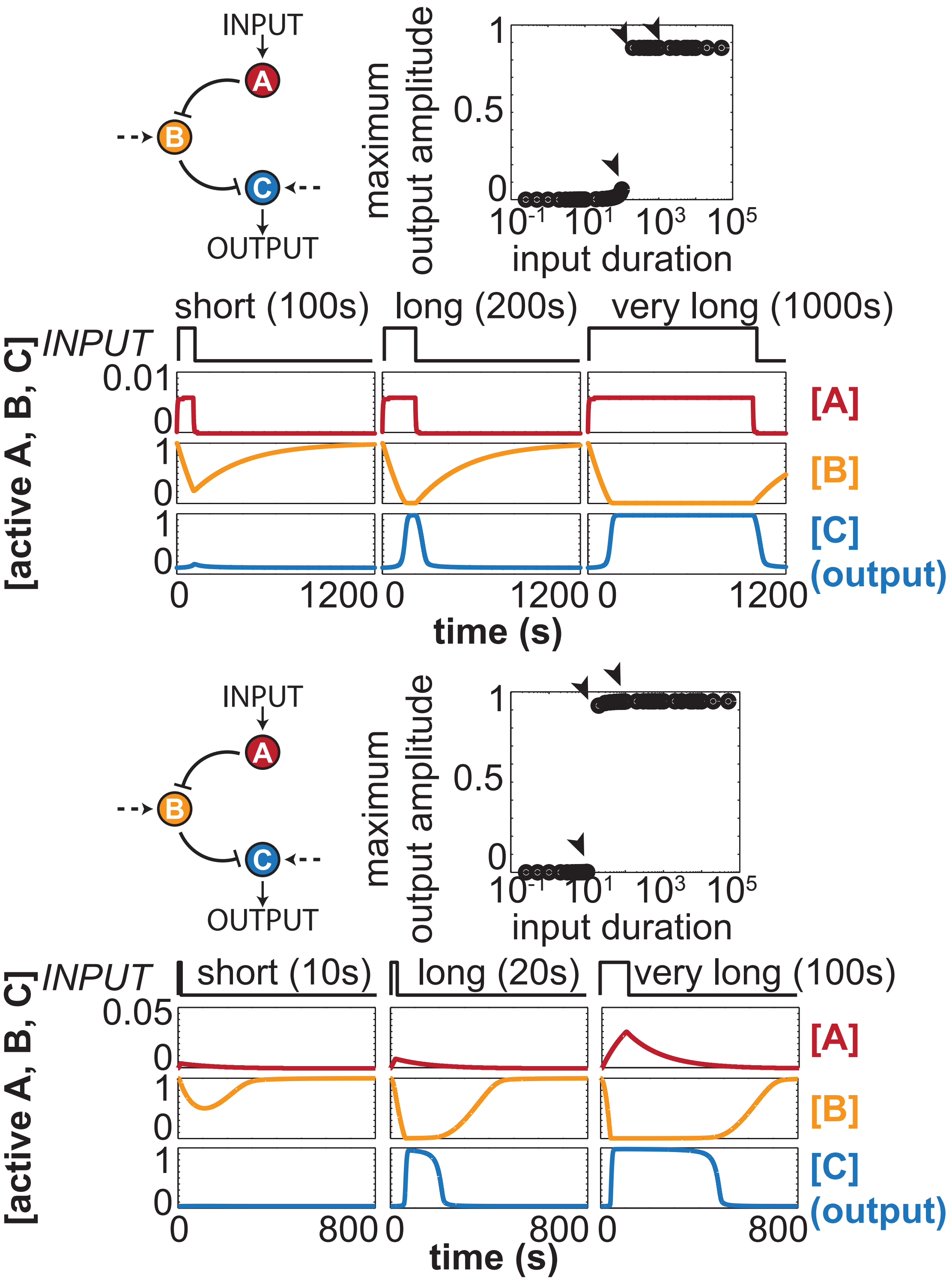


Figure 4.7. Additional temporal dose response curves and timecourses for 3-node buffering double inhibition circuits in series.

Buffering Double Inhibition (2-node)

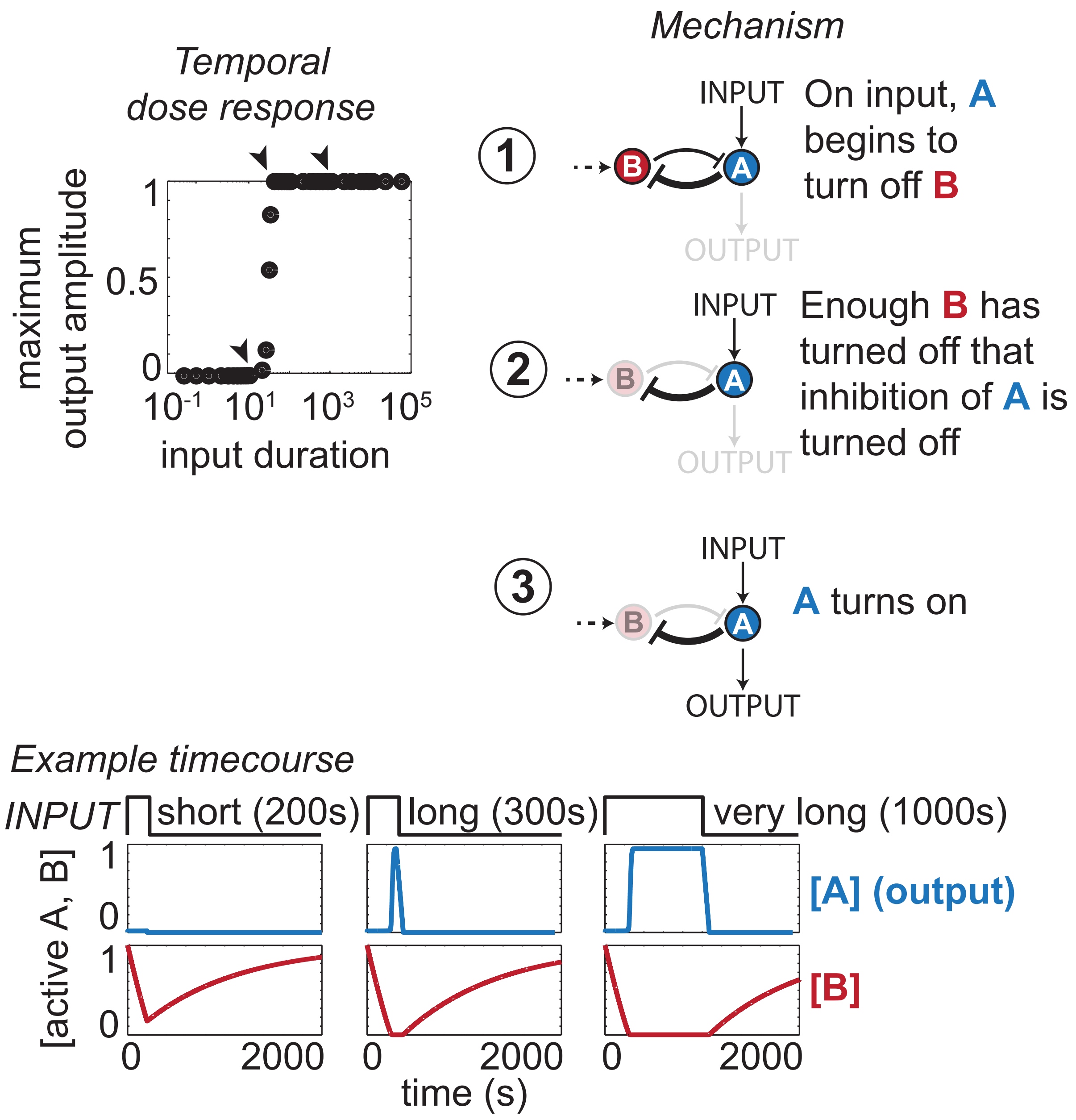


Figure 4.8. Representative temporal dose response, timecourse, and mechanism of a buffering double inhibition circuit with the inhibitors arranged in feedback.

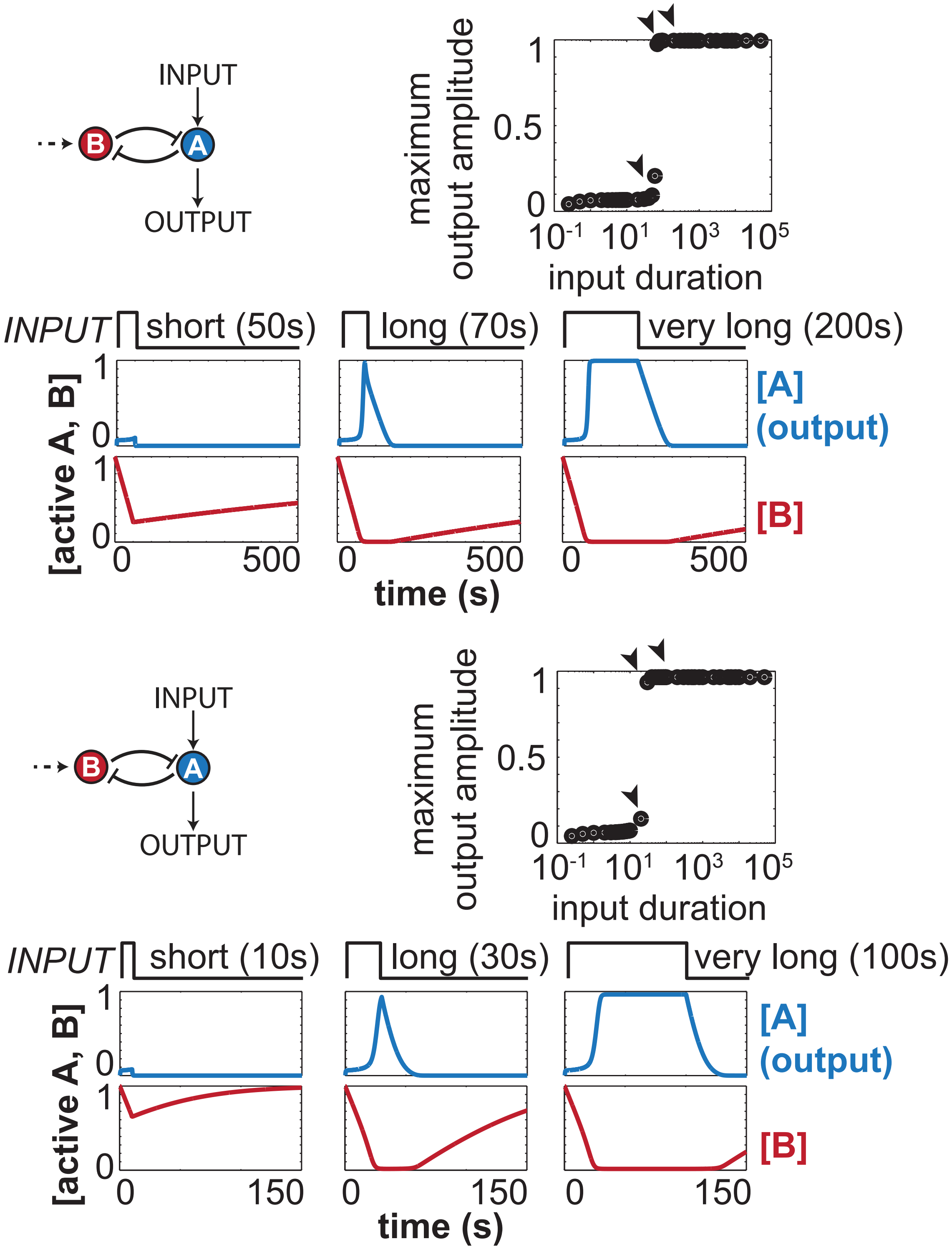


Figure 4.9. Additional temporal dose response curves and timecourses for 2-node buffering double inhibition circuits with inhibitions arranged in feedback.

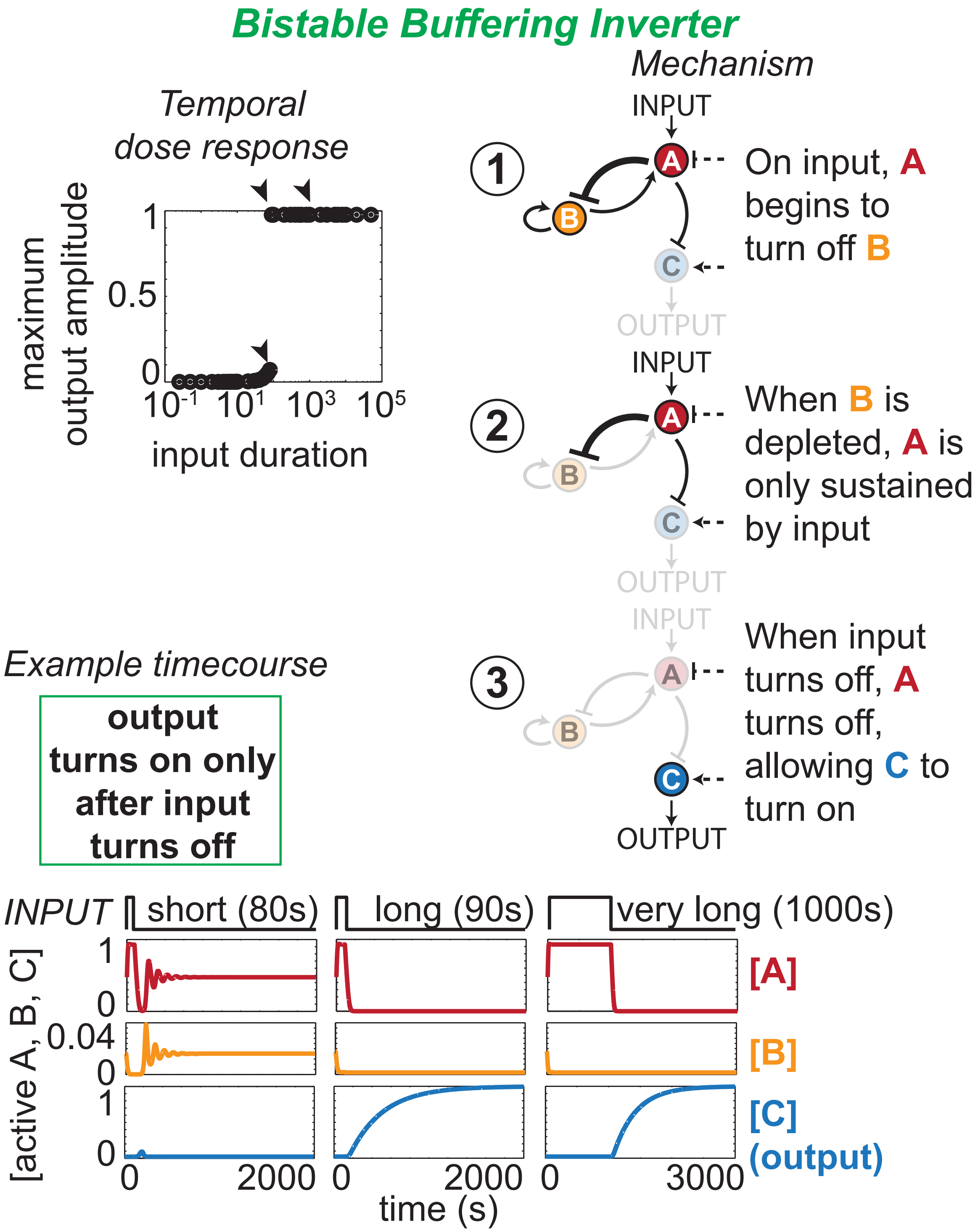


Figure 4.10. Representative temporal dose response, timecourse, and mechanism of a bistable buffering inverter circuit.

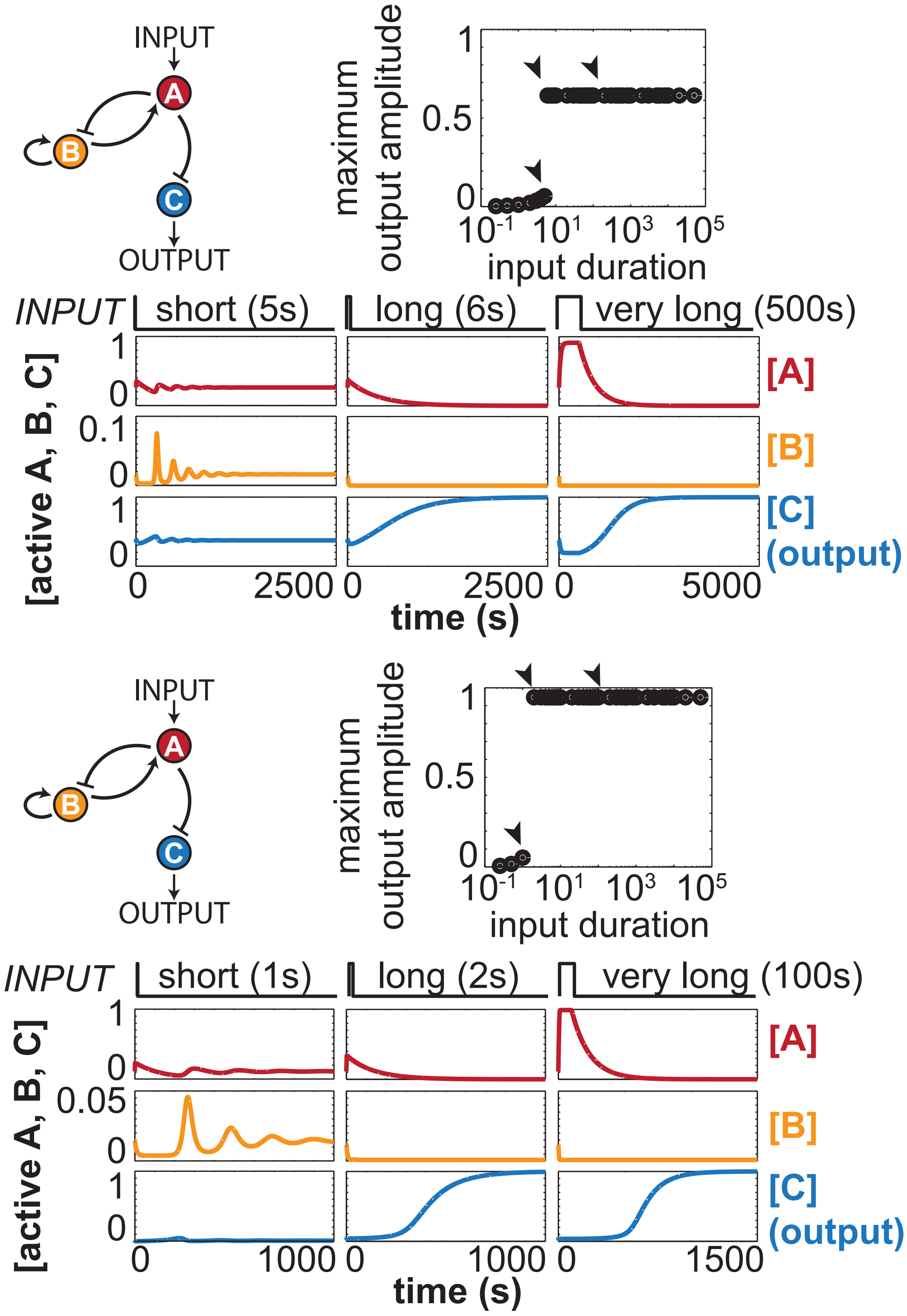


Figure 4.11. Additional temporal dose response curves and timecourses for bistable buffering inverter circuits.

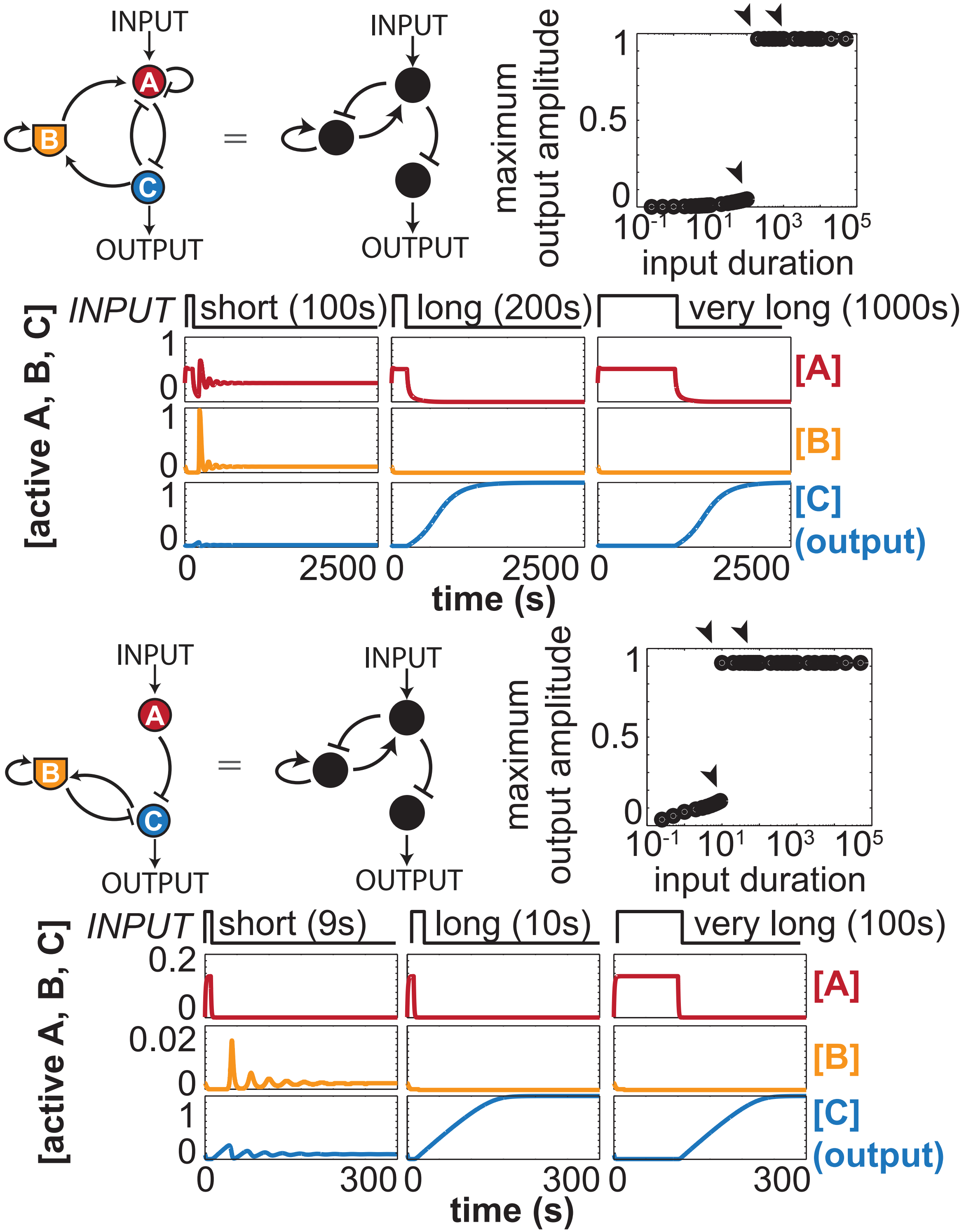


Figure 4.12. Minimal topologies with AND logic on node B are phenotypically equivalent to all-OR bistable buffering inverter topologies.

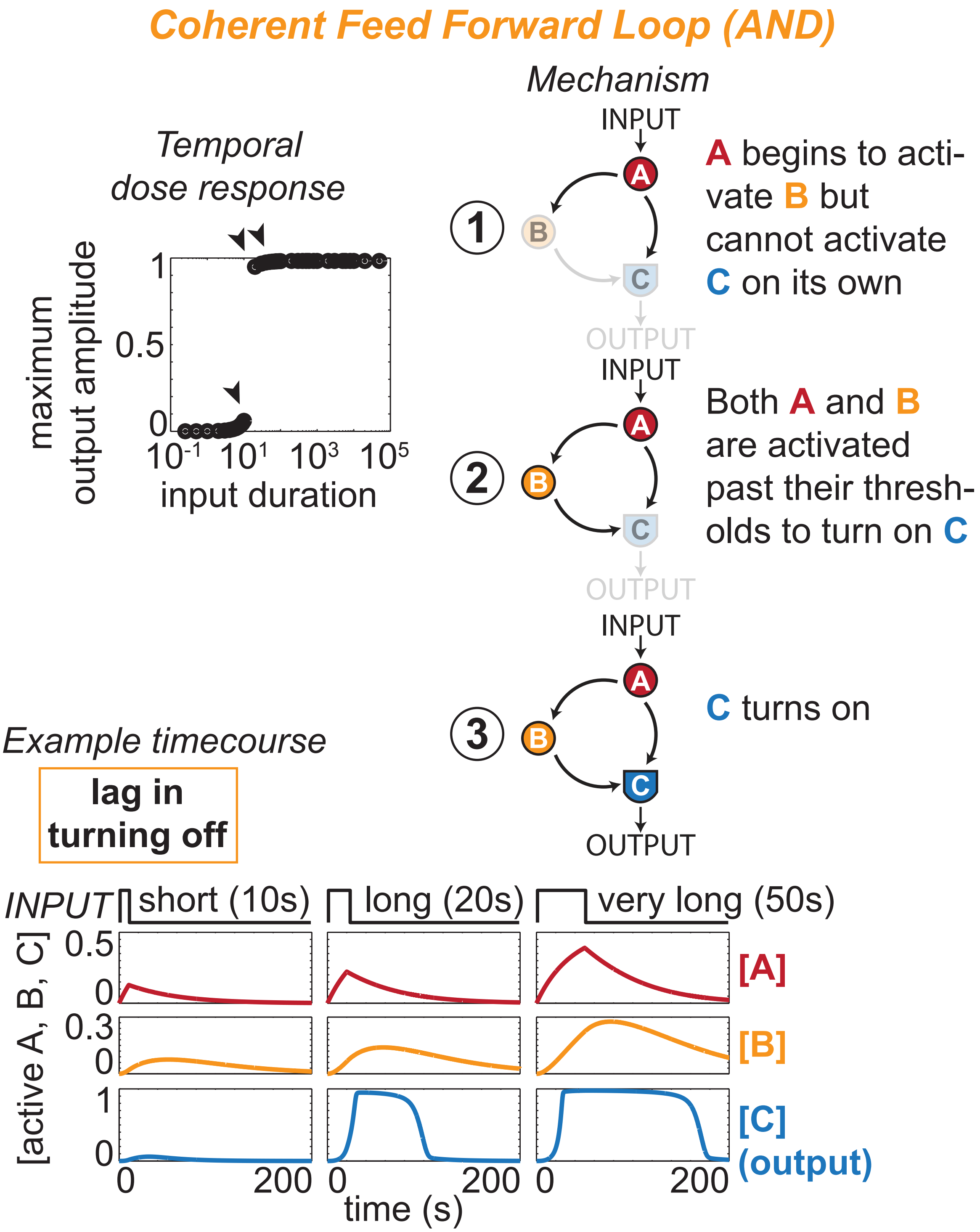


Figure 4.13. Representative temporal dose response, timecourse, and mechanism of a coherent feed forward loop with AND logic.

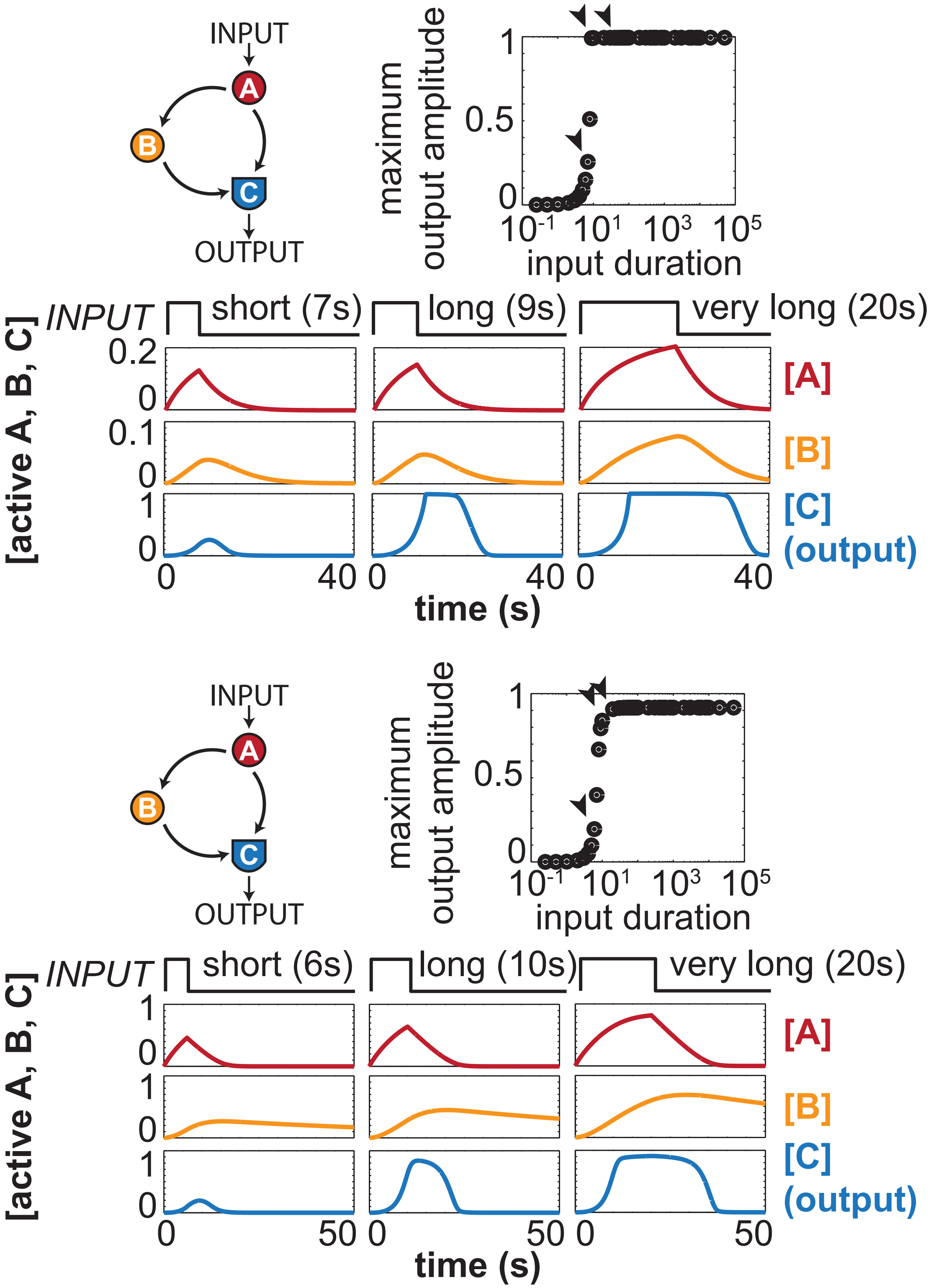


Figure 4.14. Additional temporal dose response curves and timecourses for coherent feed forward loop (AND) circuits.

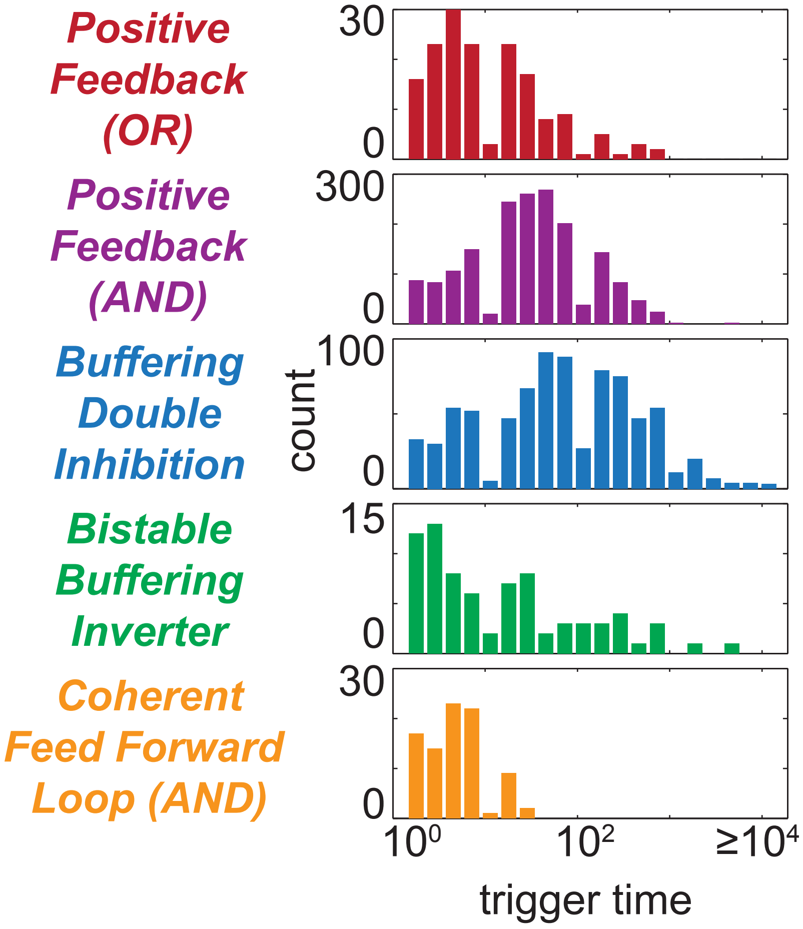


Figure 4.15. Distributions of trigger times of all parameter sets of all 25 minimal kinetic filtering topologies of 1, 2, or 3 nodes.

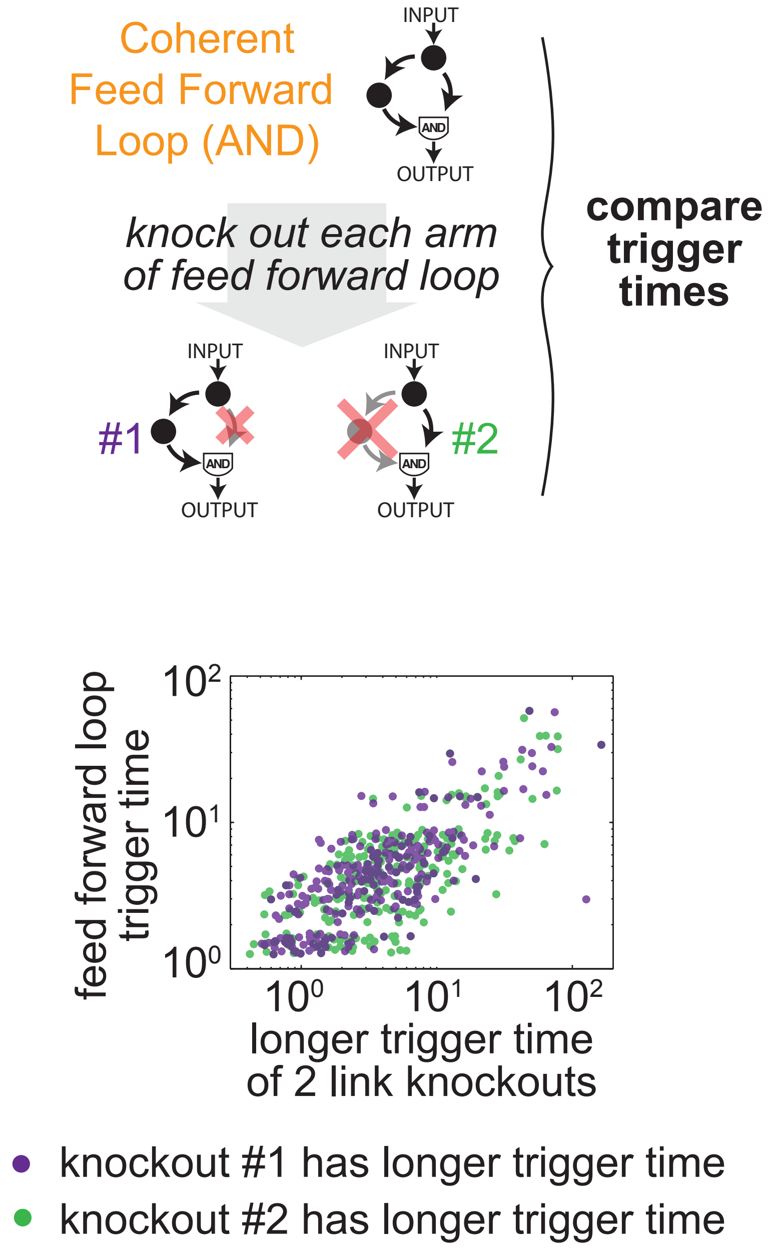


Figure 4.16. Trigger time of the slower arm governs overall coherent feed forward loop trigger time. Linear fit: slope = 0.32 ± 0.07, r = 0.70.

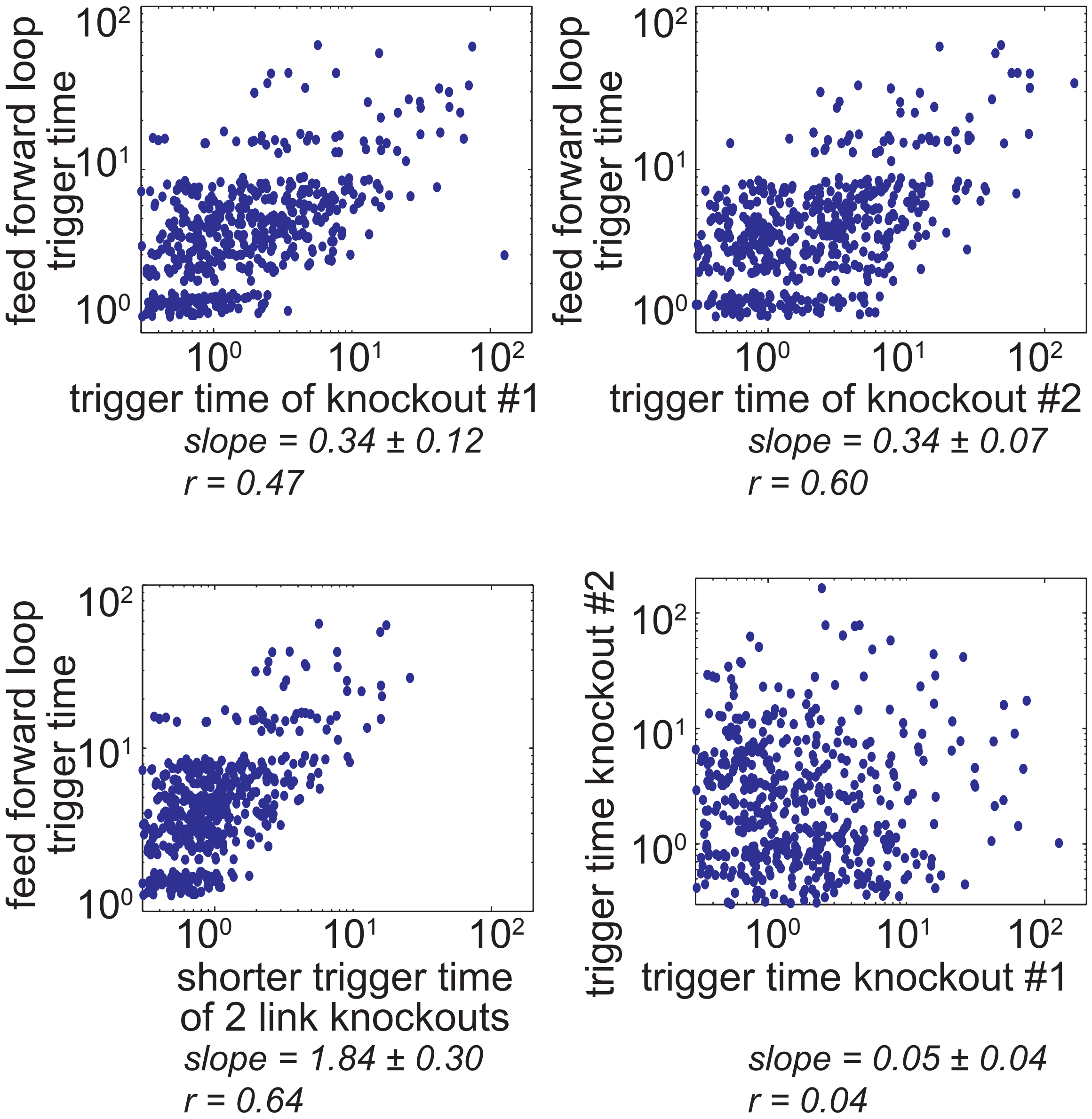


Figure 4.17. Trigger time of feed forward loop correlates less well with either knockout on its own and the shorter knockout trigger time. Knockout trigger times do not correlate with each other.

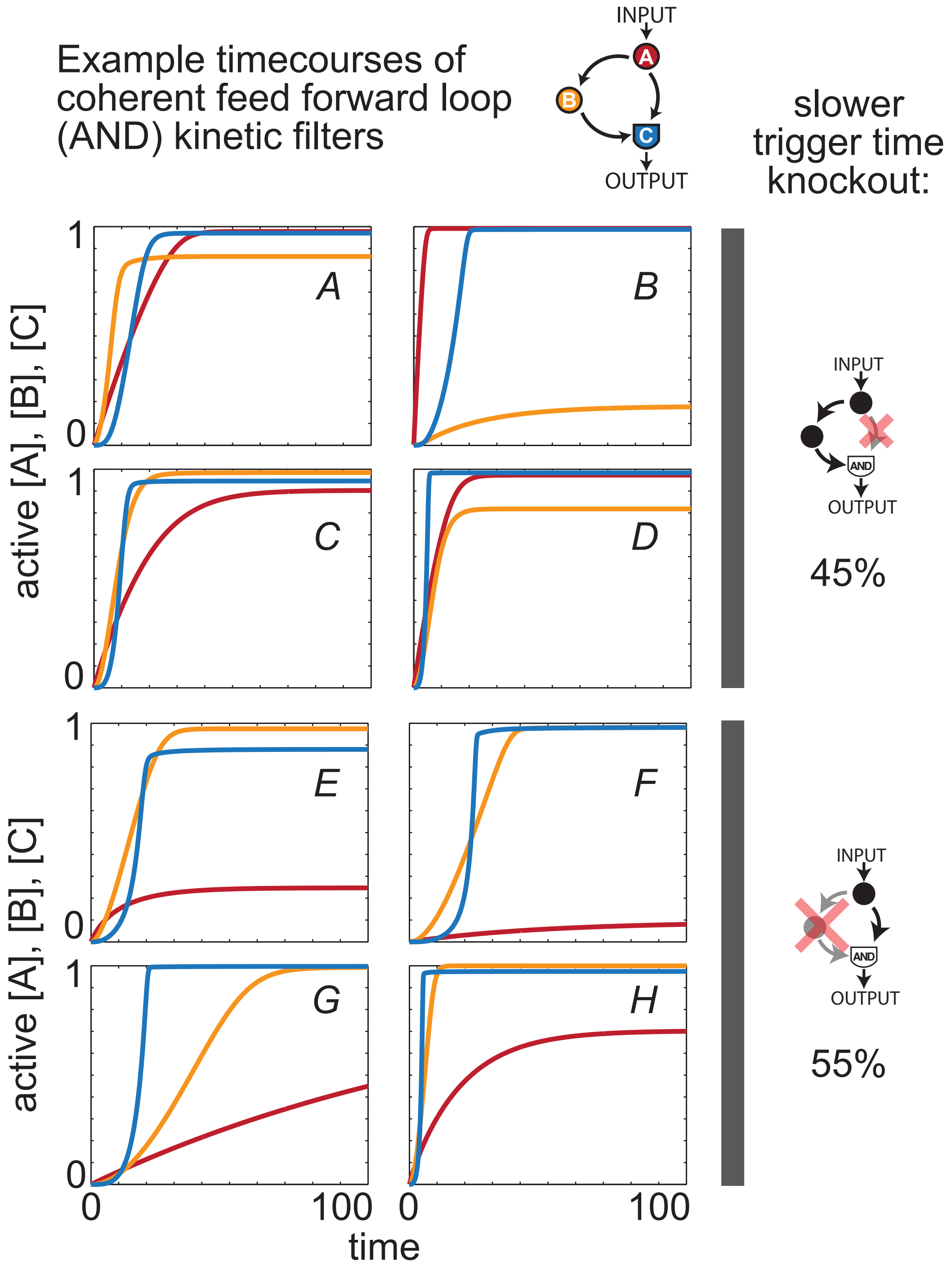
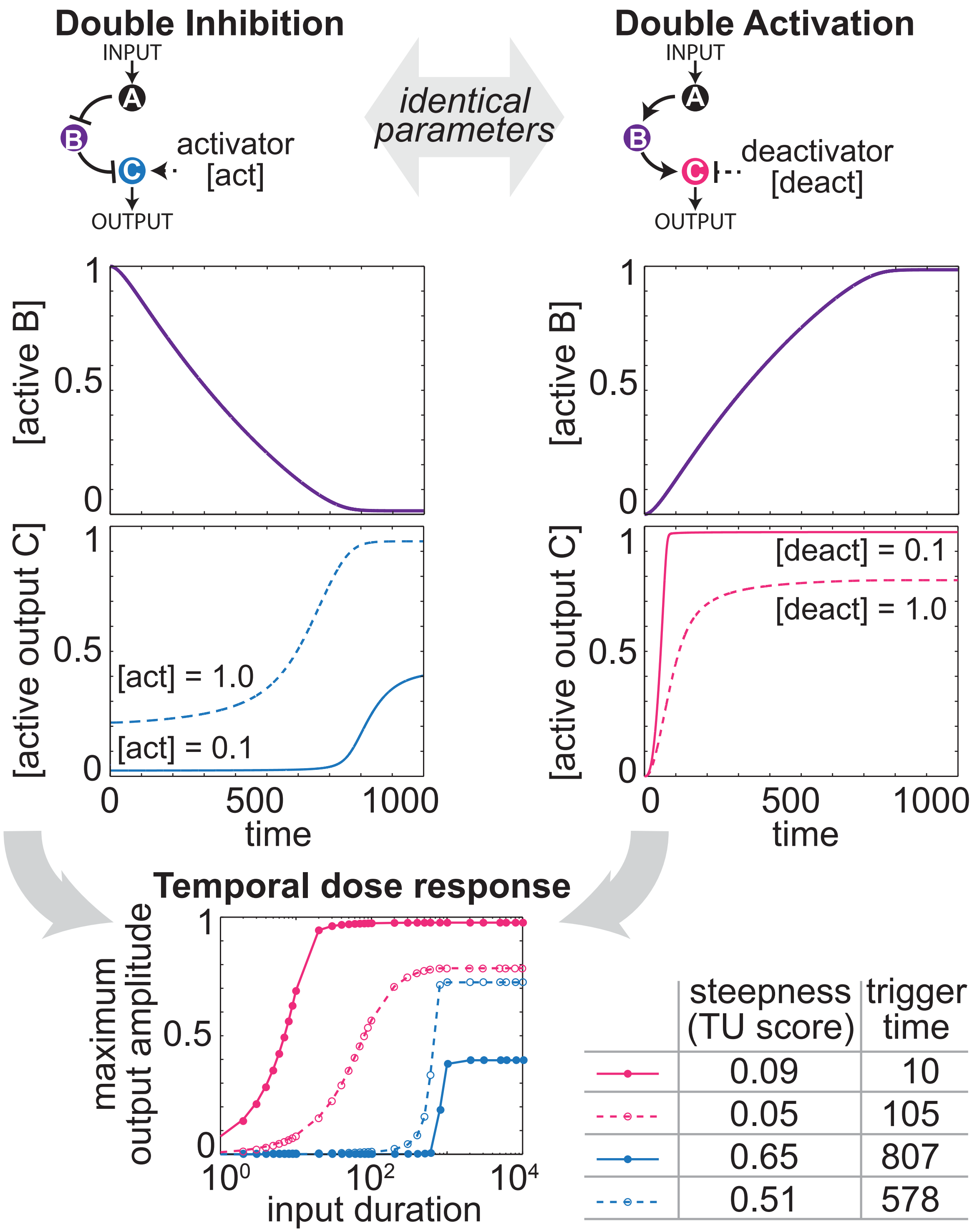


Figure 4.18. Representative timecourses of feed forward loop circuits where knocking out the right arm (top) or left arm (bottom) results in the slower trigger time. Single step input



	steepness (TU score)	trigger time
—●—	0.09	10
- -○- -	0.05	105
—●—	0.65	807
- -○- -	0.51	578

Figure 4.19. Comparison of timecourses and temporal dose responses of double inhibition and double activation circuits with identical parameters.

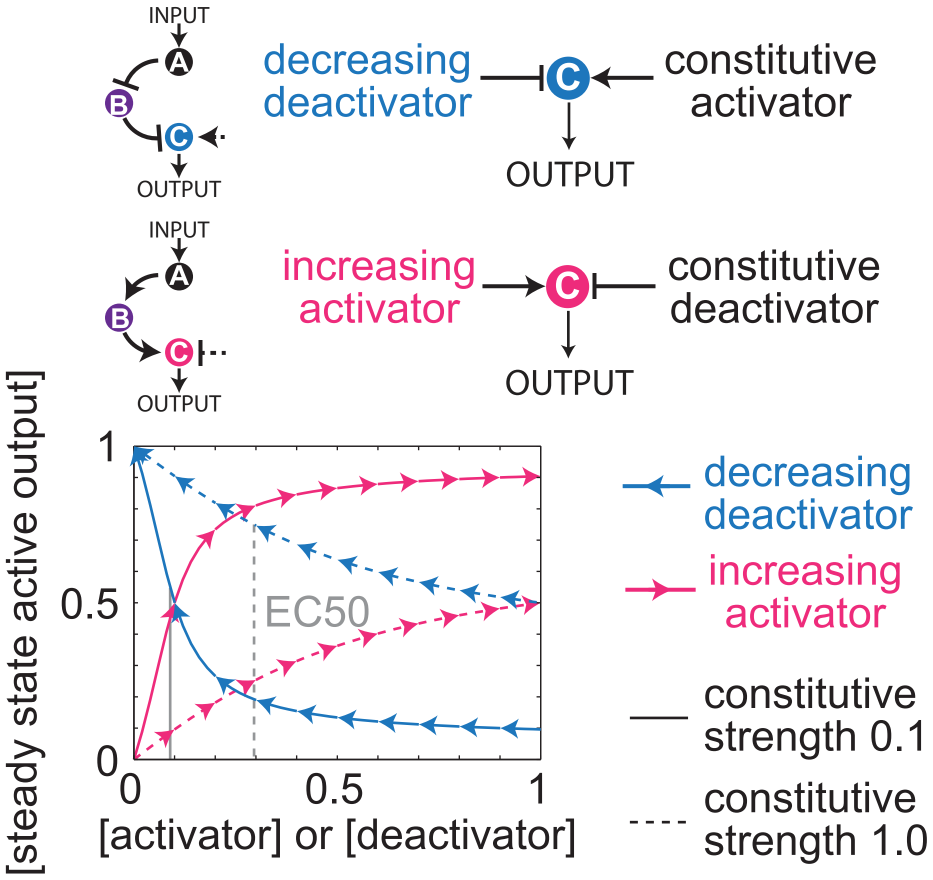


Figure 4.20. Steady state behavior drives asymmetry in activation vs deactivation.

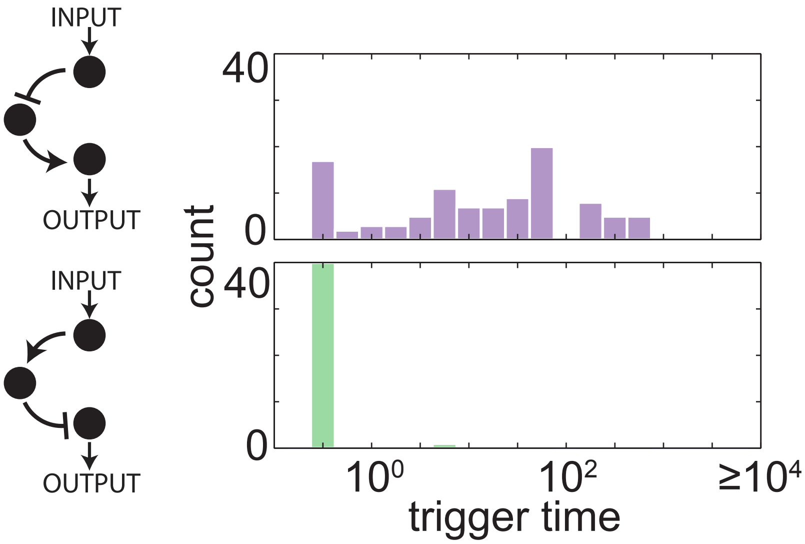


Figure 4.21. “Activator off” circuits have longer trigger time than “deactivator on” circuits. Each circuit was sampled at 10,000 parameter sets. Trigger time was measured only for parameter sets resulting in TU score ≥ 0.5.

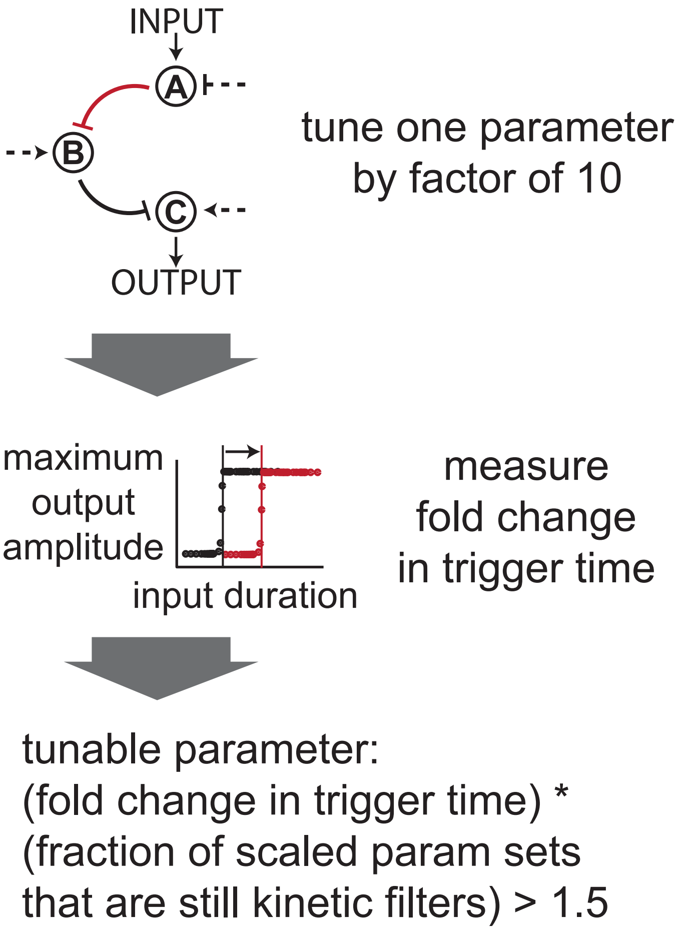

INPUT
A
B
C
OUTPUT
tune one parameter by factor of 10
maximum output amplitude
input duration
measure fold change in trigger time
tunable parameter:
(fold change in trigger time) *
(fraction of scaled param sets
that are still kinetic filters) > 1.5

Figure 4.22. Measuring tunability of circuit links. Tunability of each parameter is quantified by measuring the fold change in trigger time resulting from scaling the parameter by a factor of 10.

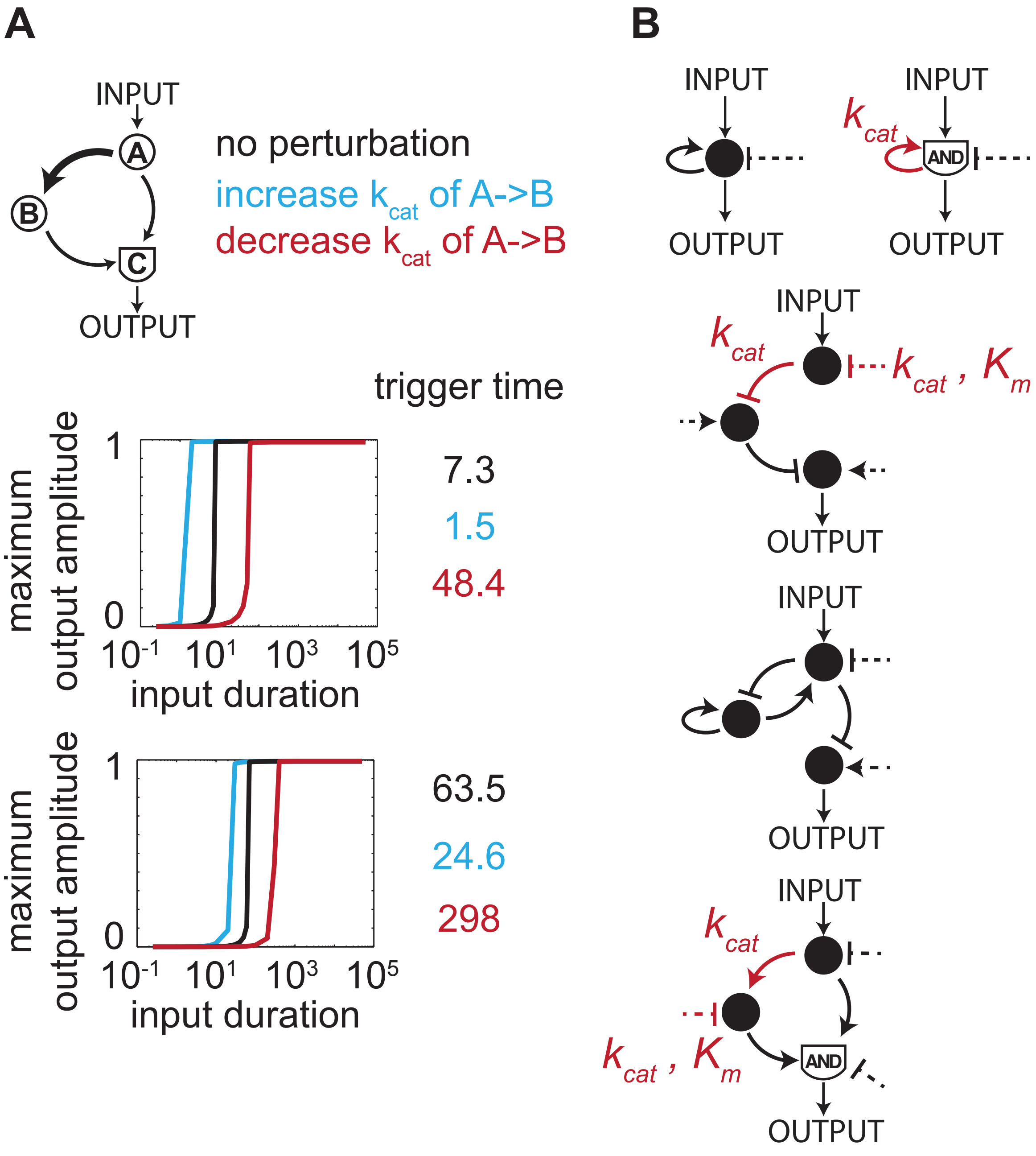


Figure 4.23. Tunability of prototype architectures.
A. Example shifting of temporal dose response curves when one parameter is tuned.
B. Tunable parameters of 5 prototypical architectures are shown in red.

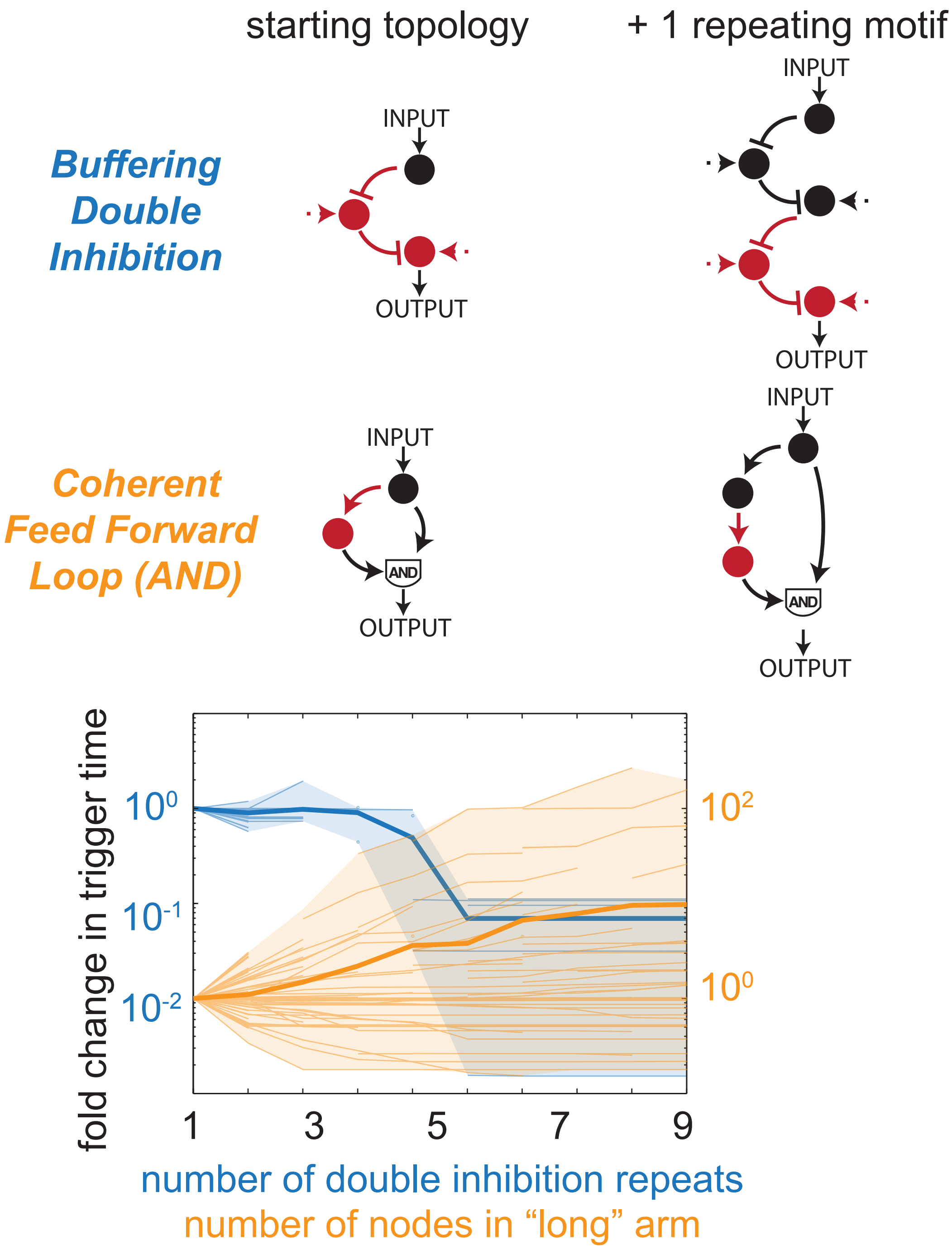


Figure 4.24. Chaining motifs in buffering double inhibition circuits shortens trigger time and destroys kinetic filtering behavior. In feed forward loop circuits, adding additional nodes in one arm can lengthen trigger time.

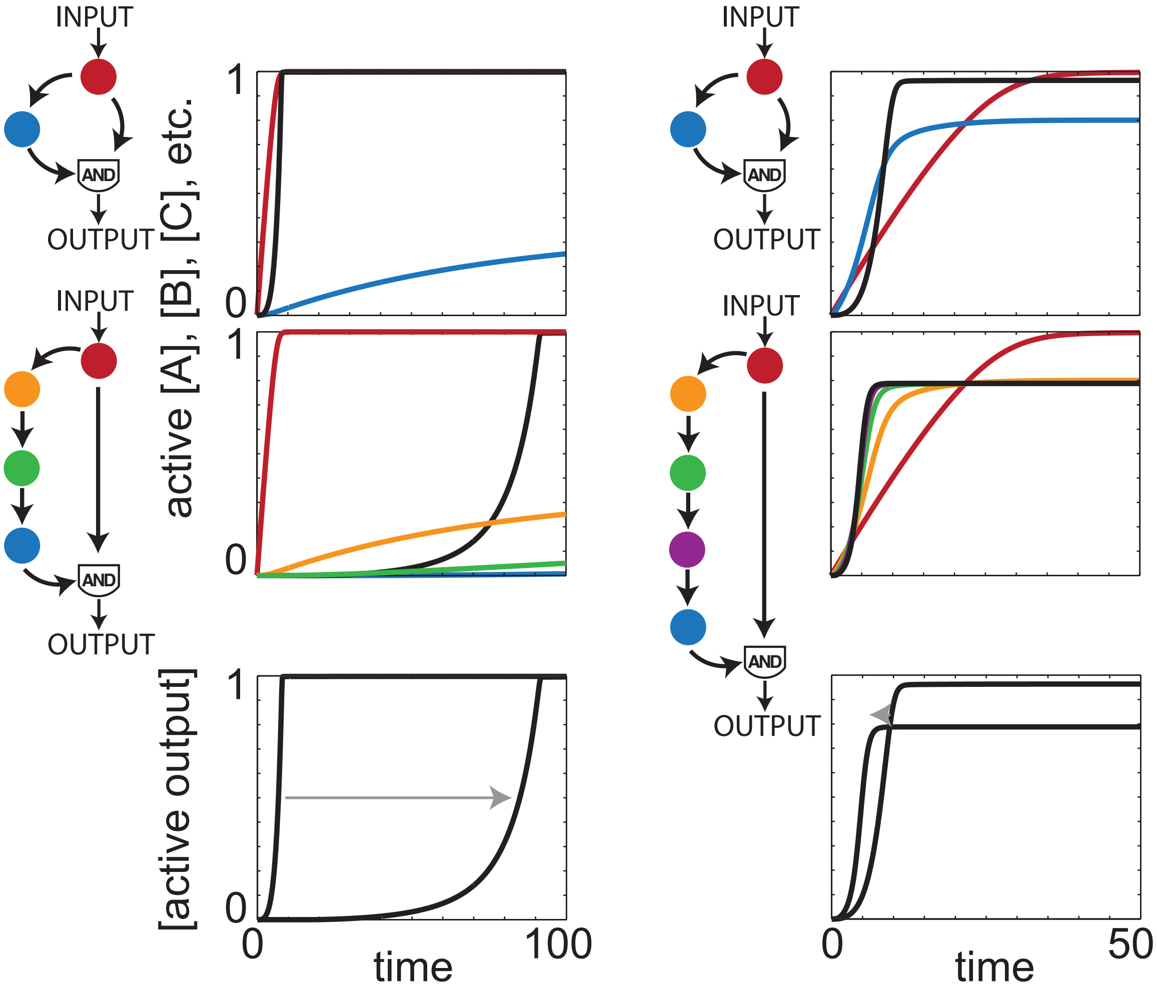


Figure 4.25. Representative timecourses of feed forward loop kinetic filters where adding additional nodes to one arm lengthens (left) or shortens (right) trigger time.

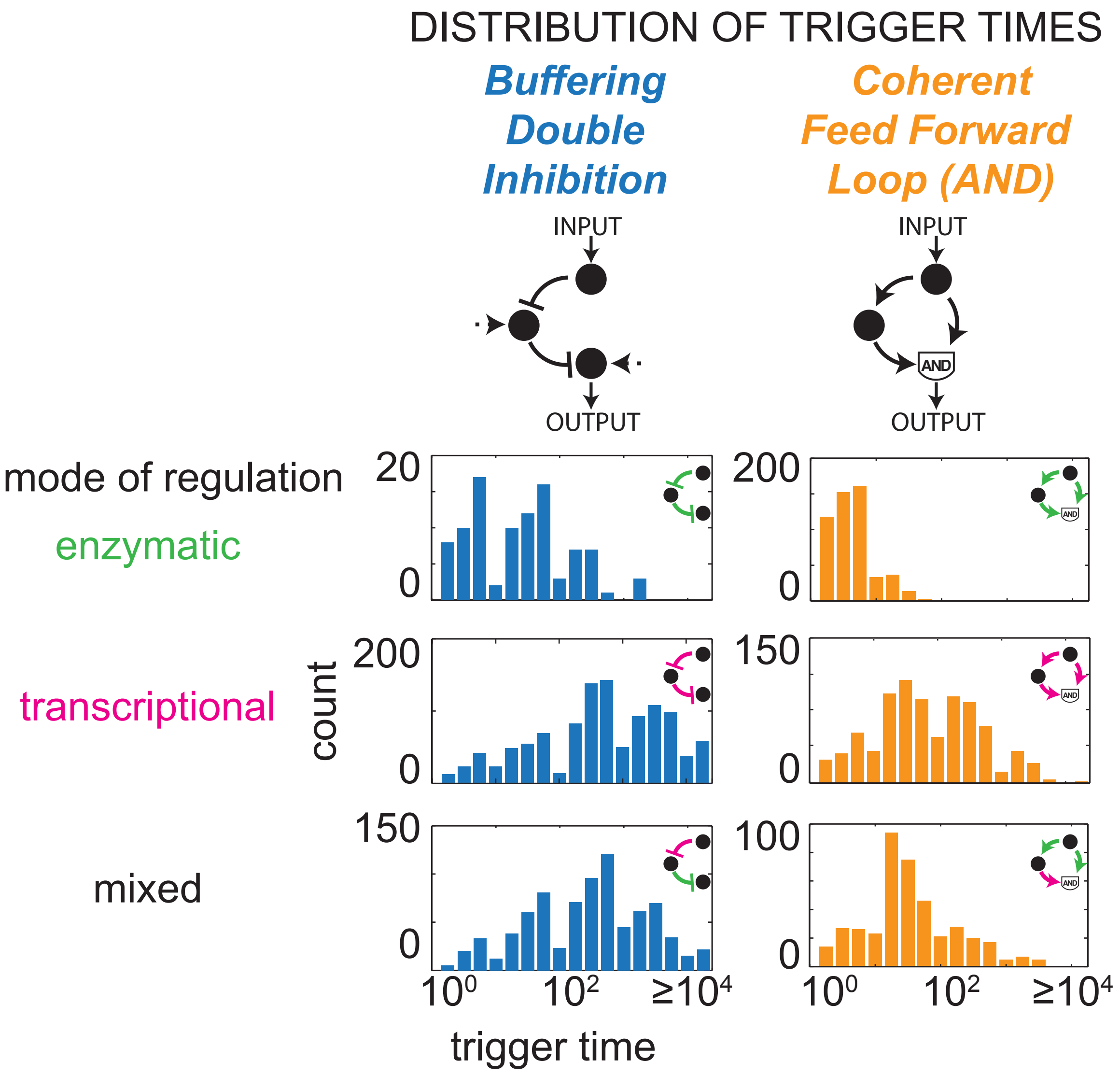


Figure 4.26. Transcriptional and mixed regulation circuits can have longer trigger times.

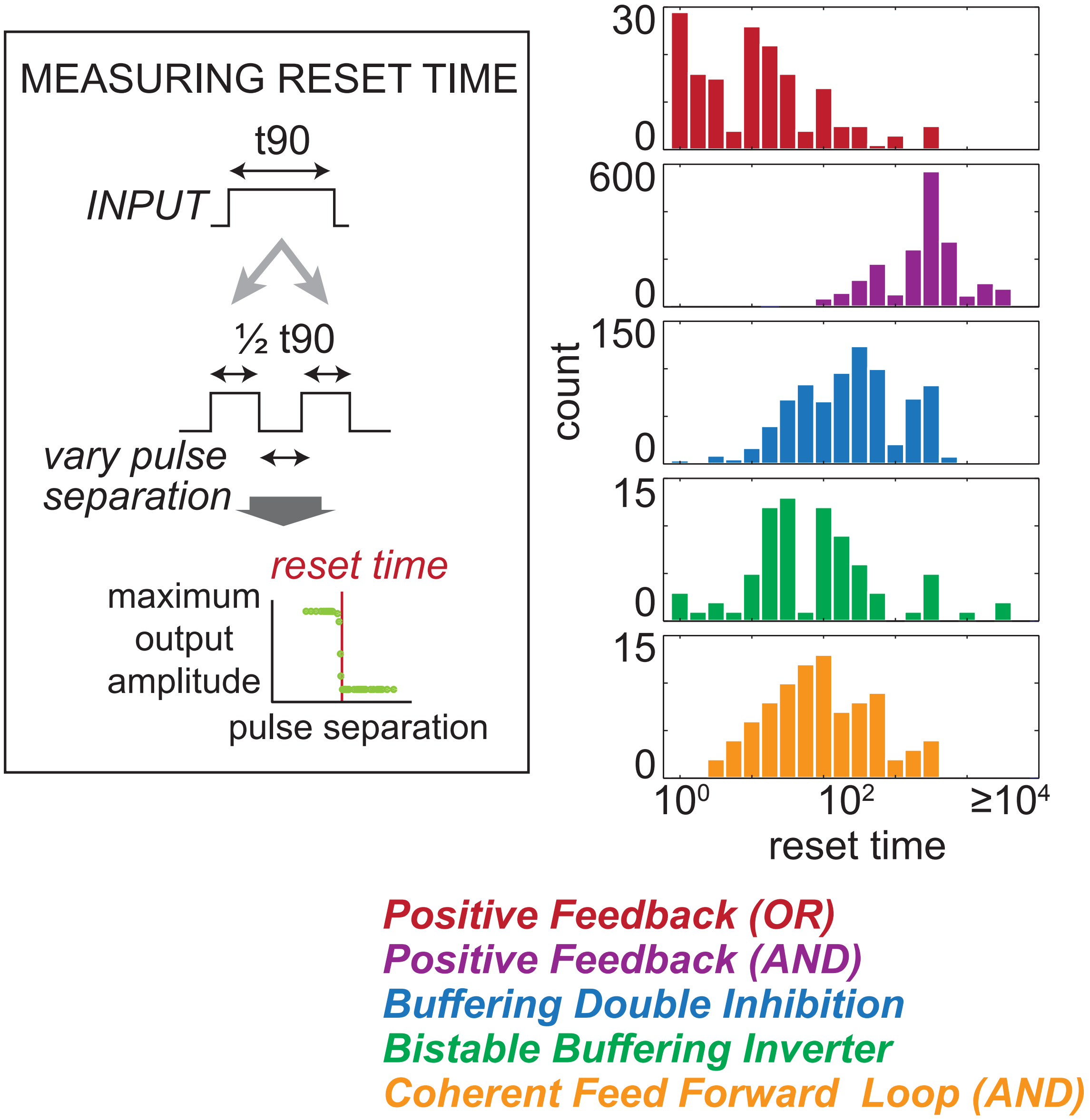


Figure 4.27. Reset times of minimal kinetic filters.

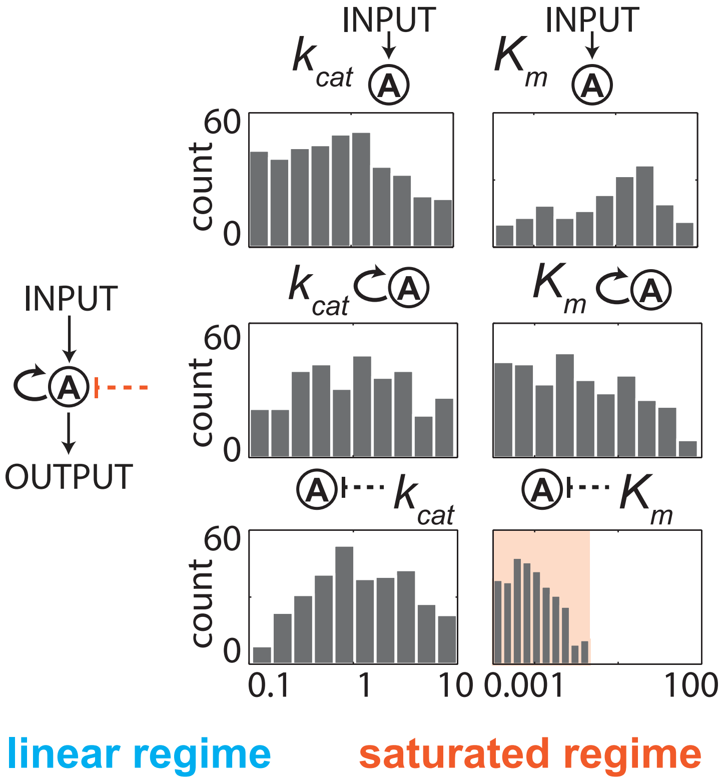


Figure 4.28. Parameter regime restrictions on positive feedback OR circuits (323 parameter sets).

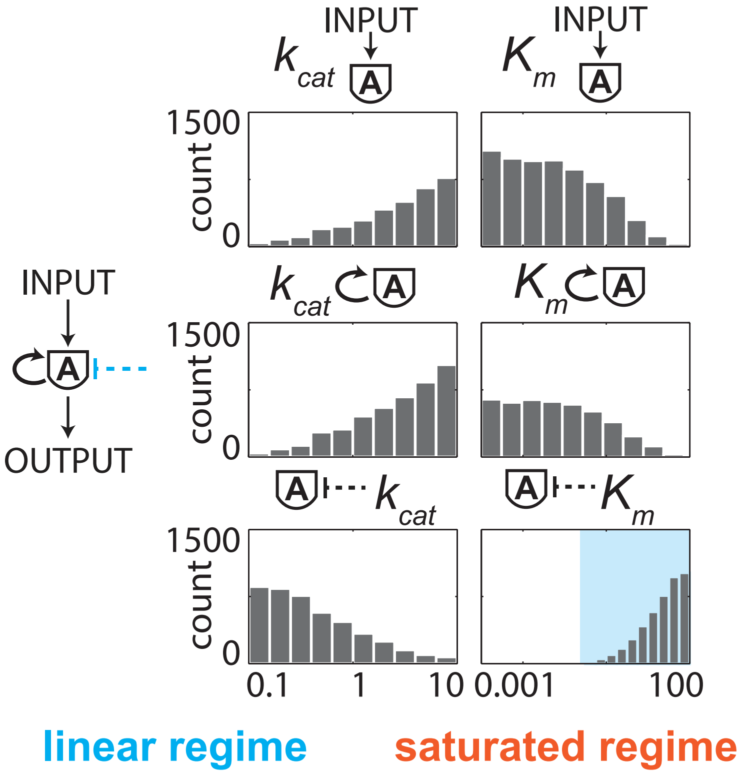


Figure 4.29. Parameter regime restrictions on positive feedback AND circuits (4391 parameter sets).

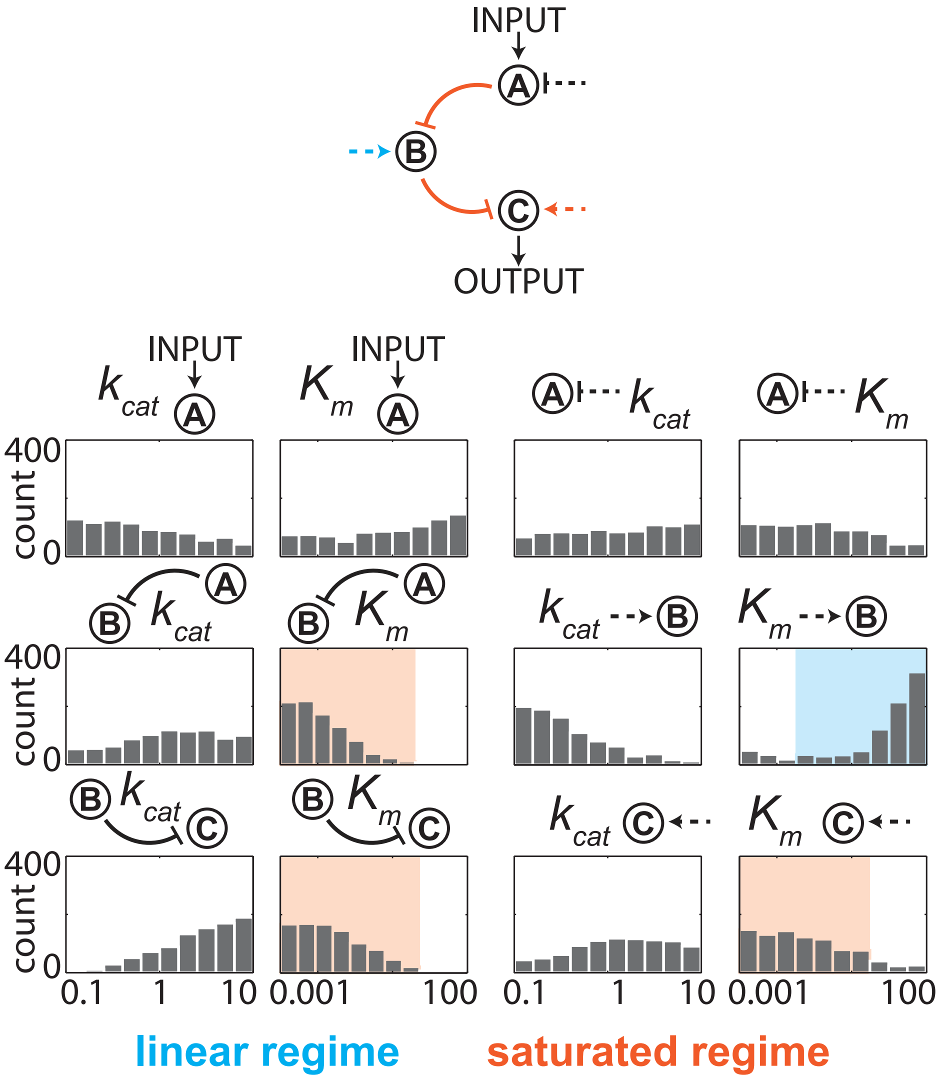


Figure 4.30. Parameter regime restrictions on buffering double inhibition circuits (907 parameter sets).

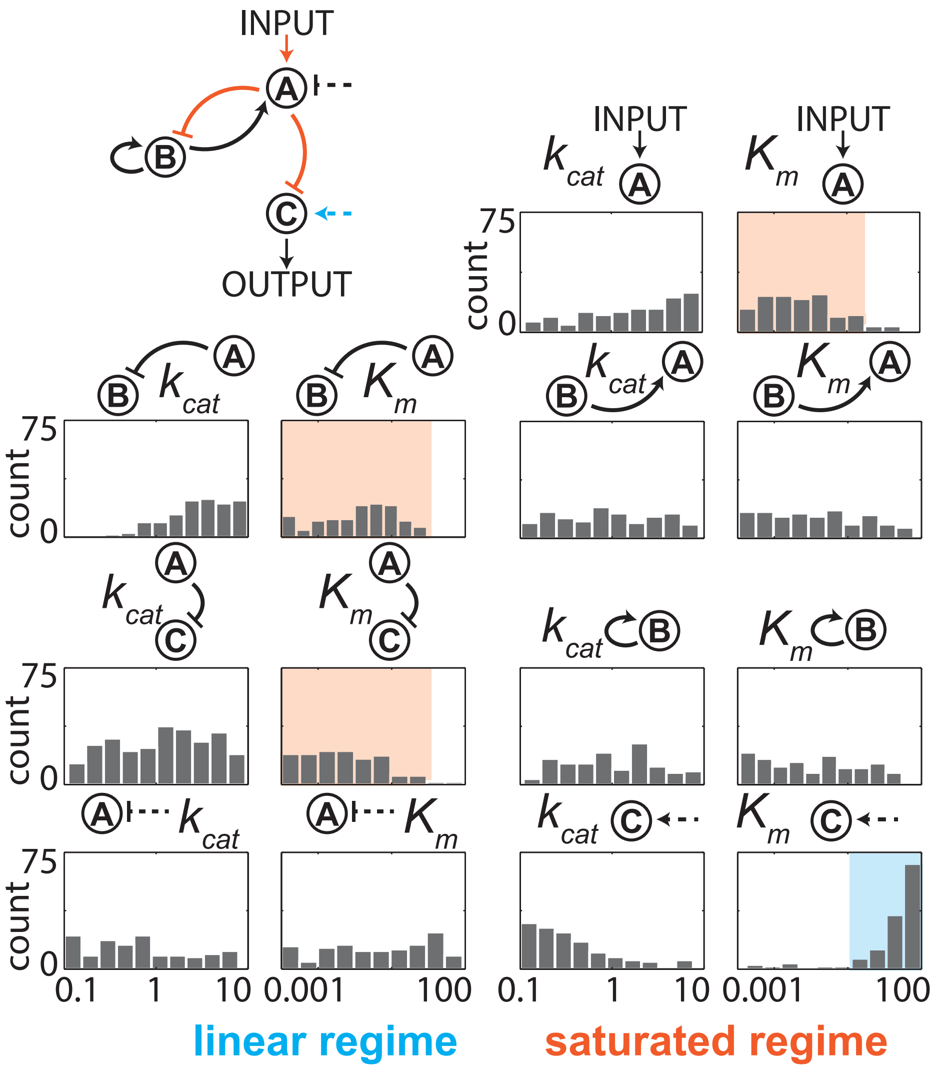


Figure 4.31. Parameter regime restrictions on bistable buffering inverter circuits (347 parameter sets).

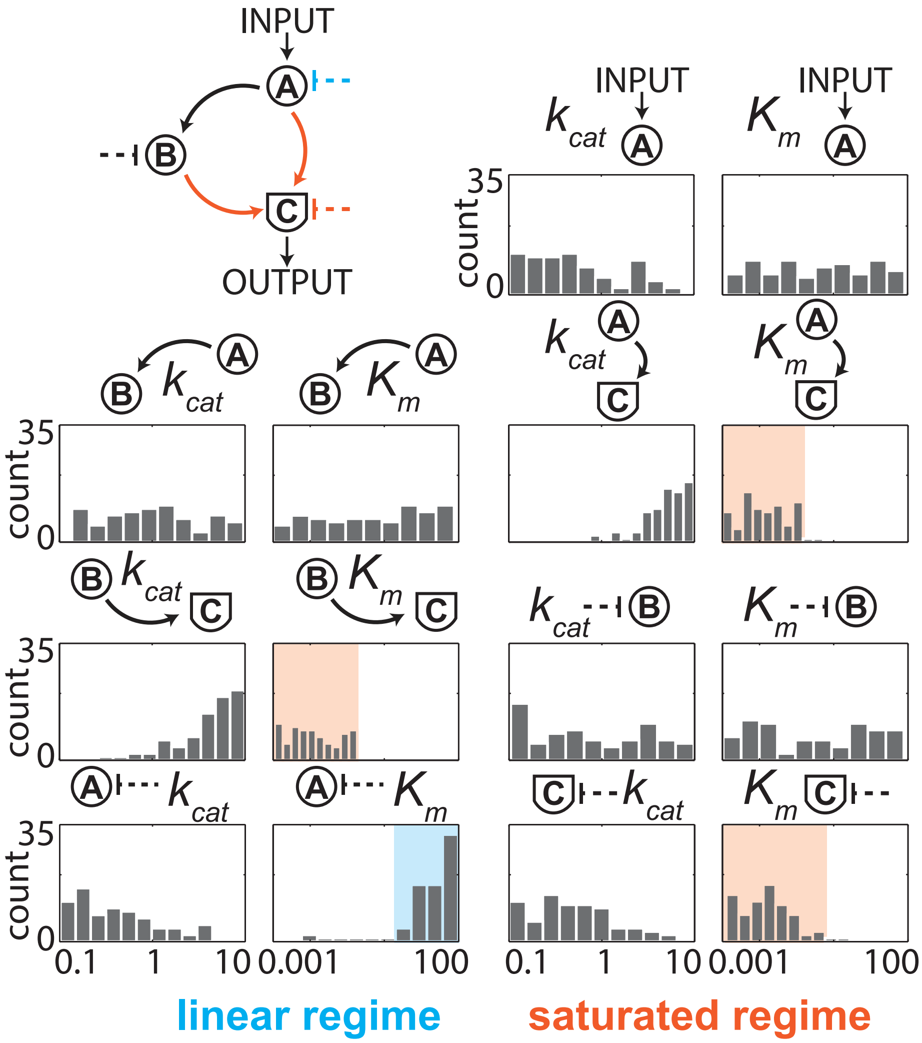


Figure 4.32. Parameter regime restrictions on coherent feed forward AND circuits (518 parameter sets).

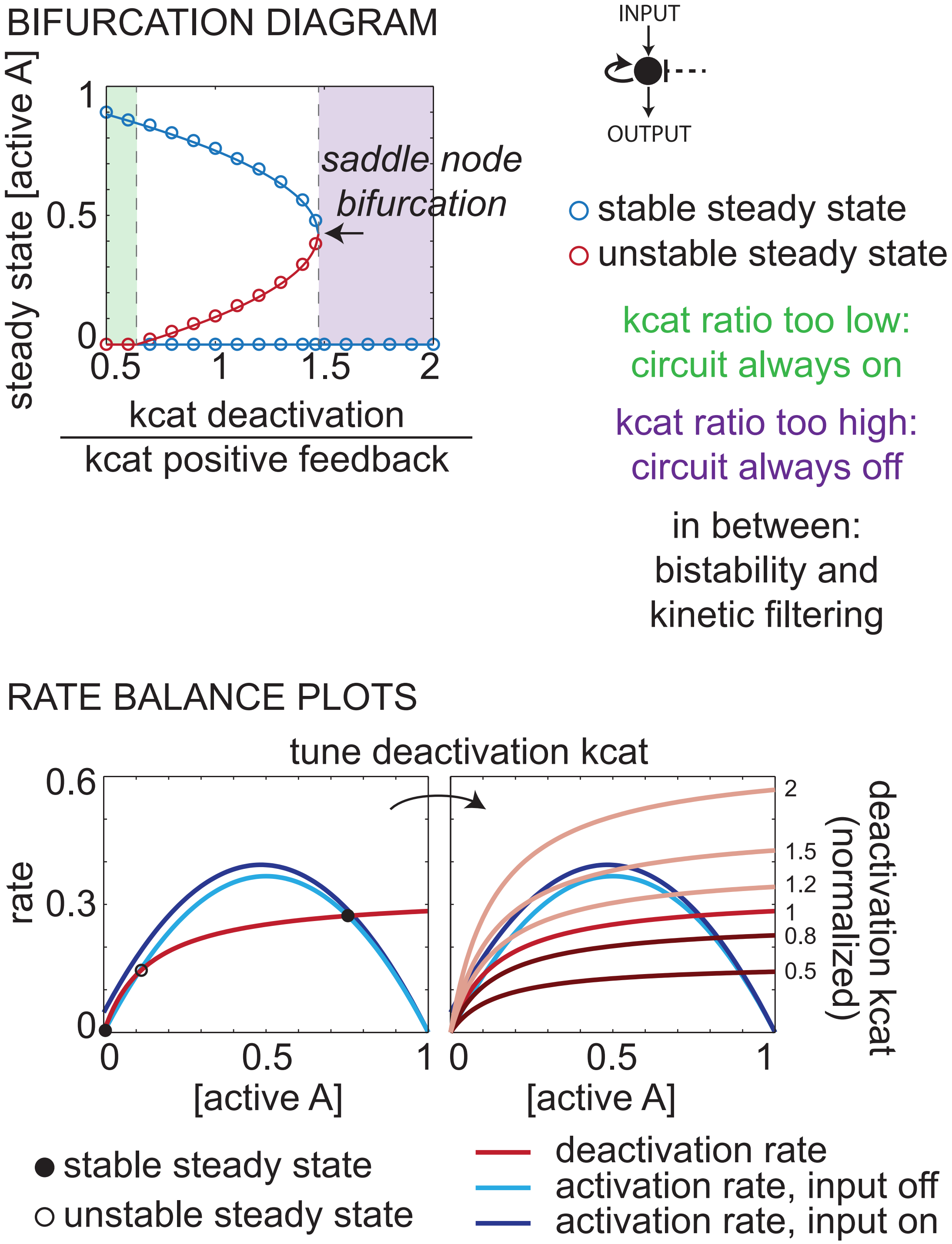
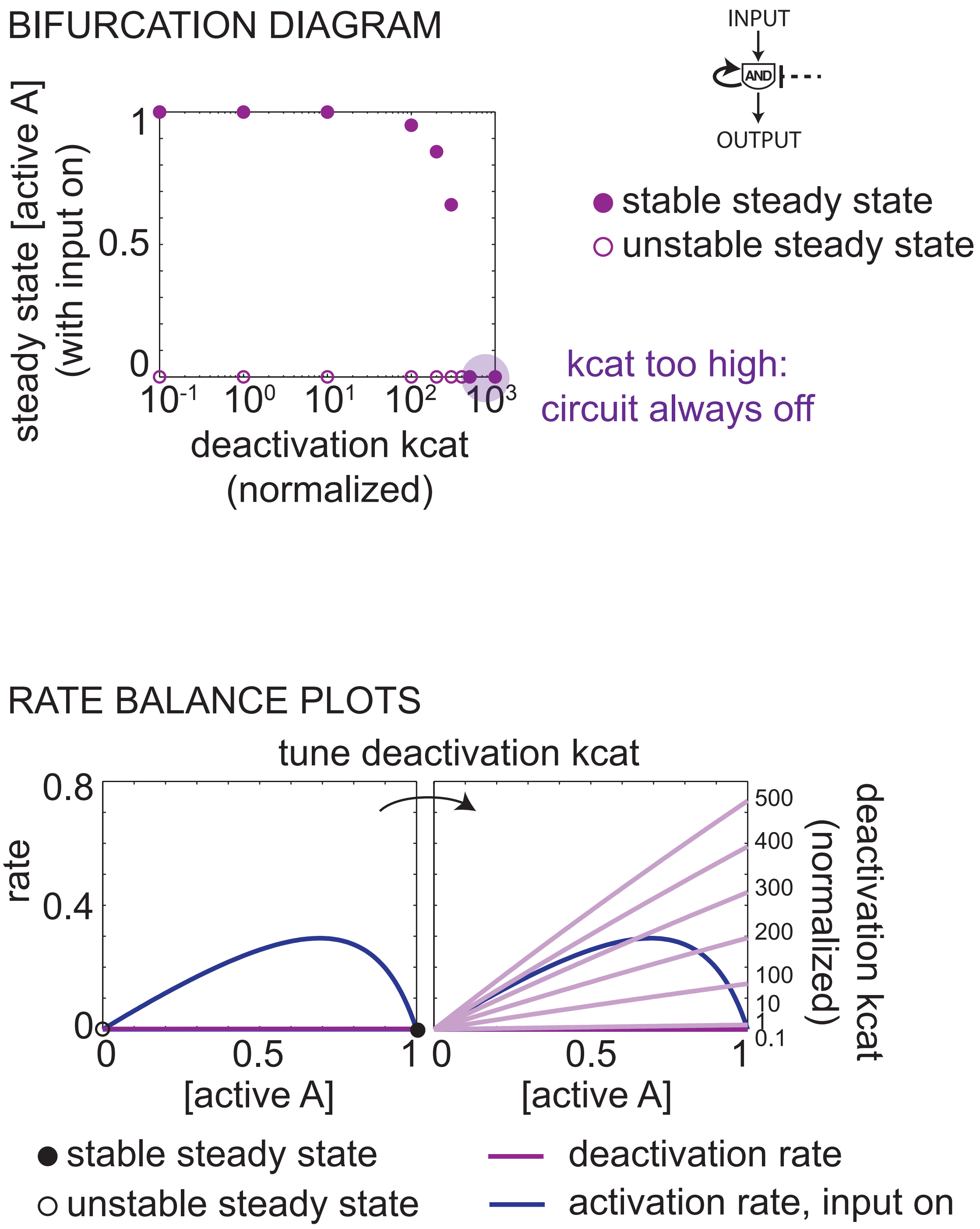


Figure 4.33. Constraints on kcat's of positive feedback OR circuits.



BIFURCATION DIAGRAM

RATE BALANCE PLOTS

Figure 4.34. Constraints on kcat's of positive feedback AND circuits.

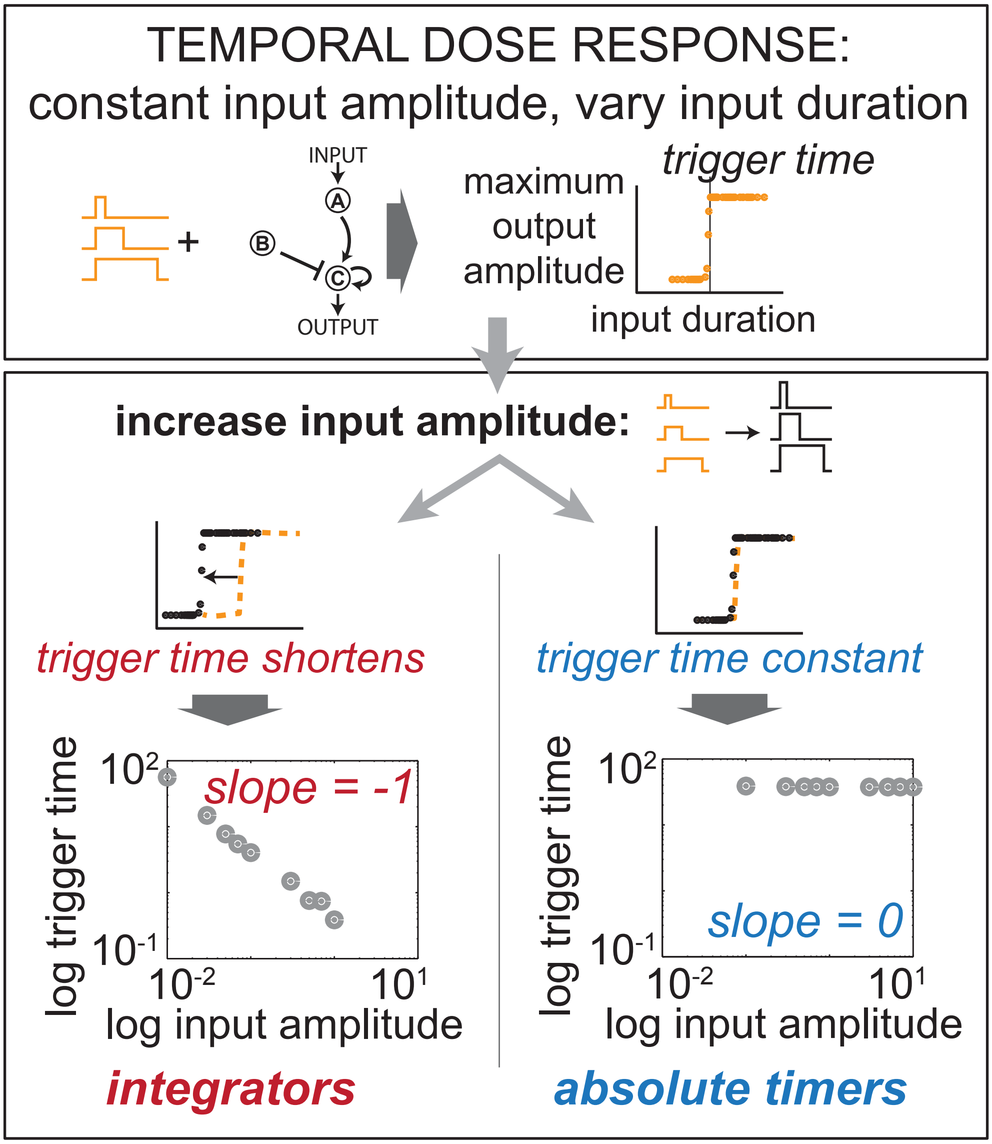


Figure 4.35. Identifying integrators and absolute timers.

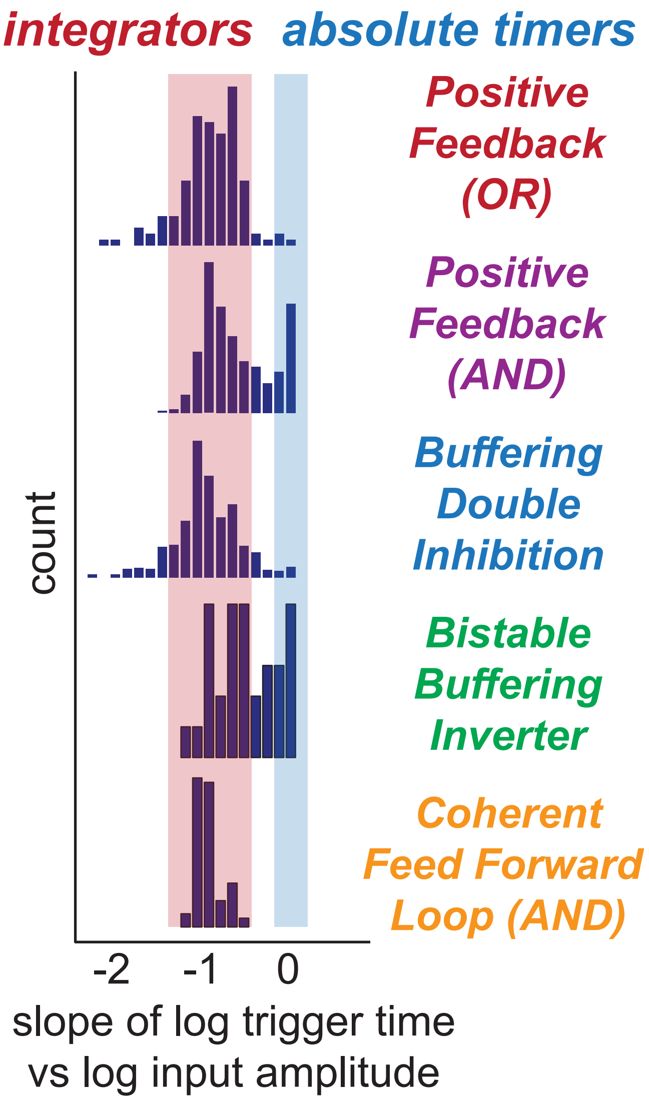


Figure 4.36. Distribution of integrators and absolute timers in minimal kinetic filtering circuits.

Chapter 5

DISCUSSION

OVERVIEW

The five types of kinetic filtering circuits identified and characterized in the previous two chapters provide an array of design options for forward engineering of kinetic filtering behavior. Biological examples of kinetic filters include the coherent feed forward loop with AND logic as well as many examples of buffering double inhibition circuits that are involved in circuits where timing is important. Future directions for both experimental and computational efforts are described.

5.1 TWO STRATEGIES FOR KINETIC FILTERING

We found that kinetic filtering signaling circuits of three or fewer nodes can be classified into five types that differ in both architecture and phenotype (Figure 3.13). Our enumerative approach allowed us to sample all possible kinetic filters of 1, 2, or 3 nodes in a consistent and unbiased manner and compare circuit performances. We recovered the coherent feed forward loop with AND logic, the only known kinetic filter, as one of our five types.

Four of the five kinetic filtering architectures use similar mechanisms to perform kinetic filtering. In the positive feedback OR and AND, buffering double inhibition, and bistable buffering inverter circuits, an input-activated regulation must overcome a constitutive or pre-existing regulation before output turns on (Figure 5.1). The time it takes for the input-activated regulation to “win” sets the trigger time and hence the clock of the circuit.

In contrast, coherent feed forward loop circuits use a fast arm/slow arm mechanism to distinguish between short and long inputs, where the input signal must reach the output node through both arms in order for the output to turn on, but signal takes longer to travel through the slower arm; as discussed in Chapter 4, the slower arm may be either arm of the coherent feed forward loop. Slowing down parameters on the longer side of the circuit increases trigger time.

These five architecture types are not the only simple circuits that are kinetic filters. A simple cascade of two nodes can satisfy temporal dose response steepness (temporal ultrasensitivity score) ≥ 0.5 for a small number of parameter sets, but the trigger time for these circuits is always very low: less than 1s. In contrast, topologies in the five architecture types we found can have trigger times spanning up to 5 orders of magnitude in time and achieve trigger times 5 orders of magnitude longer than a 2- or 3-node cascade. Of the 5 kinetic filtering architecture families, the bistable buffering inverter and coherent feed forward loop tend to have the shortest trigger time, while the buffering double inhibition circuit has both the longest trigger times and the largest range of accessible trigger time.

As shown in Chapter 4, trigger times of circuits where a regulator is turned off are longer than trigger times of circuits where a regulator is turned on. As long as the regulator that is being turned off begins in excess concentrations, the circuit will have long trigger time. Tuning the regulator concentration is an easy way to tune the circuit’s trigger time without changing k_{cat}’s and K_m’s, making circuits that use this strategy particularly attractive in terms of finer control over temporal behavior.

5.2 DESIGN TABLE OF KINETIC FILTERING CIRCUITS

The five classes of kinetic filters present a diverse array of behaviors (Figure 5.2; see Chapters 3 and 4 for more details). Positive feedback OR and bistable buffering inverter circuits, while structurally and mechanistically dissimilar, are both bistable; these circuits can only fire once, and by locking on, they conflate cell fate decisions with kinetic filtering behavior.

In contrast, circuits in the other three classes of kinetic filters are able to reset after firing, allowing them to be sensitive to new inputs. These reusable circuits require a separate downstream module that translates their amplitude into cell fate decisions. Note that positive feedback AND circuits require a long recovery period before being able to respond to new input without lingering effects of the previous input. Thus, the most flexible and modular kinetic filters are the buffering double inhibition and coherent feed forward loop AND topologies.

As discussed in Chapter 4, buffering double inhibition circuits show the greatest range of trigger times and are able to access much longer trigger times than the other classes of kinetic filters. In general, circuits where output is controlled by turning off an activator or deactivator have longer trigger times than circuits where output is controlled by turning on an activator or deactivator. This principle can likely be extended to designing kinetic filters with any regulatory mechanism: if input triggers reactions that deplete a regulatory component that begins in excess, the circuit's temporal dose response can show a long trigger time. Indeed, a rewired yeast mating pathway shows a marked delay in turning on of transcriptional activity when an inhibitor must first be titrated away[42]. Using genetic circuits, experimenters have constructed a mutual inhibition toggle switch that can act as a timer[43,44]. However, recall that kinetic filtering behavior also requires high steepness in the temporal dose response curve, and long trigger time and high steepness do not necessarily correlate positively.

Buffering double inhibition circuits are also the only type of kinetic filter capable of turning off output immediately following the input turning off: positive feedback AND circuits decay very slowly, coherent feed forward loop AND circuits show a lag in turning off, and the other two circuit types do not turn off. If a cell needs a kinetic filtering circuit to follow the shape of the input, the buffering double inhibition topology is the only choice when constrained to three or fewer circuit components.

Because positive feedback AND circuits decay slowly after input has turned off, these circuits may act as frequency detectors, responding to high but not low frequency input spikes. Further work is needed to test this hypothesis and delineate the requirements for frequency detection.

When constructing a synthetic kinetic filter, engineers may desire different properties for different situations. On the whole, however, the buffering double inhibition circuit is clearly the best design choice for a generic kinetic filter due to its modularity, reusability, and wide range of accessible trigger times.

5.3 BIOLOGICAL EXAMPLES OF KINETIC FILTERING CIRCUITS

In addition to being the only previously proposed kinetic filtering circuit, the coherent feed forward loop is also the only architecture experimentally demonstrated to distinguish between transient and sustained inputs. In growth factor response in PC-12 cells, ERK induces expression of the transcription factor cFos, which also requires subsequent phosphorylation by ERK in order to be active (Figure 5.3). Consistent with our findings, the feed forward mechanism functions by integrating a slow step (transcription and translation of cFos) and a fast step (phosphorylation of cFos) with AND logic (cFos is only functional if both steps are performed).

None of the other types of kinetic filters have been shown to exhibit temporal ultrasensitivity in biology, but many biological systems thought to measure time contain at least one of the five types. For example, many circuits that regulate cell cycle transitions use double inhibition motifs (Figure 5.3), which we have shown can exhibit kinetic filtering behavior.

Our work may help answer why many biological signaling pathways use double inhibition rather than activation of an activator to implement what appears to be the same function. The ability to filter out spurious input is particularly valuable when the ultimate effect of the signaling pathway is drastic and irreversible, such as cell division or cell death. Double inhibition also allows the cell to navigate tricky timings such as waiting for completion of spindle assembly before simultaneously separating sister chromatids during anaphase.

5.4 FUTURE DIRECTIONS

5.4.1 Experimental

Now that we have several designs of kinetic filtering circuits in hand, the next step is to construct them *in vivo*. We suggest the buffering double inhibition circuit as the template due to its simplicity and easy tunability: over- or underexpressing the regulator node will change the circuit's trigger time. Although our findings are based on enzymatic simulations, we believe that implementing this circuit with other regulatory modes will likely also result in kinetic filtering behavior as long as the middle regulator node begins in excess conditions.

Direct measurement of temporal dose response curves has traditionally been challenging due to the difficulty of applying precise durations of input signals. Recent advances in optogenetics have made possible just this type of experiment, and we look forward to measurement of temporal ultrasensitivity of both biological and synthetic designed circuits.

5.4.2 Computational

There remain many areas of kinetic filtering to investigate computationally. How does mode of regulation affect circuit behavior? Biological circuits implementing coherent feed forward loops as kinetic filters use transcription as the slow arm and post-translational modification as the fast arm. We showed in Chapter 4 that these mixed regulation coherent feed forward loop circuits can have much longer trigger time than purely enzymatic coherent feed forward loop circuits, but it remains to be seen if and how different regulatory modes allow or disallow our five architecture families to show kinetic filtering behavior.

Our enzymatic simulations did not include the possibility of cooperative behavior other than through feedback loops. An interesting question is whether expanding our parameter space to include Hill coefficients would also expand the architectural space of circuits that are kinetic filters, even under our conditions of fewer than 3 nodes.

As previously described, the positive feedback AND circuit turns off very slowly after input turns off and may act as a frequency detector. Both frequency detection and delay circuits are behaviors closely related to kinetic filtering: positive feedback AND circuits are kinetic filters and likely also frequency detectors, and buffering double inhibitions circuits are kinetic filters that show a clear delay before turning on. Further work is needed to clearly examine the relationship between these three temporal behaviors and compare the circuits capable of each.

In this study, we have focused on the behavior and properties of only the minimal kinetic filtering circuits. Recall that our enumerative screen identified thousands of robust kinetic filters, all of which contained at least one of the minimal kinetic filtering circuits as a substructure. Because we wanted to uncover the core architectural elements capable of driving kinetic filtering, we have ignored all of the non-minimal kinetic filters. However, Figure 3.2 and 3.7 show that non-minimal topologies can be much more robust than minimal topologies alone. What structural features contribute to greater robustness?

In Figure 5.4, we compare robustness between two minimal kinetic filters and two non-minimal kinetic filters that contain at least one of the minimal kinetic filters as substructures. Adding positive feedback to a minimal buffering double inhibition kinetic filter can increase robustness 6-fold, and combining multiple minimal kinetic filtering motifs within the same topology can increase robustness nearly 10-fold. Rigorous, systematic study of robustness is not possible with our relatively sparse parameter sampling, but identifying motifs and mechanisms by which additional robustness is conferred is of interest both in understanding extra layers of complexity in biological kinetic filters and in building robust synthetic circuits.

In addition to increasing circuit robustness, combining multiple minimal kinetic filtering motifs within the same topology can result in phenotypes not accessible to minimal kinetic filters alone. Combining positive feedback OR and buffering double inhibition motifs, for example, results in circuits that show the characteristic delay in turning on observed in buffering double inhibition circuits and are bistable for very long inputs like positive feedback OR circuits (Figure 5.5). Additional work can extend our phenotypic analysis covered in Chapter 3 to all robust kinetic filters of 1, 2, or 3 nodes, and we may find even more phenotypes of interest.

While we are confident that we have identified all robust kinetic filters of 1, 2, or 3 nodes, it is possible that new architectures can perform kinetic filtering under conditions of more than 3 nodes. At this time, enumerating 4-node circuit space with reasonable parameter sampling is beyond our computational resources, but a genetic algorithm study would be capable of evolving kinetic filters of higher complexity that may have interesting architectural and phenotypic features.

Finally, it would be interesting to search the entire space of mapped biological circuits for our kinetic filtering motifs. The coherent feed forward loop has been shown to be overrepresented in genetic circuits[15]; we anticipate that the simple kinetic filtering motifs such as positive feedback and double inhibition are also likely to be very common.

This study is part of recent efforts to build a map between circuit structure and circuit function. We envision a “periodic table” of signaling circuits, classifying a coarse-grained space of circuits into regions capable of observed cell behavior such as perfect adaptation, oscillations, spatial polarization, and persistence detection. Such a table would be invaluable for forward engineering in synthetic biology. Kinetic filters would be powerful tools for synthetic biologists building complex cellular behavior, particularly behaviors involving temporal regulation and dynamics, and an understanding of not only the architecture but also the robustness and limitations of candidate network topologies is essential for future engineering efforts.

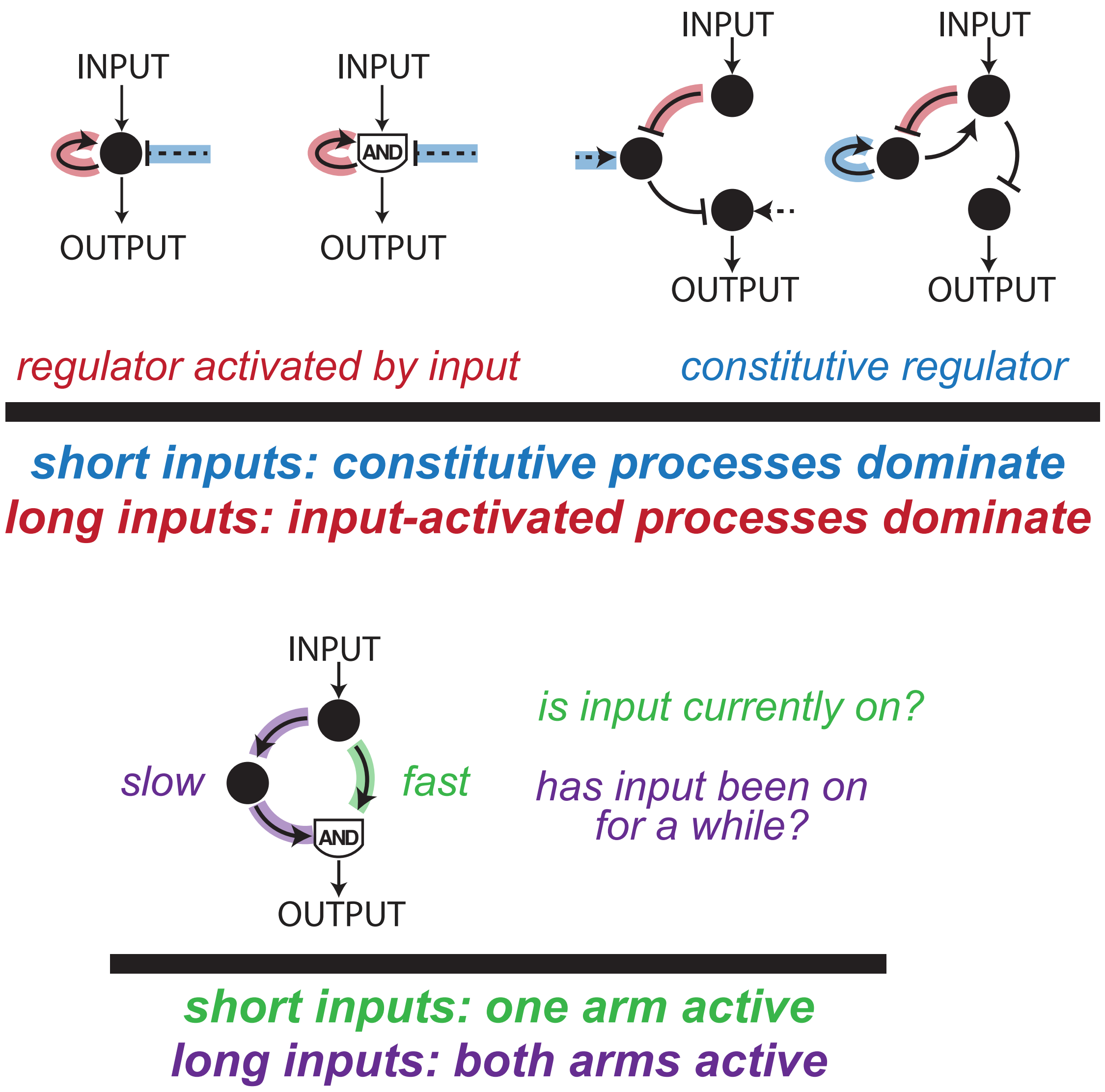

INPUT
AND
OUTPUT
regulator activated by input
constitutive regulator
short inputs: constitutive processes dominate
long inputs: input-activated processes dominate
slow
fast
is input currently on?
has input been on for a while?
short inputs: one arm active
long inputs: both arms active

Figure 5.1. Two strategies for kinetic filtering.

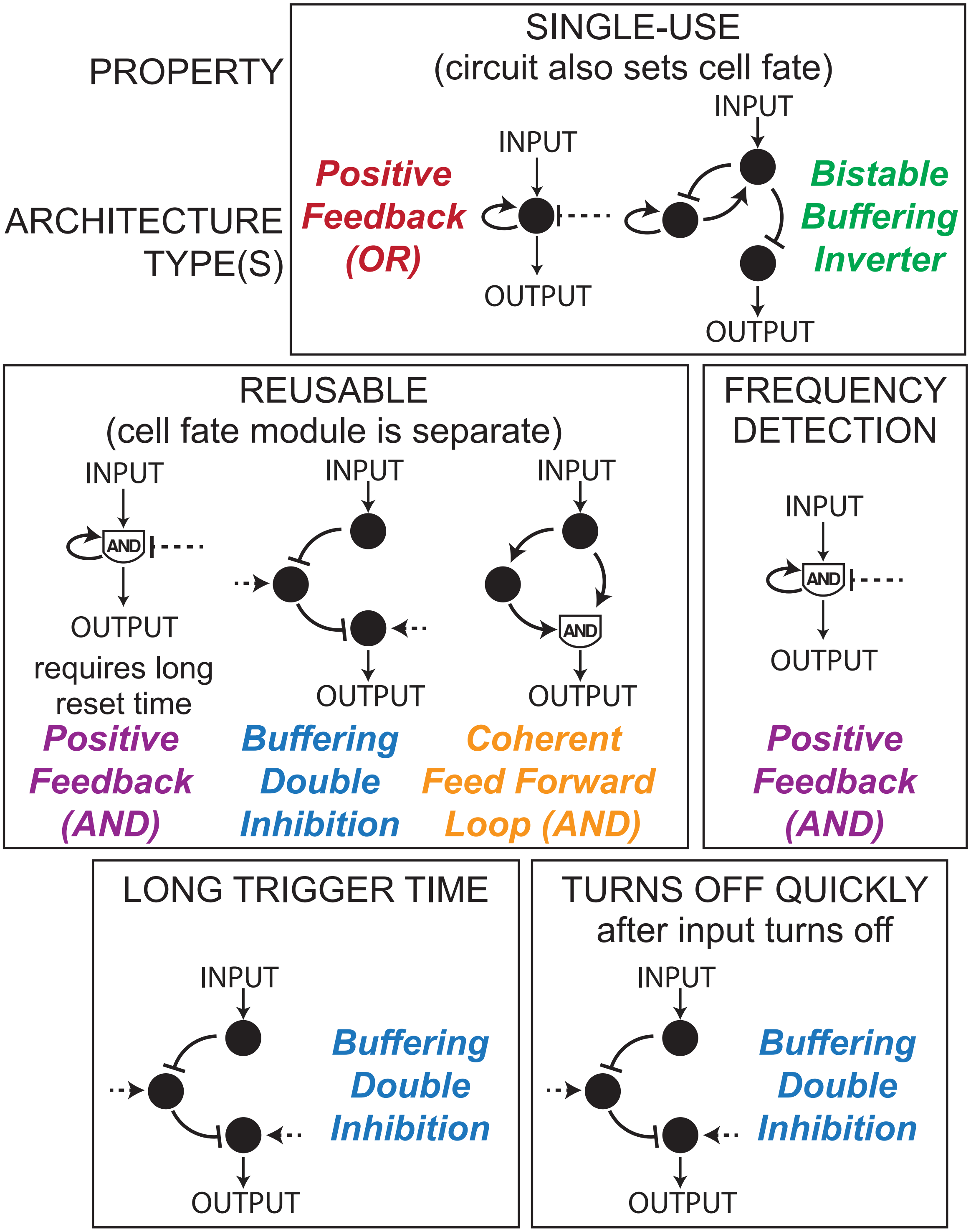


Figure 5.2. Design table for kinetic filters.

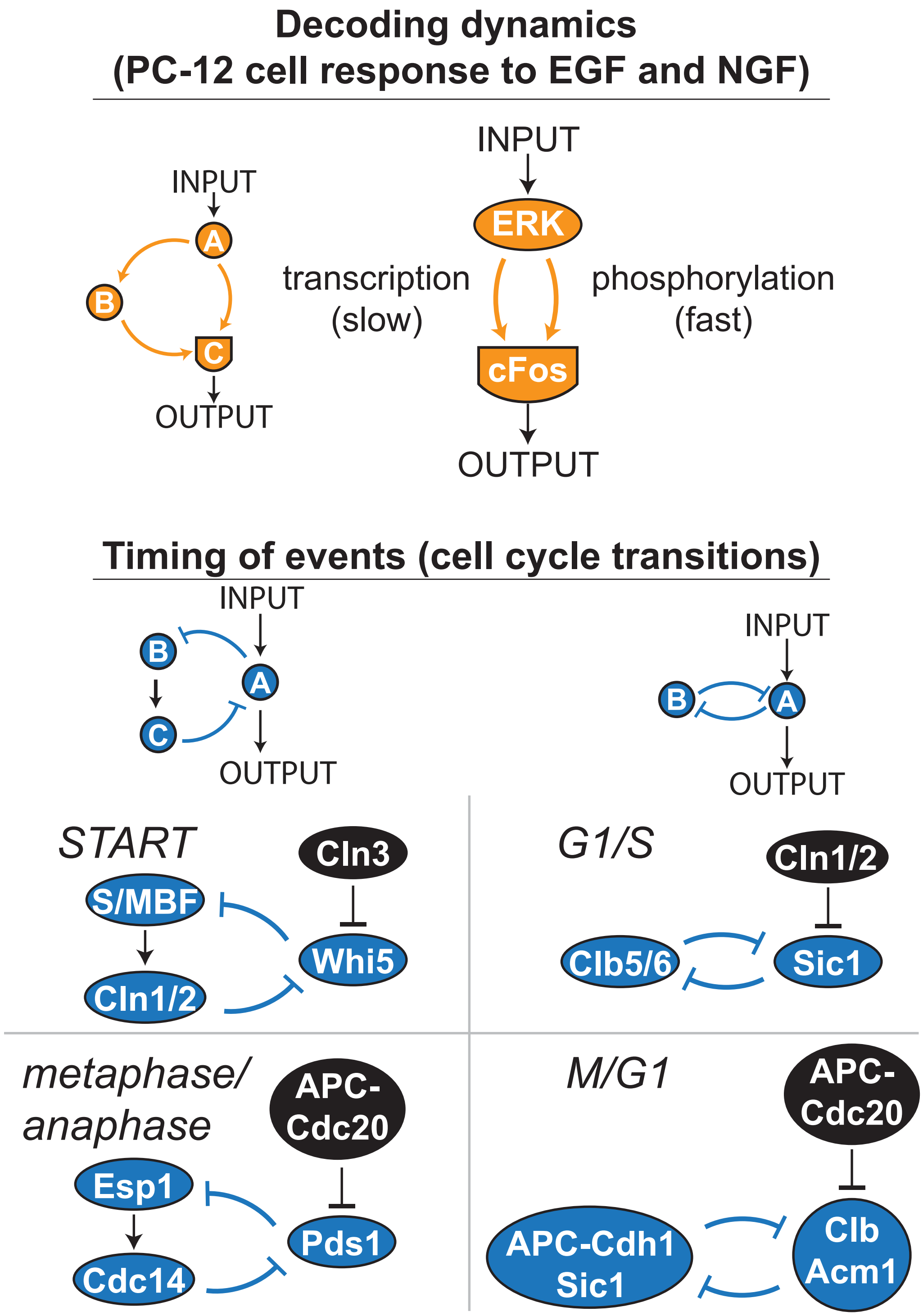


Figure 5.3. Biological examples of kinetic filtering circuits.

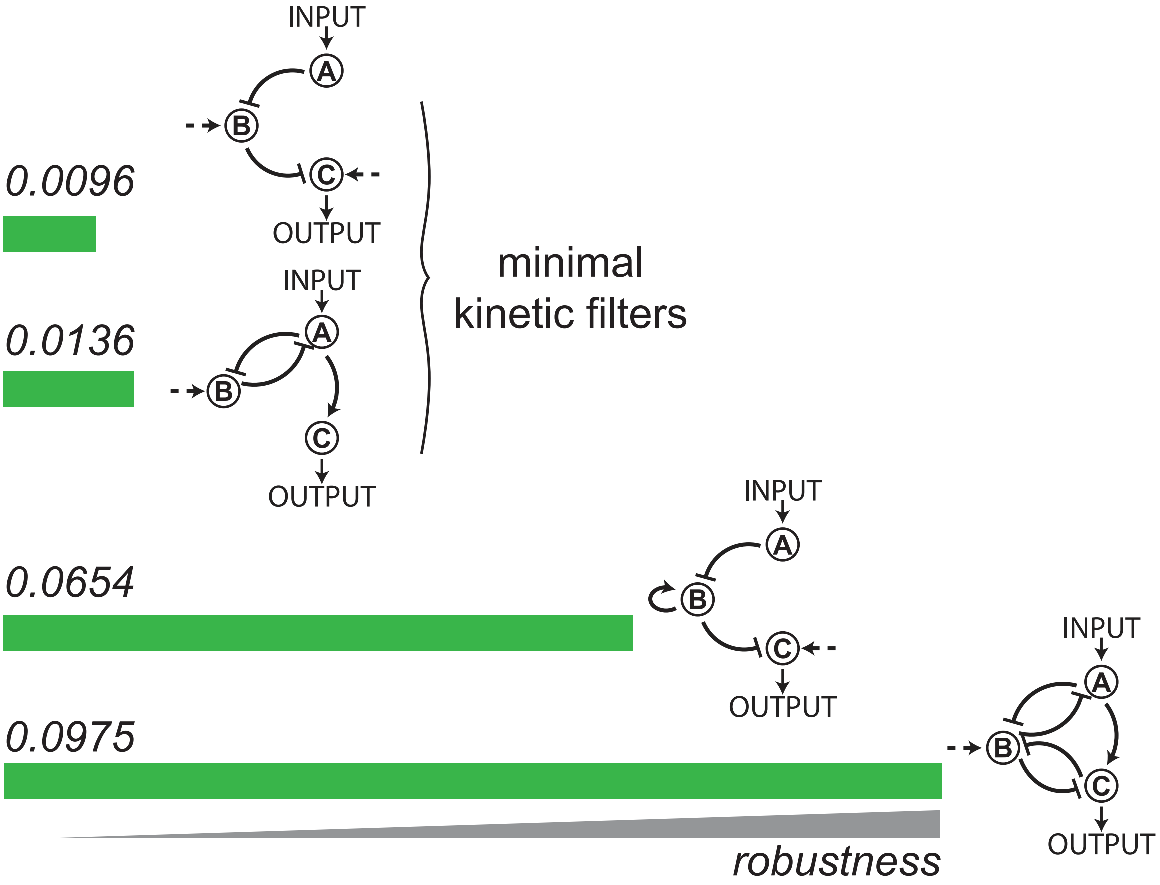


Figure 5.4. Adding positive feedback or combining minimal kinetic filtering topologies can lead to higher robustness in buffering double inhibition circuits.

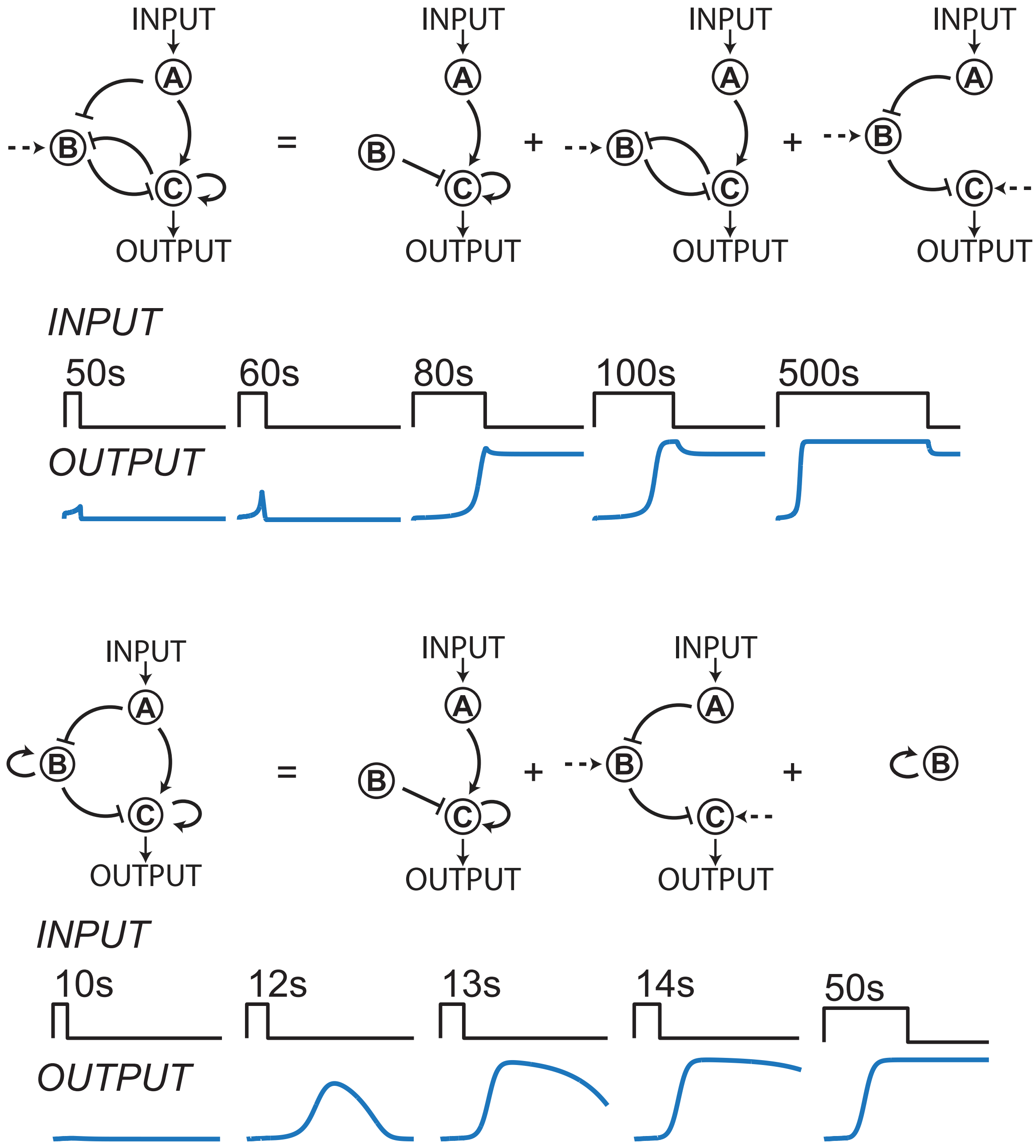


Figure 5.5. Topologies containing both positive feedback OR and buffering double inhibition motifs display both a delay in turning on and bistability for very long inputs.

REFERENCES

1. Kholodenko, B. N. Cell-signalling dynamics in time and space. *Nat Rev Mol Cell Biol* **7,** 165–176 (2006).

2. Ackers, G. K., Johnson, A. D. & Shea, M. A. Quantitative model for gene regulation by lambda phage repressor. *Proc. Natl. Acad. Sci. U.S.A.* **79,** 1129–1133 (1982).

3. Goldbeter, A. & Koshland, D. E. An amplified sensitivity arising from covalent modification in biological systems. *Proc. Natl. Acad. Sci. U.S.A.* **78,** 6840–6844 (1981).

4. Tyson, J. J., Chen, K. C. & Novak, B. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Current Opinion in Cell Biology* **15,** 221–231 (2003).

5. Paliwal, S. *et al.* MAPK-mediated bimodal gene expression and adaptive gradient sensing in yeast. *Nature* **446,** 46–51 (2007).

6. Colman-Lerner, A. *et al.* Regulated cell-to-cell variation in a cell-fate decision system. *Nature* **437,** 699–706 (2005).

7. Ferrell, J. E. & Machleder, E. M. The biochemical basis of an all-or-none cell fate switch in Xenopus oocytes. *Science* **280,** 895–898 (1998).

8. McCubrey, J. A. *et al.* Advances in targeting signal transduction pathways. *Oncotarget* **3,** 1505–1521 (2012).

9. Elowitz, M. B. & Leibler, S. A synthetic oscillatory network of transcriptional regulators. *Nature* **403,** 335–338 (2000).

10. Chau, A. H., Walter, J. M., Gerardin, J., Tang, C. & Lim, W. A. Designing synthetic regulatory networks capable of self-organizing cell polarization. *Cell* **151,** 320–332 (2012).

11. Lewis, J. Autoinhibition with Transcriptional Delay. *Current Biology* **13,** 1398–1408 (2003).

12. Tsai, T. Y. C. *et al.* Robust, Tunable Biological Oscillations from Interlinked Positive and Negative Feedback Loops. *Science* **321,** 126–129 (2008).

13. Brandman, O. & Meyer, T. Feedback loops shape cellular signals in space and time. *Science* **322,** 390–395 (2008).

14. Hornung, G. & Barkai, N. Noise propagation and signaling sensitivity in biological networks: a role for positive feedback. *PLoS Comput Biol* **4,** e8 (2008).

15. Shen-Orr, S. S., Milo, R., Mangan, S. & Alon, U. Network motifs in the transcriptional regulation network of Escherichia coli. *Nat. Genet.* **31,** 64–68 (2002).

16. Shah, N. A. & Sarkar, C. A. Robust network topologies for generating switch-like cellular responses. *PLoS Comput Biol* **7,** e1002085 (2011).

17. Ma, W., Lai, L., Ouyang, Q. & Tang, C. Robustness and modular design of the Drosophila segment polarity network. *Mol Syst Biol* **2,** (2006).

18. Ma, W., Trusina, A., Samad, El, H., Lim, W. A. & Tang, C. Defining Network Topologies that Can Achieve Biochemical Adaptation. *Cell* **138,** 760–773 (2009).

19. François, P. & Hakim, V. Design of genetic networks with specified functions by evolution in silico. *Proc. Natl. Acad. Sci. U.S.A.* **101,** 580–585 (2004).

20. François, P. & Siggia, E. D. A case study of evolutionary computation of biochemical adaptation. *Phys. Biol.* **5,** 026009 (2008).

21. François, P., Hakim, V. & Siggia, E. D. Deriving structure from evolution: metazoan segmentation. *Mol Syst Biol* **3,** (2007).

22. Hopfield, J. J. Kinetic proofreading: a new mechanism for reducing errors in biosynthetic processes requiring high specificity. *Proc. Natl. Acad. Sci. U.S.A.* **71,** 4135–4139 (1974).

23. Santos, S. D. M., Verveer, P. J. & Bastiaens, P. I. H. Growth factor-induced MAPK network topology shapes Erk response determining PC-12 cell fate. *Nat. Cell Biol.* **9,** 324–330 (2007).

24. Batchelor, E., Loewer, A., Mock, C. & Lahav, G. Stimulus-dependent dynamics of p53 in single cells. *Mol Syst Biol* **7,** 1–8 (2011).

25. Marshall, C. J. Specificity of receptor tyrosine kinase signaling: transient versus sustained extracellular signal-regulated kinase activation. *Cell* **80,** 179–185 (1995).

26. Murphy, L. O., Smith, S., Chen, R.-H., Fingar, D. C. & Blenis, J. Molecular interpretation of ERK signal duration by immediate early gene products. *Nat. Cell Biol.* (2002). doi:10.1038/ncb822

27. Horn, H. F. & Vousden, K. H. Coping with stress: multiple ways to activate p53.

Oncogene **26,** 1306–1316 (2007).

28. Lahav, G. *et al.* Dynamics of the p53-Mdm2 feedback loop in individual cells. *Nat. Genet.* **36,** 147–150 (2004).

29. Batchelor, E., Mock, C. S., Bhan, I., Loewer, A. & Lahav, G. Recurrent initiation: a mechanism for triggering p53 pulses in response to DNA damage. *Molecular Cell* **30,** 277–289 (2008).

30. Levine, J. H., Fontes, M. E., Dworkin, J. & Elowitz, M. B. Pulsed feedback defers cellular differentiation. *PLoS Biol* **10,** e1001252 (2012).

31. Samoilov, M., Arkin, A. & Ross, J. Signal Processing by Simple Chemical Systems. *J. Phys. Chem. A* **106,** 10205–10221 (2002).

32. Locasale, J. W. Signal duration and the time scale dependence of signal integration in biochemical pathways. *BMC Syst Biol* **2,** 108 (2008).

33. Mangan, S. & Alon, U. Structure and function of the feed-forward loop network motif. *Proc. Natl. Acad. Sci. U.S.A.* **100,** 11980–11985 (2003).

34. Mangan, S., Zaslaver, A. & Alon, U. The Coherent Feedforward Loop Serves as a Sign-sensitive Delay Element in Transcription Networks. *J. Mol. Biol.* **334,** 197–204 (2003).

35. Iman, R. L., Davenport, J. M. & Zeigler, D. K. *Latin Hypercube Sampling (program User's Guide)*. (1980).

36. Tzafriri, A. R. Michaelis-Menten kinetics at high enzyme concentrations. *Bull. Math. Biol.* **65,** 1111–1129 (2003).

37. Gomez-Uribe, C., Verghese, G. C. & Mirny, L. A. Operating regimes of signaling cycles: statics, dynamics, and noise filtering. *PLoS Comput Biol* **3,** e246 (2007).

38. Ciliberto, A., Capuani, F. & Tyson, J. J. Modeling networks of coupled enzymatic reactions using the total quasi-steady state approximation. *PLoS Comput Biol* **3,** e45 (2007).

39. Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. *Numerical Recipes in C++: The Art of Scientific Computing, 2nd Edition*. (Cambridge University Press, 2002).

40. Savageau, M. A. & Rosen, R. Biochemical systems analysis: a study of function and design in molecular biology. (1976).

41. Alon, U. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. (Chapman and Hall/CRC, 2006).

42. Bashor, C. J., Helman, N. C., Yan, S. & Lim, W. A. Using Engineered Scaffold Interactions to Reshape MAP Kinase Pathway Signaling Dynamics. *Science* **319,** 1539–1543 (2008).

43. Ellis, T., Wang, X. & Collins, J. J. Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat Biotechnol* **27,** 465–471 (2009).

44. Gardner, T. S., Cantor, C. R. & Collins, J. J. Construction of a genetic toggle switch in Escherichia coli. *Nature* **403,** 339–342 (2000).

Appendix A

SIMULATION CODE

This appendix contains 3 software programs to run simulations of signaling circuits and analyze their temporal dose response curves.

A.1 simulate.cc

```
/****************************************************************
 * simulate.cc
 * Jaline Gerardin 2010
 *
 * Modeling enzymatic circuits, 3 nodes, OR/AND, 1 pulse of input
 *
 * simulate.cc is given a number of command line arguments, simulates
 * a circuit's response to 1 pulse of input, and outputs to stdout in
 * one of 2 modes: metrics (no RECORD mode) and full timecourse
 * (RECORD mode).
 *
 * Command line arguments:
 *
 * [input duration] [basal input] [change in input]
 *
 * [node A logic] [node B logic] [node C logic] where 0 is OR and 1 is
 * AND
 *
 * 26 kcat's and Kms:
 * [kcat of input acting on node A] [Km of input acting on node A]
 * [kcat of A acting on A] [Km of A acting on A] [kcat of A acting on
 * B] [Km of A acting on B] [kcat of A acting on C] [Km of A acting on
 * C] [kcat of constitutive regulator of A][Km of constitutive
 * regulator of A]
 * + similar for node B and node C
 *
 * Note that regulation type (activator, inhibitor, absent) is coded
 * in kcat: positive kcat = activator, negative kcat = inhibitor, and
 * 0 kcat = absent.
 *
 * Optional arguments:

 * Initial concentrations of nodes A, B, and C. If none are given,
 * simulate.cc will initialize the circuit from initial concentration
 * of 0.1 for each node, then either output the initialized values
 * (metrics mode) or continue onward to apply input (fill timecourse
 * mode). If the circuit cannot reach steady state during
 * initialization, an error of "-1" is outputed. If initial
 * concentrations are given, simulate.cc will apply input without
 * pre-initializing and simulate from there.
 *
 * ODEs are defined in the getplus() function and integrated using a
 * fifth-order Runge-Kutta method with adaptive stepsize control.
 *
 * In metrics mode, simulate.cc will calculate and output metrics
 * immediately after the input pulse as well as after the circuit has
 * reached steady state once input has turned off. Metrics include
 * initial output value, final output value, output amplitude, and
 * other metrics. Normally simulate.cc is set to output an error
 * message of "-1" if the output node decreases in value in response
 * to input.
 ****************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <algorithm>
#include <limits>
#include <vector>
#include <iostream>
using namespace std;

/*** CONSTANTS AND PARAMETERS ***/
// mode toggler: uncomment to output timecourse; comment to output
timecourse metrics
//#define RECORD

#define RECNODE 2 // output node
#define NUMNODES 3 // number of nodes
#define INIT 0.1 // initial concentrations
#define FCONC 0.1 // constitutive enzyme conc
#define TOTALNODE 1 // total concentration of each node

// parameters for determining if steady state has been reached
#define ZERO 0.000001 // below 1e-6 -> set to 0.
#define SSCHECKSTART 50
#define SSCHECKEVERY 200

// adaptive Runge-Kutta parameters (ODE integrator)
#define DELTAZ 0.000001
#define SAFETY 0.9
#define ERRCON 0.000189
#define HMIN 0.001
#define HMAX 0.1

// parameters for metric measurement
#define INTTIME 50000 // duration to integrate over for measuring
integration of output

// global parameters set in command line
```

```
double base_input; // basal input level
int Aand, Band, Cand; // logic of each node
double Ainit = INIT, Binit = INIT, Cinit = INIT; // initial value of
each node

// kcat's and Km's for 3-node circuit
double kIA, KIA;
double kAA, KAA, kBA, KBA, kCA, KCA, kFAA, KFAA;
double kAB, KAB, kBB, KBB, kCB, KCB, kFBB, KFBB;
double kAC, KAC, kBC, KBC, kCC, KCC, kFCC, KFCC;


/*** FUNCTION DECLARATIONS ***/
// initializations
void start(vector<double> &ks, vector<double> &Ks, vector<int>
&andtrack);

// Runge-Kutta ODE solver
void step_rkck(double h1, vector<double> &innodes, vector<double>
&outnodes, vector<double> &yerr,
           vector<double> &Ks, vector<double> &ks, vector<int>
&andtrack);
double step_rkas(double h1, double &h2, vector<double> &nodes,
vector<double> &Ks,
       vector<double> &ks, vector<int> &andtrack);
double mymax(vector<double> &x);

// ODEs defined
void getplus(vector<double> &plus, vector<double> &nodes,
vector<double> &Ks,
         vector<double> &ks, vector<int> &andtrack);

// calculate timecourse metrics
int get_metrics(vector<double> &timecourse, vector<double> &timevec,
double &init, double &final, double &peakheight,
      double &peaktime, double &halfup, double &halfdown, double
&sstime, double &integral, double &ten,
      double &ninety,  double &inttopeak, double &inttohalf);


/*** MAIN ***
 *
 * 1. Read in parameters from command line
 * 2. Initialize nodes to pre-input steady state
 * 3. Apply input and simulate response
 * 4. Remove input and simulate response until steady state is reached
 *
 */
int main(int argc, char** argv)
{

  /*** READ IN AND SET SIMULATION PARAMETERS ***/

  double input_duration, input_change;
  int initialization_duration = 1000000; // maximum allowable time for
```

```
initialization
  bool initialize = false;

  if(argc < 33) // if not enough arguments in command line
    {
      cout << "Syntax: ./FILENAME input_duration basal_input 
change_in_input [node logic x 3] [26 kcats and Kms]. You have "
       << argc << "\n";
      return 1;
    }

  int k = 1;
  input_duration = atof(argv[k++]);
  base_input = atof(argv[k++]);
  input_change = atof(argv[k++]);

  // node logic. 0 = OR, 1 = AND
  Aand = atoi(argv[k++]);
  Band = atoi(argv[k++]);
  Cand = atoi(argv[k++]);

  // kcat and Km of input acting on node A
  kIA = atof(argv[k++]);
  KIA = atof(argv[k++]);

  // regulations by node A
  kAA = atof(argv[k++]);
  KAA = atof(argv[k++]);
  kAB = atof(argv[k++]);
  KAB = atof(argv[k++]);
  kAC = atof(argv[k++]);
  KAC = atof(argv[k++]);
  // constitutive regulator of A
  kFAA = atof(argv[k++]);
  KFAA = atof(argv[k++]);

  // regulations by node B
  kBA = atof(argv[k++]);
  KBA = atof(argv[k++]);
  kBB = atof(argv[k++]);
  KBB = atof(argv[k++]);
  kBC = atof(argv[k++]);
  KBC = atof(argv[k++]);
  // constitutive regulator of B
  kFBB = atof(argv[k++]);
  KFBB = atof(argv[k++]);

  // regulations by node C
  kCA = atof(argv[k++]);
  KCA = atof(argv[k++]);
  kCB = atof(argv[k++]);
  KCB = atof(argv[k++]);
  kCC = atof(argv[k++]);
  KCC = atof(argv[k++]);
  // constitutive regulator of C
  kFCC = atof(argv[k++]);
  KFCC = atof(argv[k++]);

  // if initialized node concentrations are given, use those;
otherwise, simulation will initialize
  if(argc == 36)
    {
      Ainit = atof(argv[k++]);
      Binit = atof(argv[k++]);
      Cinit = atof(argv[k++]);
    }
  else
    initialize = true;

 // starting node concentrations, parameter values, node logic set
  vector<double> nodes(NUMNODES);
  nodes[0] = Ainit;
  nodes[1] = Binit;
  nodes[2] = Cinit;
  vector<double> ks(NUMNODES*(NUMNODES+1));
  vector<double> Ks(NUMNODES*(NUMNODES+1));
  vector<int> andtrack(NUMNODES);
  start(ks, Ks, andtrack);

  // variables for tracking steady state
  vector<double> sstrack = nodes;
  bool foundss = false;

  // variables for managing ODE solver
  double h, hnext = 0.05; // adaptive stepsize starting size
  double time = 0;

  /*** IF NECESSARY, INITIALIZE NODE CONCENTRATIONS BY LETTING THE
SYSTEM COME TO STEADY STATE ***/
  if(initialize)
    {
      for(int i = 0; i < initialization_duration; i++)
     {
       // one step in ODE solver
       h = hnext;
       h = step_rkas(h, hnext, nodes, Ks, ks, andtrack);
       if(h < 0)
         break;
       time += h;

       // check whether steady state has been reached; if so, stop
initializing
       if(i%SSCHECKEVERY == 0 && time > SSCHECKSTART)
         if(fabs(nodes[0] - sstrack[0]) < ZERO && fabs(nodes[1] -
sstrack[1]) < ZERO && fabs(nodes[2] - sstrack[2]) < ZERO)
           {
      foundss = true;
      break;
           }
         else
```

```
          for(int j = 0; j < NUMNODES; j++)
      sstrack[j] = nodes[j];
     }

      // if timecourse is not being outputed, output initialized node
concentrations or output "-1" for error message
#ifndef RECORD
      if(foundss == false)
     {
       cout << -1 << "\t" << -1 << "\t" << -1 << endl;
       return 1;
     }
      else
     {
       cout << nodes[0] << " " << nodes[1] << " " << nodes[2] << endl;
       return 0;
     }
#endif
    }


  /*** APPLY INPUT AND SIMULATE RESPONSE ***/

  // variables for measuring timecourse metrics
  double initC, finalC, peakC, integral;
  double peaktime, halfup, halfdown, sstime;
  double ten, ninety;
  double inttopeak, inttohalf;
  vector<double> timecourse;
  vector<double> timevec;

  // input is changed
  base_input += input_change;

  // record starting output concentration and starting time
  time = 0;
  timecourse.push_back(nodes[RECNODE]);
  timevec.push_back(time);

  // ODE solver management
  h = 0.01;

  // simulate for duration input_duration
  while(time < input_duration)
    {
      // one step in ODE solver
      double myh = step_rkas(h, hnext, nodes, Ks, ks, andtrack);
      if(myh < 0) // solver error
     {
       cout << -1 << "\t" << -1 << "\t" << -1 << endl;
       return 1;
     }
      time += myh;
      h = hnext;
      // record output concentration and time
      timecourse.push_back(nodes[RECNODE]);
      timevec.push_back(time);

      // if timecourse is being outputed, do that
#ifdef RECORD
      cout << time << "\t" << nodes[0] << "\t" << nodes[1] << "\t" <<
nodes[2] << endl;
#endif

    }

  // calculate timecourse metrics
  int sign = get_metrics(timecourse, timevec, initC, finalC, peakC,
peaktime, halfup, halfdown, sstime, integral, ten, ninety, inttopeak,
inttohalf);
  // if metrics are being outputed, do that
#ifndef RECORD
  cout << initC << "\t" << finalC << "\t" << peakC << "\t" << peaktime
<< "\t" << halfup << "\t" << halfdown
       << "\t" << sstime << "\t" << integral << "\t" << ten << "\t" <<
ninety << "\t" << inttopeak << "\t"
       << inttohalf << "\t";
#endif


  /*** REMOVE INPUT AND SIMULATE RESPONSE ***/

  // steady state tracking variables
  foundss = false;
  sstrack = nodes;

  // return input to initial level
  base_input -= input_change;

  // ODE solver
  h = 0.01;

  // simulate until steady state is reached or 86400 seconds, whichever
comes first
  int i = 0;
  while(time < 86400)
    {
      i++;
      // one step in ODE solver
      double myh = step_rkas(h, hnext, nodes, Ks, ks, andtrack);
      if(myh < 0)
    {
      cout << -1 << "\t" << -1 << "\t" << -1 << endl;
      return 1;
    }
      time += myh;
      h = hnext;

      // record output concentration and time
      timecourse.push_back(nodes[RECNODE]);
      timevec.push_back(time);
      // if timecourse is being outputed, do that
#ifdef RECORD
      cout << time << "\t" << nodes[0] << "\t" << nodes[1] << "\t" <<
nodes[2] << endl;
#endif

      // check whether steady state has been reached
      if(i%SSCHECKEVERY == 0 && i > SSCHECKSTART)
    if(fabs(nodes[0] - sstrack[0]) < ZERO && fabs(nodes[1] -
sstrack[1]) < ZERO && fabs(nodes[2] - sstrack[2]) < ZERO
       && fabs(nodes[RECNODE] - timecourse[timecourse.size()-2]) <
ZERO)
      {
        foundss = true;
        break;
      }
    else
      for(int j = 0; j < NUMNODES; j++)
        sstrack[j] = nodes[j];
    }

  // calculate timecourse metrics
  sign = get_metrics(timecourse, timevec, initC, finalC, peakC,
peaktime, halfup, halfdown, sstime, integral, ten, ninety, inttopeak,
inttohalf);

  // if metrics are being outputed, do that
#ifndef RECORD
  if(foundss == false || sign < 0) // if steady state was not reached,
or if output turned off in response to input, output error
    cout << "-1\n";
  else
    cout << initC << "\t" << finalC << "\t" << peakC << "\t" <<
peaktime << "\t" << halfup << "\t" << halfdown
     << "\t" << sstime << "\t" << integral << "\t" << ten << "\t" <<
ninety << "\t" << inttopeak << "\t"
     << inttohalf << endl;
#endif

  return 0;
}


/*** FUNCTION DEFINITIONS ***/
/* find maximum element in vector */
double mymax(vector<double> &x)
{
  double max = x[0];
  for(unsigned int i = 1; i < x.size(); i++)
    if(x[i] > max)
      max = x[i];
  return max;
}
```

```
/* adaptive stepsize control for Runge-Kutta ODE solver */
double step_rkas(double h1, double &h2, vector<double> &nodes,
vector<double> &Ks,
       vector<double> &ks, vector<int> &andtrack)
{
  double delta1, h = h1, htemp;
  vector<double> outnodes(NUMNODES);
  vector<double> yerr(NUMNODES);

  time_t rkstart = time(NULL);

  for(;;)
    {
      time_t rknow = time(NULL);
      if (rknow - rkstart > 100)
    return -1;
      step_rkck(h, nodes, outnodes, yerr, Ks, ks, andtrack);
      delta1 = mymax(yerr);
      if(h < HMIN || delta1 < DELTAZ)
    break;
      if(delta1 <= DELTAZ)
    break;
      htemp = h*SAFETY*pow(fabs(DELTAZ/delta1), 0.2);
      if(htemp < 0.1*h)
    h = 0.1*h;
      else
    h = htemp;
    }
  if(delta1 > ERRCON)
    h2 = h*SAFETY*pow(fabs(DELTAZ/delta1), 0.25);
  else
    h2 = 5*h;
  if(h2 < HMIN)
    h2 = HMIN;
  if(h2 > HMAX)
    h2 = HMAX;

  for(int i = 0; i < NUMNODES; i++)
    {
      nodes[i] = outnodes[i];
      if(nodes[i] < ZERO)
    nodes[i] = 0;
      else if(nodes[i] > TOTALNODE)
    nodes[i] = TOTALNODE;
    }

  return h;

}
/* Fifth-order Runge-Kutta step */
void step_rkck(double h1, vector<double> &innodes, vector<double>
&outnodes, vector<double> &yerr,
           vector<double> &Ks, vector<double> &ks, vector<int>
&andtrack)
```

```
{
  vector<double> plus(NUMNODES);
  getplus(plus, innodes, Ks, ks, andtrack);
  vector<double> rkk1(NUMNODES);
  vector<double> rkk2(NUMNODES);
  vector<double> rkk3(NUMNODES);
  vector<double> rkk4(NUMNODES);
  vector<double> rkk5(NUMNODES);
  vector<double> rkk6(NUMNODES);
  vector<double> tempnodes(NUMNODES);

  for(int i = 0; i < NUMNODES; i++)
    {
      rkk1[i] = h1*plus[i]; // k1
      tempnodes[i] = innodes[i] + (1./5)*rkk1[i];
    }
  getplus(plus, tempnodes, Ks, ks, andtrack);
  for(int i = 0; i < NUMNODES; i++)
    {
      rkk2[i] = h1*plus[i]; // k2
      tempnodes[i] = innodes[i] + (3./40)*rkk1[i] + (9./40)*rkk2[i];
    }
  getplus(plus, tempnodes, Ks, ks, andtrack);
  for(int i = 0; i < NUMNODES; i++)
    {
      rkk3[i] = h1*plus[i]; // k3
      tempnodes[i] = innodes[i] + (3./10)*rkk1[i] + (-9./10)*rkk2[i] +
(6./5)*rkk3[i];
    }
  getplus(plus, tempnodes, Ks, ks, andtrack);
  for(int i = 0; i < NUMNODES; i++)
    {
      rkk4[i] = h1*plus[i]; // k4
      tempnodes[i] = innodes[i] + (-11./54)*rkk1[i] + (5./2)*rkk2[i] +
(-70./27)*rkk3[i] + (35./27)*rkk4[i];
    }
  getplus(plus, tempnodes, Ks, ks, andtrack);
  for(int i = 0; i < NUMNODES; i++)
    {
      rkk5[i] = h1*plus[i]; // k5
      tempnodes[i] = innodes[i] + (1631./55296)*rkk1[i] +
(175./512)*rkk2[i] + (575./13824)*rkk3[i] + (44275./110592)*rkk4[i] +
(253./4096)*rkk5[i];
    }
  getplus(plus, tempnodes, Ks, ks, andtrack);
  for(int i = 0; i < NUMNODES; i++)
    {
      rkk6[i] = h1*plus[i]; // k6
      outnodes[i] = innodes[i] + (37./378)*rkk1[i] + (250./621)*rkk3[i]
+ (125./594)*rkk4[i] + (512./1771)*rkk6[i];
      yerr[i] = (37./378 - 2825./27648)*rkk1[i] + (250./621 -
18575./48384)*rkk3[i] + (125./594 - 13575./55296)*rkk4[i]
    + (-277./14336)*rkk5[i] + (512./1771 - 1./4)*rkk6[i];
    }
}
```

```
/* differential equations defined */
void getplus(vector<double> &plus, vector<double> &nodes,
vector<double> &Ks,
     vector<double> &ks, vector<int> &andtrack)
{
 for(int i = 0; i < NUMNODES; i++)
  plus[i] = 0;
 vector<double> conc(NUMNODES*(NUMNODES+1));

 for(int i = 0; i < NUMNODES; i++)
  {
   for(int j = 0; j < (NUMNODES+1); j++)
  if(ks[i*(NUMNODES+1)+j] > 0)
   conc[i*(NUMNODES+1)+j] = TOTALNODE - nodes[i];
  else
   conc[i*(NUMNODES+1)+j] = nodes[i];

   if(andtrack[i] == 0) // node is under OR logic
  {
   for(int j = 0; j < (NUMNODES); j++)
    {
     if(fabs(ks[i*(NUMNODES+1)+j]) > 0)
   plus[i] +=
nodes[j]*ks[i*(NUMNODES+1)+j]*conc[i*(NUMNODES+1)+j]/(conc[i*(NUMNODES+
1)+j] + Ks[i*(NUMNODES+1)+j] + nodes[j]);
    }
  }
   else // node is under AND logic
  {
   if(i != 0) // node is not input node
    {
     double in1 = 0, in2 = 0, in3 = 0;
     int sign1 = 1, sign2 = 1, sign3 = 1;
     if(Ks[i*(NUMNODES+1)+0] > 0) // if there's regulation by A on
node i
   {
    in1 = ks[i*(NUMNODES+1)+0]*nodes[0]/(conc[i*(NUMNODES+1)+0] +
Ks[i*(NUMNODES+1)+0] + nodes[0]);
    if(ks[i*(NUMNODES+1)+0] < 0)
          sign1 = -1;
        }
     if(ks[i*(NUMNODES+1)+0] * ks[i*(NUMNODES+1)+1] > 0) // if reg
by A and by B have the same sign
   in1 *= ks[i*(NUMNODES+1)+1]*nodes[1]/(conc[i*(NUMNODES+1)+1] +
Ks[i*(NUMNODES+1)+1] + nodes[1]);
     else if(Ks[i*(NUMNODES+1)+1] > 0) // if reg by A and by B
have different sign
   {
    in2 = ks[i*(NUMNODES+1)+1]*nodes[1]/(conc[i*(NUMNODES+1)+1] +
Ks[i*(NUMNODES+1)+1] + nodes[1]);
    if(ks[i*(NUMNODES+1)+1] < 0)
          sign2 = -1;
        }
     if(ks[i*(NUMNODES+1)+0] * ks[i*(NUMNODES+1)+2] > 0) // if reg
```

```
by A and by C have the same sign
      in1 *= ks[i*(NUMNODES+1)+2]*nodes[2]/(conc[i*(NUMNODES+1)+2] +
Ks[i*(NUMNODES+1)+2] + nodes[2]);
          else if(ks[i*(NUMNODES+1)+1] * ks[i*(NUMNODES+1)+2] > 0) //
if reg by B and by C have the same sign
      in2 *= ks[i*(NUMNODES+1)+2]*nodes[2]/(conc[i*(NUMNODES+1)+2] +
Ks[i*(NUMNODES+1)+2] + nodes[2]);
          else if(Ks[i*(NUMNODES+1)+2] > 0) // if reg by C exists
      {
        in3 = ks[i*(NUMNODES+1)+2]*nodes[2]/(conc[i*(NUMNODES+1)+2] +
Ks[i*(NUMNODES+1)+2] + nodes[2]);
        if(ks[i*(NUMNODES+1)+2] < 0)
                    sign3 = -1;
                }

              if(in1 > 0 && sign1 < 0)
      in1 *= sign1;
              if(in2 > 0 && sign2 < 0)
                in2 *= sign2;
              if(in3 > 0 && sign3 < 0)
                in3 *= sign3;

          plus[i] += in1*conc[i*(NUMNODES+1)+0] +
in2*conc[i*(NUMNODES+1)+1] + in3*conc[i*(NUMNODES+1)+2];
        }
      else // node is input node
        {
          double in0 = base_input*kIA/((TOTALNODE-nodes[0]) + KIA +
base_input);
          double in1 = 0, in2 = 0, in3 = 0;
          int sign1 = 1, sign2 = 1, sign3 = 1;
          if(ks[0*(NUMNODES+1)+0] > 0) // if reg by A has the same sign
as input (positive)
      in0 *= ks[0*(NUMNODES+1)+0]*nodes[0]/(conc[0*(NUMNODES+1)+0] +
Ks[0*(NUMNODES+1)+0] + nodes[0]);
          else if(Ks[0*(NUMNODES+1)+0] > 0) // if there's regulation by
A on node i
      {
        in1 = ks[0*(NUMNODES+1)+0]*nodes[0]/(conc[0*(NUMNODES+1)+0] +
Ks[0*(NUMNODES+1)+0] + nodes[0]);
        if(ks[0*(NUMNODES+1)+0] < 0)
                    sign1 = -1;
                }
          if(ks[0*(NUMNODES+1)+1] > 0) // if reg by B has the same sign
as input
      in0 *= ks[0*(NUMNODES+1)+1]*nodes[1]/(conc[0*(NUMNODES+1)+1] +
Ks[0*(NUMNODES+1)+1] + nodes[1]);
          else if(ks[0*(NUMNODES+1)+0] * ks[0*(NUMNODES+1)+1] > 0) //
if reg by A and by B have the same sign
      in1 *= ks[0*(NUMNODES+1)+1]*nodes[1]/(conc[0*(NUMNODES+1)+1] +
Ks[0*(NUMNODES+1)+1] + nodes[1]);
          else if(Ks[0*(NUMNODES+1)+1] > 0) // if reg by B exists
      {
        in2 = ks[0*(NUMNODES+1)+1]*nodes[1]/(conc[0*(NUMNODES+1)+1] +
Ks[0*(NUMNODES+1)+1] + nodes[1]);
```

```
    if(ks[0*(NUMNODES+1)+1] < 0)
          sign2 = -1;
        }
     if(ks[0*(NUMNODES+1)+2] > 0) // if reg by C has the same sign
as input
   in0 *= ks[0*(NUMNODES+1)+2]*nodes[2]/(conc[0*(NUMNODES+1)+2] +
Ks[0*(NUMNODES+1)+2] + nodes[2]);
     else if(ks[0*(NUMNODES+1)+0] * ks[0*(NUMNODES+1)+2] > 0) //
if reg by A and by C have the same sign
   in1 *= ks[0*(NUMNODES+1)+2]*nodes[2]/(conc[0*(NUMNODES+1)+2] +
Ks[0*(NUMNODES+1)+2] + nodes[2]);
     else if(ks[0*(NUMNODES+1)+1] * ks[0*(NUMNODES+1)+2] > 0) //
if reg by B and by C have the same sign
   in2 *= ks[0*(NUMNODES+1)+2]*nodes[2]/(conc[0*(NUMNODES+1)+2] +
Ks[0*(NUMNODES+1)+2] + nodes[2]);
     else if(Ks[0*(NUMNODES+1)+2] > 0) // if reg by C exists
   {
    in3 = ks[0*(NUMNODES+1)+2]*nodes[2]/(conc[0*(NUMNODES+1)+2] +
Ks[0*(NUMNODES+1)+2] + nodes[2]);
    if(ks[0*(NUMNODES+1)+2] < 0)
          sign3 = -1;
        }

       if(in1 > 0 && sign1 < 0)
   in1 *= sign1;
       if(in2 > 0 && sign2 < 0)
        in2 *= sign2;
       if(in3 > 0 && sign3 < 0)
        in3 *= sign3;

     plus[0] += in0*(TOTALNODE - nodes[0]) +
in1*conc[0*(NUMNODES+1)+0] + in2*conc[0*(NUMNODES+1)+1] +
in3*conc[0*(NUMNODES+1)+2];
    }
  }
   // effect of constitutive activator/deactivator
   if(ks[i*(NUMNODES+1)+3] != 0)
  plus[i] +=
FCONC*ks[i*(NUMNODES+1)+3]*conc[i*(NUMNODES+1)+3]/(conc[i*(NUMNODES+1)+
3] + Ks[i*(NUMNODES+1)+3] + FCONC);
  }

 if(andtrack[0] == 0) // add effect of input if input node is using OR
logic
  plus[0] += base_input*kIA*(TOTALNODE-nodes[0])/((TOTALNODE-
nodes[0]) + KIA + base_input);
}

/* calculate timecourse metrics */
int get_metrics(vector<double> &timecourse, vector<double> &timevec,
double &init,
   double &final, double &peakheight, double &peaktime, double
&halfup,
   double &halfdown, double &sstime, double &integral, double &ten,
   double &ninety, double &inttopeak, double &inttohalf)
```

```
{
  // initialize quantities to track
  double halfheight;
  bool pastpeak = false, pasthalf = false;
  init = timecourse[0]; // output initial value
  final = timecourse.back(); // output final value
  peakheight = 0; // maximum output amplitude
  peaktime = 0; // time maximum output amplitude was reached
  halfup = 0; // time 50% max output amplitude was reached, starting
from low amplitude
  halfdown = sstime; // time 50% max output amplitude was reached,
starting from high amplitude
  sstime = timecourse.size() - 1; // time output reached steady state
  integral = 0; // integral of output
  ten = 0; // time 10% max output amplitude was reached, starting from
low amplitude
  ninety = 0; // time 90% max output amplitude was reached, starting
from high amplitude
  inttopeak = 0; // integral to 50% max amplitude
  inttohalf = 0; // integral to maximum amplitude

  int sign = 1; // = (1, -1) if output turns (on, off) in response to
input
  int ipeaktime = 1;
  vector<double> temp(timecourse.size());

  // make a copy of the timecourse, shifting initial value to zero and
taking absolute value.
  for(unsigned int i = 0; i < timecourse.size(); i++)
    {
      temp[i] = fabs(timecourse[i] - init);
      if(timevec[i] < INTTIME)
    integral += temp[i];
    }

  for(unsigned int i = 0; i < temp.size(); i++)
    if(temp[i] > peakheight)
      {
    peakheight = temp[i];
    ipeaktime = i;
    peaktime = timevec[i];
      }
  halfheight = peakheight/2;
  for(unsigned int i = 0; i < temp.size(); i++)
    {
      if(ninety == 0 and temp[i] >= 0.9*peakheight)
    ninety = timevec[i];
      if(ten == 0 and temp[i] >= 0.1*peakheight)
    ten = timevec[i];
      if(halfup == 0 and temp[i] >= halfheight)
    {
      halfup = timevec[i];
      pasthalf = true;
    }
      if(pastpeak == false && temp[i] >= peakheight)
```

```
     {
       pastpeak = true;
       pasthalf = false;
     }
      if(pastpeak == true && pasthalf == false && temp[i] <=
halfheight)
     {
       halfdown = timevec[i];
       break;
     }
    }
  for(unsigned int i = 0; i < temp.size(); i++)
    {
      if(temp[i] < halfheight)
     inttohalf += temp[i];
      if(temp[i] < peakheight)
     inttopeak += temp[i];
      else
     break;
    }

  if(timecourse[ipeaktime] < init)
    sign = -1;
  return sign;
}

/* initialize parameter vectors */
void start(vector<double> &ks, vector<double> &Ks, vector<int>
&andtrack)
{
  // initialize kcat's
  ks[0*(NUMNODES+1)+0] = kAA;
  ks[0*(NUMNODES+1)+1] = kBA;
  ks[0*(NUMNODES+1)+2] = kCA;
  ks[0*(NUMNODES+1)+3] = kFAA;
  ks[1*(NUMNODES+1)+0] = kAB;
  ks[1*(NUMNODES+1)+1] = kBB;
  ks[1*(NUMNODES+1)+2] = kCB;
  ks[1*(NUMNODES+1)+3] = kFBB;
  ks[2*(NUMNODES+1)+0] = kAC;
  ks[2*(NUMNODES+1)+1] = kBC;
  ks[2*(NUMNODES+1)+2] = kCC;
  ks[2*(NUMNODES+1)+3] = kFCC;

  // initialize Km's
  Ks[0*(NUMNODES+1)+0] = KAA;
  Ks[0*(NUMNODES+1)+1] = KBA;
  Ks[0*(NUMNODES+1)+2] = KCA;
  Ks[0*(NUMNODES+1)+3] = KFAA;
  Ks[1*(NUMNODES+1)+0] = KAB;
  Ks[1*(NUMNODES+1)+1] = KBB;
  Ks[1*(NUMNODES+1)+2] = KCB;
  Ks[1*(NUMNODES+1)+3] = KFBB;
  Ks[2*(NUMNODES+1)+0] = KAC;
  Ks[2*(NUMNODES+1)+1] = KBC;
  Ks[2*(NUMNODES+1)+2] = KCC;
  Ks[2*(NUMNODES+1)+3] = KFCC;

  // initialize logic coder
  andtrack[0] = Aand;
  andtrack[1] = Band;
  andtrack[2] = Cand;
}

/* end */
```

A.2 doseResponseMetrics.cc

```
/*****************************************************
 * doseResponseMetrics.cc
 * Jaline Gerardin 2010
 *
 * given a dose response, calculate steepness and EC50
 *
 * command line arguments:
 * [number of data points] [responses] [doses]
 *****************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <vector>
#include <iostream>
using namespace std;

int main(int argc, char**argv)
{
  // number of data points in dose response
  int numpoints = atoi(argv[1]);

  // read in response values
  vector<double> ydata(numpoints);
  for(int i = 0; i < numpoints; i++)
    ydata[i] = atof(argv[i+2]);

  // find maximum and minimum response values
  double max = ydata[numpoints-1];
  double min = ydata[0];
  for(int i = 0; i < numpoints; i++)
    {
      if(ydata[i] > max)
    max = ydata[i];
      if(ydata[i] < min)
    min = ydata[i];
    }

  // calculate values for 90%, 50%, and 10% response
  double ninety = (max - min)*0.9 + min;
  double fifty = (max - min)*0.5 + min;
  double ten = (max - min)*0.1 + min;

  // read in dose values
  vector<double> xdata(numpoints);
  for(int i = 0; i < numpoints; i++)
    xdata[i] = atof(argv[i+2+numpoints]);

  // identify indexes bracketing 10%, 50%, and 90% response
  int tenm = 0, tenp = 1;
  int ninem = numpoints - 2, ninep = numpoints - 1;
  int fifm = 0, fifp = numpoints-1;
  for(int i = 0; i < numpoints; i++)
    {
      if(ydata[i] > ten)
    {
      tenm = i-1;
      tenp = i;
      break;
    }
    }
  for(int i = 0; i < numpoints; i++)
    {
      if(ydata[i] > fifty)
    {
      fifm = i-1;
      fifp = i;
      break;
    }
    }
  for(int i = numpoints-1; i >= 0; i--)
    {
      if(ydata[i] < ninety)
    {
      ninem = i;
      ninep = i+1;
      break;
    }
    }

  // use linear interpolation between bracketing indexes to estimate
  // doses for 10%, 50%, and 90%
  ten -= ydata[tenm];
  ten /= (ydata[tenp] - ydata[tenm]);
  fifty -= ydata[fifm];
  fifty /= (ydata[fifp] - ydata[fifm]);
  ninety -= ydata[ninem];
  ninety /= (ydata[ninep] - ydata[ninem]);

  double tenint = (xdata[tenp] - xdata[tenm])*ten + xdata[tenm];
  double nineint = (xdata[ninep] - xdata[ninem])*ninety + xdata[ninem];
  double fifint = (xdata[fifp] - xdata[fifm])*fifty + xdata[fifm];

  // write to stdout
  cout << max << "\t" << tenint/nineint << "\t"
       << fifint << endl;

  return 0;
}
```

A.3 runSimulations.py

```
#!/usr/bin/python

####################################################################
# runSimulations.py
# Jaline Gerardin 2010
#
# Given a parameter file, the function runOneGroup simulates circuit
# response to inputs of various duration. Temporal dose responses and
# kintic filtering metrics (temporal ultrasensitivity score and
# trigger time) are calculated and outputed to file.
#
####################################################################

import commands
import os

HOMEDIR = '/netapp/home/gerardin/noise/'

# runOneGroup reads in parameters from file and outputs a temporal
# dose response and kinetic filtering metrics for each parameter set
# that ran successfully.
#
# runOneGroup's 'program' argument is the code that simulates circuit
# response to input.
#
# required format for parameter file:
# line 1: basal input, change in input, node logic for 3 nodes,
# topology ID number, starting parameter ID number
# subsequent lines: 26 kcat's and Km's for each line. Link regulation
# type (activator, inhibitor, absent) is coded in kcat values
# (positive, negative, 0 respectively)
#
# Temporal dose responses are outputed to rundir and kinetic filtering
# metrics are outputed to directory stem_[node A logic][node B
# logic][node C logic]
def runOneGroup(program, paramfile, rundir, stem):

    # open parameter file and read in settings in first line
    finparam = open(paramfile)
    settings = finparam.readline()
    [base_input, change, Aand, Band, Cand, topnum, circuit_index] = [n
for n in settings.split()]
    circuit_index = int(circuit_index)

    # file for storing temporal dose response curves
    rawoutfile = rundir + 'raw/data_' + str(topnum)
    # file for storing temporal dose reponse metrics
    metdir = HOMEDIR + stem + '_' + Aand + Band + Cand + '/'
    outfile = metdir + 'data_' + str(topnum)

    # start indexing parameters at circuit_index
    param = circuit_index
    # input durations to test
    intimes = [0.25, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 
50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 800, 1000, 2000, 
3000, 5000, 6000, 8000, 10000, 20000, 50000]

    # for each parameter set in parameter file
    for thisparam in finparam.readlines() :
        print 'running top ' + str(topnum) + ' param ' + str(param)

        data = [] # stores maximum output amplitudes
        times = [] # stores input durations
        params = ' '.join(thisparam.split())

        # try to initialize the circuit
        initialization = runOneTime(program, intimes[0], base_input, 
change, Aand, Band, Cand, params, '')
        # if circuit did not reach steady state
        if(initialization < 0) :
            param += 1
            continue # continue to next parameter set

        # apply shortest input duration
        diff0 = runOneTime(program, intimes[0], base_input, change, 
Aand, Band, Cand, params, initialization)
        # if error in simulation
        if diff0 < 0 :
            param += 1
            continue # continue to next parameter set

        # apply longest input duration
        difflast = runOneTime(program, intimes[len(intimes)-1], 
base_input, change, Aand, Band, Cand, params, initialization)
        # if amplitude is too small or amplitude is identical for 
shortest and longest inputs
        if(difflast < 10**-30 or abs(difflast - diff0) < 10**-30) :
            param += 1
            continue # continue to next parameter set

        # store shortest input duration and output amplitude
        times.append(intimes[0])
        data.append(diff0)

        # check that all input duration simulations finished without 
error
        allok = 1

        # test the remaining input durations
        for i in range(1, len(intimes)-1) :
            next = runOneTime(program, intimes[i], base_input, change, 
Aand, Band, Cand, params, initialization)
            if next < 0 : # if error in simulations, break out of loop
                allok = 0
                break
            # store output amplitude and input duration
            data.append(next)
            times.append(intimes[i])
            # starting with 6th input duration, check if output
amplitude is equivalent to amplitude for longest input duration
            if i > 4 and abs(next - difflast) < 10**-15 : # if so, skip
remaining input durations and break out of loop
                break

        # if simulation error occurred, continue to next parameter set
        if allok == 0 :
            param += 1
            continue

        # if all input durations were run, append data for longest
duration
        if len(times) == len(intimes) - 1 :
            data.append(difflast)
            times.append(intimes[len(intimes)-1])

        # record maximum output amplitudes to file
        with open(rawoutfile, 'a') as fout :
            fout.write(str(topnum) + '\t' + str(param) + '\t')
            for i in range(len(data)):
                fout.write(str(data[i])+'\t')
            fout.write('\n')

        # calculate temporal dose response metrics and write to file if
calculations were error-free
        ultradata = getultra(times, data)
        if len(str(ultradata)) > 0 and ultradata != -1:
            with open(outfile, 'a') as fout :
                fout.write(str(topnum) + '\t' + str(param) + '\t' +
str(ultradata) + '\t' + str(max(data) - min(data)) + '\n')

        # increment parameter tracker
        param += 1
    finparam.close()


# run simulation of one parameter set, one input duration
def runOneTime(program, intime, base_input, change, Aand, Band, Cand,
params, initialization) :

    line = commands.getoutput('%(program)s %(intime)s %(base_input)s
%(change)s %(Aand)s %(Band)s %(Cand)s %(params)s %(initialization)s' %
vars())

    try :
        data = [float(n) for n in line.split()]
    except ValueError:
        print line
        return -1

    if(data[0] == -1) : # if circuit could not reach steady state
during initialization, return error
        return -1
    if initialization == '' : # if in initialization phase, return
initial steady state
        return line
    if data[10] == -1 or len(data) < 24 : # if something went wrong
during simulation, return error
        return -1

    return data[14] # otherwise return maximum output amplitude


# calculate temporal dose response metrics
def getultra(times, data) :

    # program used to calculate temporal dose response metrics
    program = 'doseResponseMetrics'

    # calculate metrics
    numpoints = len(times)
    mytimes = ' '.join(['%.2f'%x for x in times])
    mydata = ' '.join(['%.15f'%x for x in data])
    TUmetrics = commands.getoutput(HOMEDIR + program + ' ' +
str(numpoints) + ' ' + mydata + ' ' + mytimes)

    # if error in calculating metrics, return -1
    if ('nan' in TUmetrics or 'inf' in TUmetrics) :
        return -1

    # return metrics
    return TUmetrics
```

Appendix B

PARAMETERS USED TO CONSTRUCT FIGURES

Figure	Circuit	Link	kcat	Km
4.1	1-node positive feedback OR	Input	2.09411	3.37287
		AA	1.5531	0.06026
		A deactivator	3.24638	0.0415
4.2 top	1-node positive feedback OR	Input	3.05351	31.369
		AA	3.34811	13.2434
		A deactivator	0.44648	0.0062
4.2 bottom	1-node positive feedback OR	Input	0.19724	7.14496
		AA	3.74628	28.7409
		A deactivator	0.27277	0.02429
4.3	1-node positive feedback AND	Input	4.33112	0.14962
		AA	0.94493	0.06173
		A deactivator	0.10195	12.7057
4.4 top	1-node positive feedback AND	Input	2.85365	0.00429
		AA	6.9024	8.67961
		A deactivator	1.13868	65.013
4.4 bottom	1-node positive feedback AND	Input	1.7402	0.00135
		AA	0.13728	0.00123
		A deactivator	0.31915	95.4993
4.5	2-node positive feedback AND	Input	9.79039	6.56901
		AB	1.58489	0.13474
		BA	5.80497	0.00314
		A deactivator	3.79665	61.0239
		B deactivator	1.92575	0.00387
4.6	3-node double inhibition	Input	1.71791	25.704
		AB	1.03134	0.00245471
		BC	4.96821	0.00389493
		A deactivator	0.493856	0.0989692
		B activator	0.29964	2.43781
		C activator	7.67361	0.21208
4.7 top	3-node double inhibition	Input	0.680769	34.4747
		AB	1.70687	0.00958297

		BC	0.591834	0.00338844
		A deactivator	0.560273	0.0615177
		B activator	0.810588	23.632
		C activator	0.846837	0.351156
4.7 bottom	3-node double inhibition	Input	0.128115	28.6748
		AB	4.10771	0.00145714
		BC	4.79071	0.0273842
		A deactivator	3.47696	50.2343
		B activator	0.105439	0.00858025
		C activator	1.95794	0.213059
4.8	2-node double inhibition	Input	0.74611	0.13725
		AB	0.29147	0.01939
		BA	3.6191	0.00254
		B activator	0.74405	71.6143
4.9 top	2-node double inhibition	Input	2.95529	0.00217
		AB	0.24729	0.00168
		BA	4.58775	0.05689
		B activator	0.34277	43.8026
4.9 bottom	2-node double inhibition	Input	2.98263	0.02121
		AB	0.65433	0.01152
		BA	7.38244	0.63023
		B activator	0.15689	0.52541
4.10	Buffering bistable inverter	Input	0.565458	0.00248599
		AB	3.85301	15.1705
		AC	0.125314	0.00300608
		BA	1.28529	0.0295801
		BB	0.153957	0.296483
		A deactivator	0.298813	0.168461
		C activator	2.47172	86.3973
4.11 top	Buffering bistable inverter	Input	4.74242	17.5792
		AB	5.90745	3.2961
		AC	0.1293	0.511093
		BA	1.69824	22.6725
		BB	0.947109	1.15744
		A deactivator	0.289601	10.1625
		C activator	5.14991	28.5759
4.11 bottom	Buffering bistable inverter	Input	1.35644	0.0658415
		AB	1.69434	1.26619

		AC	1.39959	0.0156315
		BA	6.01174	68.3912
		BB	0.137025	0.00165386
		A deactivator	6.20583	82.5087
		C activator	7.30802	13.9476
4.12 top	Buffering bistable inverter	Input	3.29458	4.72607
		AA	0.657355	4.61318
		AC	0.626037	1.1272
		BA	0.574116	2.63027
		BB	1.1102	0.0171396
		CA	0.181886	5.2723
		CB	8.34449	0.00102565
		B deactivator	4.67951	1.55597
		C activator	0.236483	4.9545
4.12 bottom	Buffering bistable inverter	Input	1.54028	0.0145211
		AC	1.82642	0.0021752
		BB	1.32617	0.00710395
		BC	7.05342	0.042462
		CB	5.47772	0.0185994
		A deactivator	5.83983	0.328473
		B deactivator	2.834	0.428549
		C activator	0.100415	0.0152581
4.13	Coherent feed forward AND	Input	1.67109	9.98849
		AB	0.196426	4.98884
		AC	4.75554	0.149796
		BC	2.90402	0.0959401
		A deactivator	0.180053	0.724436
		B deactivator	1.05779	4.43098
		C deactivator	1.61287	0.0110917
4.14 top	Coherent feed forward AND	Input	0.430527	0.172783
		AB	0.120448	0.00305141
		AC	9.38426	0.02509
		BC	5.32844	0.00387258
		A deactivator	0.585599	0.109018
		B deactivator	3.4261	1.21339
		C deactivator	2.76949	0.063314
4.14 bottom	Coherent feed forward AND	Input	1.69044	0.187932
		AB	0.774105	7.02263

		AC	2.37575	0.450298
		BC	3.96096	0.102329
		A deactivator	0.677642	0.00119536
		B deactivator	6.54034	47.152
		C deactivator	4.44631	0.0224388
4.18	Coherent feed forward AND circuit A	Input	0.45288	0.00119
		AB	2.07329	0.00659
		AC	7.2144	0.20855
		BC	0.10214	0.00491
		A deactivator	0.2638	2.08857
		B deactivator	2.83635	0.02791
		C deactivator	0.18782	0.00364
	Coherent feed forward AND circuit B	Input	3.18684	0.0402
		AB	0.1457	15.6513
		AC	3.70817	0.00194
		BC	0.67459	0.00393
		A deactivator	0.20398	0.02658
		B deactivator	0.75903	1.71771
		C deactivator	0.31307	0.00458
	Coherent feed forward AND circuit C	Input	0.85925	0.71146
		AB	8.02269	13.9508
		AC	5.41427	0.004
		BC	1.21222	1.93108
		A deactivator	0.10277	0.00828
		B deactivator	0.104	0.27428
		C deactivator	1.22524	0.00118
	Coherent feed forward AND circuit D	Input	1.1551	0.17995
		AB	2.71269	3.74498
		AC	7.2711	0.01006
		BC	1.22783	0.01661
		A deactivator	0.11411	0.00678
		B deactivator	1.11527	0.01402
		C deactivator	1.5313	0.00394
	Coherent feed forward AND circuit E	Input	0.3525	0.10859
		AB	0.4202	0.00333
		AC	0.99165	0.06039
		BC	8.82673	0.01361
		A deactivator	0.58833	0.17974

		B deactivator	0.1057	0.00447
		C deactivator	6.08808	0.02549
	Coherent feed forward AND circuit F	Input	1.6641	86.9461
		AB	1.44344	0.00487
		AC	8.02971	0.0308
		BC	8.89201	0.49017
		A deactivator	0.44974	2.24285
		B deactivator	1.6761	6.59478
		C deactivator	6.18671	0.00707
	Coherent feed forward AND circuit G	Input	0.9637	14.6707
		AB	0.16929	0.02395
		AC	3.50187	0.07134
		BC	3.23773	0.14062
		A deactivator	0.15485	23.2943
		B deactivator	0.1416	11.2954
		C deactivator	0.26436	0.00193
	Coherent feed forward AND circuit H	Input	1.63878	2.261
		AB	1.01845	0.00336
		AC	2.54706	0.01789
		BC	9.96965	0.06219
		A deactivator	0.20989	0.00236
		B deactivator	0.10858	26.9836
		C deactivator	6.37206	0.01798
4.19	3-node double inhibition	Input	0.29239	70.3396
		AB	0.1641	0.00132
		BC	0.60292	0.00134
		A deactivator	0.65322	1.97129
		B activator	0.21034	18.6509
		C activator	4.40555	30.6761
	3-node double activation	Input	0.29239	70.3396
		AB	0.1641	0.00132
		BC	0.60292	0.00134
		A deactivator	0.65322	1.97129
		B deactivator	0.21034	18.6509
		C deactivator	4.40555	30.6761
4.20	Decreasing deactivator	Activator	1	0.5
		Deactivator	1	0.5
	Increasing activator	Activator	1	0.5

		Deactivator	1	0.5
4.22B	Top, no perturbation	Input	3.92121	23.7657
		AB	0.67911	0.57372
		AC	4.67584	0.00267
		BC	9.08071	0.00136
		A deactivator	1.25147	79.919
		B deactivator	1.499	0.02126
		C deactivator	3.7418	0.00244
	Top, increase kcat	Input	3.92121	23.7657
		AB	6.7911	0.57372
		AC	4.67584	0.00267
		BC	9.08071	0.00136
		A deactivator	1.25147	79.919
		B deactivator	1.499	0.02126
		C deactivator	3.7418	0.00244
	Top, decrease kcat	Input	3.92121	23.7657
		AB	0.067911	0.57372
		AC	4.67584	0.00267
		BC	9.08071	0.00136
		A deactivator	1.25147	79.919
		B deactivator	1.499	0.02126
		C deactivator	3.7418	0.00244
	Bottom, no perturbation	Input	0.34521	73.969
		AB	4.99436	9.4504
		AC	1.43107	0.00156
		BC	4.032	0.00182
		A deactivator	0.13686	15.9074
		B deactivator	0.1614	0.00865
		C deactivator	0.45442	0.00627
	Bottom, increase kcat	Input	0.34521	73.969
		AB	49.9436	9.4504
		AC	1.43107	0.00156
		BC	4.032	0.00182
		A deactivator	0.13686	15.9074
		B deactivator	0.1614	0.00865
		C deactivator	0.45442	0.00627
	Bottom, decrease kcat	Input	0.34521	73.969
		AB	0.499436	9.4504

		AC	1.43107	0.00156
		BC	4.032	0.00182
		A deactivator	0.13686	15.9074
		B deactivator	0.1614	0.00865
		C deactivator	0.45442	0.00627
4.24 left	Coherent feed forward AND	Input	1.91602	0.00959401
		AB	0.193821	39.5367
		AC	5.03501	0.0630957
		BC	7.46449	0.00286088
		A deactivator	0.10666	22.1055
		B deactivator	8.60201	96.1612
		C deactivator	0.867361	0.00578096
4.24 right	Coherent feed forward AND	Input	0.497966	0.0824138
		AB	3.1521	0.0472063
		AC	1.18522	0.00430527
		BC	2.3518	0.0161622
		A deactivator	0.144145	14.1091
		B deactivator	6.20012	0.0840427
		C deactivator	1.01065	0.00191205
4.31	1-node positive feedback OR	Input	2.09411	3.37287
		AA	1.5531	0.06026
		A deactivator	3.24638	0.0415
4.32	1-node positive feedback AND	Input	4.33112	0.14962
		AA	0.94493	0.06173
		A deactivator	0.10195	12.7057
4.33 integrator	3-node positive feedback OR	Input	7.82348	4.30527
		AC	0.393007	0.496021
		BC	3.5678	0.00140605
		CC	1.17382	0.0325462
		A deactivator	0.116466	0.238781
4.33 absolute timer	3-node positive feedback AND	Input	9.89464	0.187499
		AC	0.348819	0.0231739
		BC	0.273401	3.97649
		CC	9.48855	5.13452
		A deactivator	1.86466	0.512271

Publishing Agreement

It is the policy of the University to encourage the distribution of all theses, dissertations, and manuscripts. Copies of all UCSF theses, dissertations, and manuscripts will be routed to the library via the Graduate Division. The library will make all theses, dissertations, and manuscripts accessible to the public and will preserve these to the best of their abilities, in perpetuity.

Please sign the following statement:

I hereby grant permission to the Graduate Division of the University of California, San Francisco to release copies of my thesis, dissertation, or manuscript to the Campus Library to provide access and preservation, in whole or in part, in perpetuity.

Author Signature

12/16/13

Date