THE SUBUNIT EXCHANGE RATE OF THE CYANOBACTERIAL CIRCADIAN CLOCK COMPONENT KAIC IS INDEPENDENT OF PHOSPHORYLATION STATE

A Thesis

by

ELIHU CARL IHMS

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2007

Major Subject: Biochemistry

THE SUBUNIT EXCHANGE RATE OF THE CYANOBACTERIAL CIRCADIAN CLOCK COMPONENT KAIC IS INDEPENDENT OF PHOSPHORYLATION STATE

A Thesis

by

ELIHU CARL IHMS

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Andy C. LiWang
Committee Members,	James W. Golden
	David H. Russell
Head of Department,	Gregory D. Reinhart

December 2007

Major Subject: Biochemistry

ABSTRACT

The Subunit Exchange Rate of the Cyanobacterial Circadian Clock Component KaiC is Independent of Phosphorylation State. (December 2007)

Elihu Carl Ihms, B.S. Indiana Wesleyan University

Chair of Advisory Committee: Dr. Andy LiWang

The study of the *in vitro* circadian oscillator of the cyanobacterium *Synechococcus elongatus* has uncovered a complex interplay of its three protein components. Synchronization of the clock's central oscillatory component, KaiC, has been thought to be achieved through subunit shuffling at specific intervals during the clock's period. By utilizing an established fluorescence-based analysis on completely phosphorylated and dephosphorylated mutants as well as wild-type KaiC, this study has shown that shuffling rates are largely unaffected by phosphorylation state. These findings conflict with previous reports and hence revise our understanding of this oscillator.

ACKNOWLEDGEMENTS

I would like to thank my committee chair and advisor, Dr. Andy LiWang, for his continued scientific guidance and insight. His insistence to quickly troubleshoot and analyze problematic situations rationally is a valuable lesson that has helped me greatly.

I would also like to thank Dr. Mauricio Lasagna, for his tireless patience in regards to many fluorescence and spectroscopy-related questions. By vicariously using his many years of practical experience, I have been able to avoid many of the potential pitfalls and caveats present in the techniques used by this study. In addition, my labmates Carl Carruthers and Yong-Ick Kim have helped immensely, not only in the cerebral discussion and analysis of clocks-related material, but for the day-to-day support and help they provided.

Finally, I would like to thank my patient and loving wife, who has kept me sane through long hours in the lab, and who has remained by my side regardless.

NOMENCLATURE

6-IAF	6-Iodoacetamidofluroescein
ATP	Adenosine Triphosphate
CBB	Coomassie Brilliant Blue
DTT	Dithiothreitol
DMSO	Dimethyl sulfoxide
EDTA	Ethylenediaminetetraacetic acid
IAEDANS	5-({[(2-iodoacetyl)amino]ethyl}amino)-naphthalene-1-sulphonic acid
FRET	Fluorescence (or Forster) Resonance Energy Transfer
GST	Glutathione-S-Transferase
TCEP	tris-(2-carboxyethyl)phosphine
Tris	2-Amino-2-(hydroxymethyl)-1,3-propanediol
WT	Wild-Type

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION: SYNCHRONIZATION OF THE KAI OSCILLATOR

Circadian clocks are endogenous oscillators responsible for the 24 hour periodic behavior of many organisms. The central oscillator of the cyanobacteria *Synechococcus elongatus* clock, responsible for preparing the organism for the upcoming light phase of the day, has been amenable for study due to its relatively simple construction and solved structures of its components(1,2,3). The *S. elongatus* clock is comprised of three proteins: the two domain hexameric KaiC, which is sinusoidally phosphorylated during the clock's progression; KaiA, which is responsible for stimulating KaiC autophosphorylation; and KaiB which negates the action of KaiA (4,5,6,7). These three components, expressed recombinantly and purified separately, may be combined in the presence of ATP to reconstitute a synchronized and temperature-compensated chemical clock with a ~22 hr period (8).

One of the crucial aspects of the *in vitro* circadian oscillator is the mechanism by which individual KaiC hexamers stay synchronized with each other in solution. As postulated by Emberly and Wingreen (9), and further formalized by Yoda et al. (10) this most likely occurs by subunit exchange between hexamers. Kageyama et al. demonstrated the presence of shuffling through the ability to precipitate the entire KaiC population in a mixture of 50% FLAG-tagged and 50% untagged WT KaiC monomers (11). It was also stated that KaiA significantly inhibited this shuffling. More recently, Mori et. al. was able to demonstrate the existence of shuffling by changes in

---

This thesis follows the style of *Biochemistry.*

fluorescence resonance energy transfer (FRET) between fluorescently labeled KaiC monomers over time, and generated a mathematical model dependent upon differing shuffling rates (7). In contrast with the observations of Kageyama et al. these shuffling rates did not change after the addition of the other clock components.

We show in this study that shuffling occurs irrespective of phosphorylation state, and that the average half-life of KaiC hexamers in solution is approximately 1 hour. Thus, shuffling appears to remain constant throughout the clock period. These findings conflict with previous reports and have hence revised our understanding of this oscillator.

2. EXPERIMENTAL PROCEDURES

2.1 Protein Expression and Purification

KaiA used in the phosphorylation assay was expressed in E. Coli BL21(DE3) and purified as previously described (12). Glutathione-s-transferase (GST) N-terminus tagged KaiC variants were expressed in DH5α cells grown for three days at 30°C. The host bacteria were then centrifuged and stored at -80°C. Frozen cell pellets were resuspended in Buffer A, consisting of 50 mM Tris pH 7.3, 150 mM NaCl, 5 mM ATP, 10 mM MgCl2, 1 mM EDTA, and 1 mM DTT (Dithiothreitol). Cells were then cracked twice by french press at 16,000psi, and lysate clarified by centrifugation for 1 hour at 12000g. Supernatant was loaded onto a previously equilibrated 1mL GSTrap FF (GE LifeScience, Piscataway NJ), column at 0.5 mL/min. Column was washed with 60 mL Buffer A, and protein eluted into ~8 mL with Buffer B (Buffer A + 10 mM reduced glutathione). The GST tag was cleaved from KaiC by overnight incubation at 4°C with Precission Protease (GE LifeScience, Piscataway NJ) according to the manufacturers protocol. GST and Precission Protease were then removed via a second pass through a clean 1 mL GSTrap column. Elutant containing free KaiC was buffer exchanged into reaction buffer (20 mM Tris pH 8.0, 150 mM NaCl, 5 mM MgCl2, 0.5 mM EDTA, 0.1 mM ATP) via a GE LifeScience HiPrep 26/10 desalting column. Subsequent concentration via 10,000 MWCO spin concentrator yielded KaiC to >95% purity.

2.2 Protein Labeling and Refinement

Stock solutions of probes IAEDANS (5-({[(2-iodoacetyl)amino]ethyl}amino)-naphthalene-1-sulphonic acid) and 6-IAF (6-Iodoacetamidofluroescein) (Molecular Probes, Eugene Oregon) were dissolved in DMSO and calibrated to a final concentration of 50 mM. For IAEDANS labeling, KaiC in reaction buffer was brought to 1 mM ATP and 10x molar excess of both respective label and TCEP (tris-(2-carboxyethyl)phosphine). Both were added with continuous stirring and allowed to incubate for 2 hours at room temperature. For IAF labeling, 50x molar excess of both label and TCEP were used, and labeling was allowed to occur overnight at room temperature. After the specified labeling interval, unreacted conjugate was neutralized with the addition of a molar excess of DTT. DTT, TCEP, and free conjugate were then removed via a GE Life Sciences 5 mL HiTrap desalting column equilibrated with reaction buffer containing 100 μM ATP. Labeling efficiency was determined by Bradford assay (Bio-Rad, Hercules CA) and absorption of the corresponding probes using the following molar absorptivities: IAEDANS, ε_{336} = 5,700 mol^{-1} cm^{-1}, IAF: ε_{491} = 82,000 mol^{-1} cm^{-1}.

2.3 Spectroscopic and Experimental Parameters

Samples were equilibrated at 30 °C in a water bath for 30 minutes before mixing. Immediately before the first data point (t_0), 900 μL of 3.45 μM IAEDANS-labeled KaiC, 900 μL of 3.45 μM IAF labeled KaiC, and 18 μL 100 mM ATP-containing reaction

buffer were combined in a 3.5 mL quartz cuvette (Starna Cells, Atascadero CA) and the data collection process initiated.

Fluorescence intensity measurements were taken using a SLM 4800 fluorometer modified with ISS photon-counting electronics. Instrument and data acquisition was preformed through ISS Vinci. Excitation was by xenon arc lamp. Excitation monochromator was used with a 4 nm monochromator bandpass. For spectra acquisition, readings were taken every 1 nm using an 8 nm bandpass, and 20-point averaging. Post-acquisition smoothing was achieved by an custom developed script performing a exponential averaging function. For single intensity datapoints, an interference filter of unknown manufacture with a 10 nm transmission range centered around 470 nm was used. Timepoints corresponding to 0, 10, 20, 30, 45, 60, 90, 120, 180, 240, 300, 360, 480, 600, and 720 minutes after mixing were taken with 300 point averaging. Fluorescence spectra from 450 nm to 550 nm were taken after the initial and final timepoints. Sample cuvettes were thermostated via circulating water bath at 30 °C for the duration of the experiment. Control spectra of singly-labeled labeled KaiC were taken, as well as reaction buffer containing only 1 mM ATP.

2.4 Shuffling and Labeling Effect Simulation

Four scripts written in the computer scripting language Perl were used to simulate shuffling of monomer populations between oligiomers of specified size (See Appendix A for the full source code and example input file). Script mklist.pl generated an ASCII list of colon-delimited monomers belonging to newline-delimited oligiomers

that satisfied a specified donor and acceptor labeling efficiency. Donor and acceptor-containing oligiomers were generated such that they were separate at the start of the experiment. Script move.pl randomly selected subunits within hexamers and exchanged upon satisfaction of a user-specified probability value, resulting in a pseudo shuffling "rate". Script getfret.pl calculated the number of donors in immediate proximity to acceptors in a given file, and returned this value.

The shuffle.pl program served as a wrapper script, taking a specified parameter file as input. This wrapper then called the other scripts in sequence, first building a population file, shuffling it a specified number of iterations, and collecting proximity values. "Snapshot" files containing the population state from each iteration breakpoint were saved to disk, enabling a rescue should the executable halt or malfunction.

3. RESULTS AND DISCUSSION

3.1 Spectral Properties of Labeled KaiC

Previous work by Mori et al. demonstrated the efficacy of using FRET (fluorescence or Forster resonance energy transfer) to follow the exchange of KaiC subunits between hexamers (7). WT KaiC contains three cysteines that serve as potential labeling targets by thiol-reactive fluorescent probes. In figure 1, two C-termini (CII) domains of KaiC (PDB: 1TF7) are depicted, the cysteines of the left domain labeled. Cysteine 348, present near the ATP-binding domain, is partially solvent-exposed, whereas cysteines 274 and 306 are largely buried. Based on measurements of the crystal structure, the average distance between C348 residues of neighboring subunits is 36 angstroms.

In this study, separate aliquots of KaiC were labeled with the probes IAEDANS (donor, Ex=336 nm, Em= 470 nm) and IAF (acceptor, Ex=491 nm, Em=515 nm). This pair has a Forster radius of 55 angstroms, which is the distance at which 50% transfer efficiency occurs. As the three intrinsic cysteines of KaiC all lie within 24-52 Å from one another when measured across the subunit interface, the IAEDANS/IAF pair is well suited for this experiment.

As seen in the blue emission spectra present in Figure 2, samples labeled with IAEDAN displayed a broad fluorescence peak centered on 480 nm when excited with light at 336 nm. Fluorescein-labeled KaiC emission, on the other hand, is seen as a somewhat narrower peak present at 520 nm in red. Fluorescein absorbs weakly at 336

nm, but contributes a significant portion to the spectra because of IAEDANS similarly low extinction coefficient at its lambda max (ε_{336}=5,700 M^{-1} cm^{-1}). Because of the large degree of overlap between the emission spectra of AEDANS and fluorescein emission starting at 480 nm, quenching of the donor in the presence of the acceptor was chosen as a more straightforward measure of the existence of FRET. As fluorescein λ_{max} in water is 494 nm, this overlap with AEDANS's emission spectra indicates potential energy transfer. Also present in figure 1A is the transmission profile of the 467-10 interference filter used for acquiring donor intensity during the experiment (gray outline). In this study, a filter was chosen over the available monochromator due to its higher efficiency and selective cutoff range. The minimal amount of overlap of the filter's profile with fluorescein fluorescence spectra ensures observed intensity to be primarily due to AEDANS.

The overlaid spectral lines present in Figure 3 demonstrate the change in spectra over the course of the experiment, each corresponding to a specific time point. As more acceptor-labeled subunits come into proximity to donor-labeled subunits, energy transfer occurs, resulting in donor intensity quenching and acceptor intensity enhancement. Of particular note is the reduced dynamic range present at the acceptor peak due to the concurrent donor quenching at the same wavelength, for reasons noted above. Photobleaching of acceptor and donor probes was determined to be negligible over the course of the experiment.

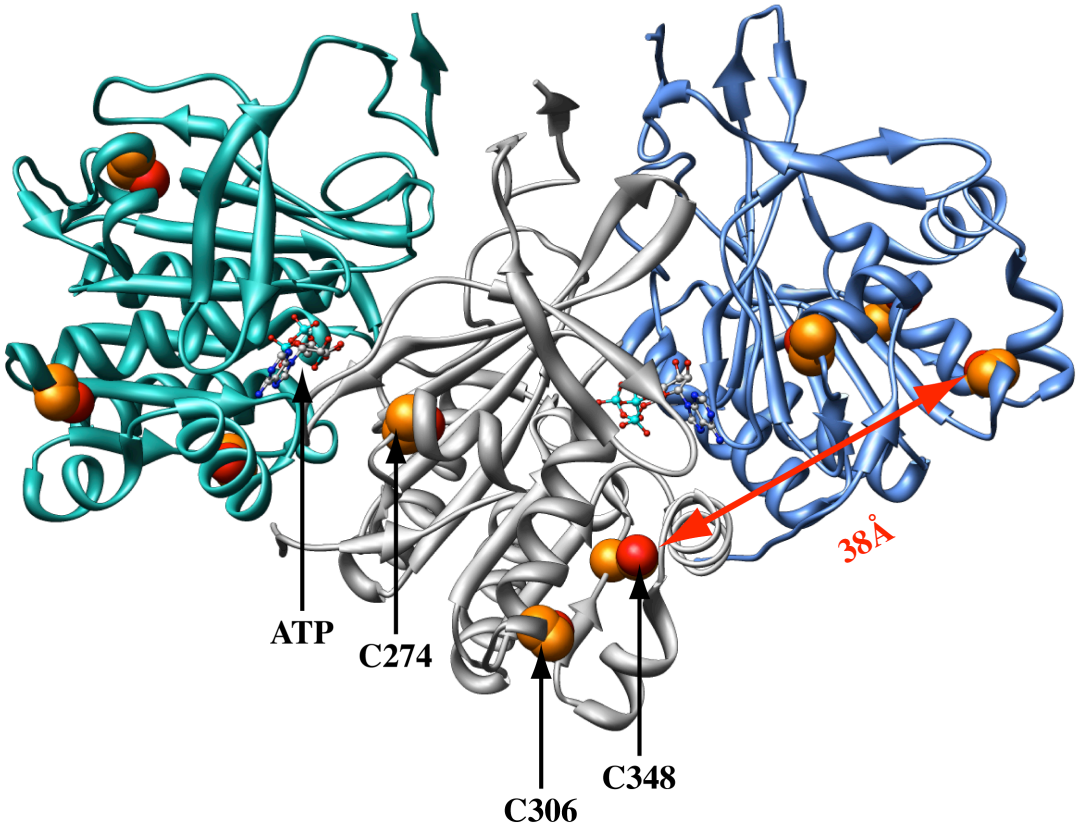

FIGURE 1: KaiC Structure Revealing Intrinsic Cysteines

Figure 1 Caption: Three CII (C-termini) domains of KaiC WT (PDB: 1TF7) are depicted. Cysteines C274 and C306 are predominately buried, whereas C348 is partially exposed to solution. Distance and putative FRET path from C348 of one subunit to C348 of neighboring subunit is shown in red. ATP analog ATP-γS is shown, present at the interface.

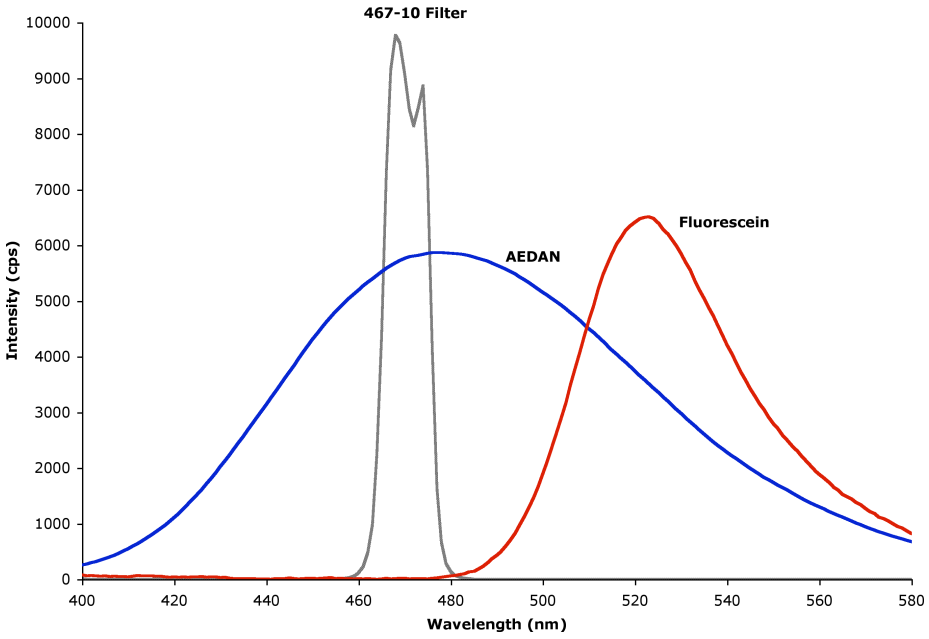

FIGURE 2: Fluorescence Spectra of Singly Labeled KaiC

Figure 2 Caption: Pure AEDANS-labeled KaiC is shown in the blue trace with maxima at approximately 480 nm. Pure AF-labeled KaiC is depicted in the red trace present at 525 nm. The gray outline between 460 nm and 480 nm is the transmission profile of the spectrophotometric filter used to exclude fluorescein intensity.

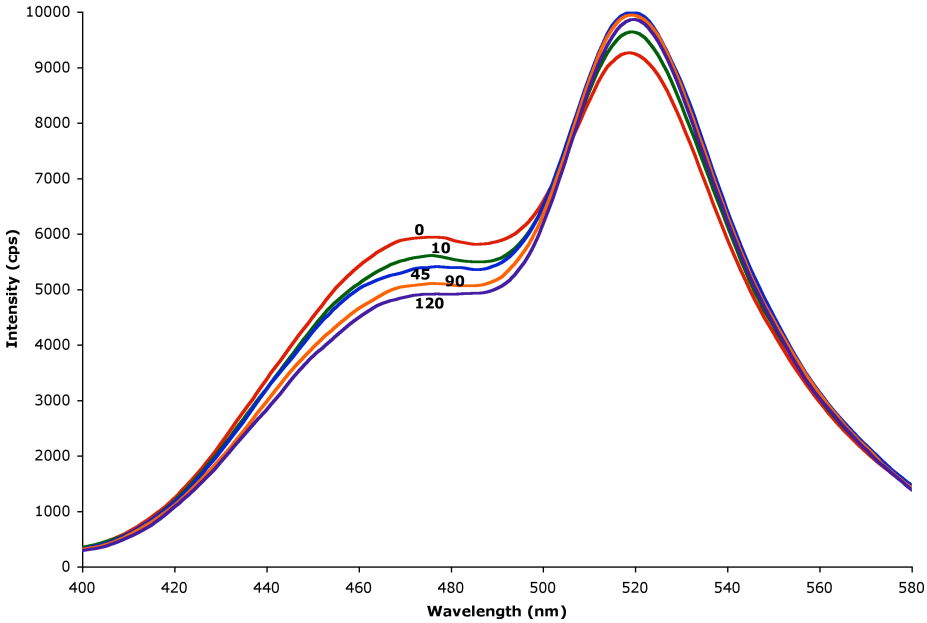

FIGURE 3: Changing KaiC Fluorescence Spectra due to FRET

Figure 3 Caption: Emission spectra from mixture of 1.73 μM AEDANS-KaiC and 1.73 μM AF-KaiC excited at 336 nm. Curves were taken at timepoints as labeled. Emission at ~480 nm is dominated by AEDANS-KaiC, as time-dependent quenching is evident. The peak at 520 nm on the other hand, is dominated by AF-KaiC, and its sensitization by donor AEDANS is evident, albeit to a lesser extent.

3.2 Observed FRET Behavior of Selected KaiC Mutants

Two mutants of KaiC were expressed and labeled in an identical manner as that of wild-type KaiC. The hypophosphorylated state of KaiC is mimicked by the double-alanine substitution S431A T432A "AA" mutant that replaces both phosphorylation sites with unphosphorylatable alanines. This mutant has been shown to form stable hexamers in solution and is capable of dampening the *in vitro* Kai oscillator (13), although previous coprecipitation assays have indicated that it does not shuffle measureably with other KaiC hexamers (11). KaiC wild-type itself dephosphorylates over a period of twelve hours in the absence of KaiA, although it retains some ATP turnover capability (14). The second mutant is that of an A-loop point mutant which disrupts a key hydrogen bond present in the loop displaced by KaiA. By disrupting this bond, KaiA's presence is simulated continuously, resulting in a stable ~100% hyperphosphorylated KaiC variant.

Figure 4 depicts a comparison of FRET ratios for these two mutants in addition to wild type over a period of six hours. Plotted is the normalized average efficiency of energy transfer in the KaiC population, calculated according to equation 3.2.1 according to the formalism of Clegg (15):

$$3.2.1: \quad E = 1 - (F_{DA}/F_D)$$

where E is the energy transfer efficiency, F_D is the fluorescence intensity of the donor alone, and F_{DA} is the fluorescence intensity of the donor + acceptor at equilibrium. If the exchange of subunits is sufficiently slow, the fluorescence signal at t_0 may be substituted for F_D (FRET intensity of donor only) and intensity at each subsequent time point directly used for F_{DA} (FRET intensity of donor + acceptor). As FRET efficiency at

equilibrium is dependent upon donor/acceptor labeling efficiency, all data points in the graph are divided by their final values at equilibrium. All of the KaiC variants used in this study reach equilibrium by approximately six hours, which correspond well with previous results using this technique on wild type KaiC (7). As a control, samples containing only donor-labeled KaiC WT were mixed with unlabeled KaiC. FRET efficiency remained unchanged at 0% (data not shown), indicating that transfer seen in samples containing both donor and acceptor was due solely to acceptor quenching.

Although previous coprecipitation assays both with and without the presence of KaiA failed to uncover evidence of KaiC–AA shuffling (11), our results indicate that this mutant shuffles readily. Coprecipitation investigations of the KaiC-AA mutant required the use of a mixture of tagged and untagged monomers which, after exchange, enabled the entire population to be precipitated. As our data indicates the KaiC-AA mutant hexamers to be somewhat weaker (i.e shuffle more rapidly) than wild-type, this might explain the inability to observe shuffling through a pulldown-type assay.

Furthermore, our findings demonstrate significant FRET efficiency at equilibrium, indicating a relatively small distance between monomers. These results conflict with previous synchronization models suggesting that KaiC subunits exist primarily as monomers in solution when either dephosphorylated or hyperphosphorylated (10). Our findings do correlate well with other studies showing that KaiC forms exclusively hexamers in solution (11).

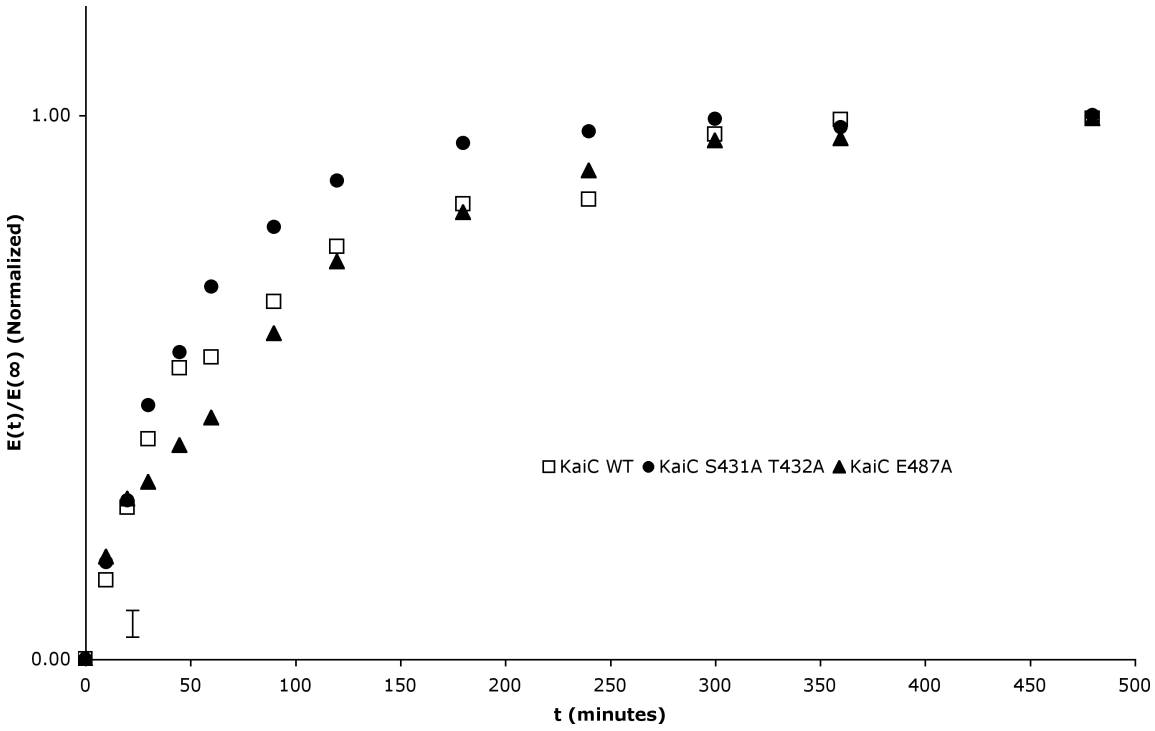

FIGURE 4: Time Dependent FRET Behavior of Three KaiC Species

Figure 4 Caption: Exchange profiles of KaiC WT (open squares) KaiC S431AT432A (closed circles) and KaiC E487A (closed triangles) are shown as a function of time. Each variant profile contains averaged data from two to three separate experiments. Y-axis is normalized energy transfer efficiency. Average error bar is located in the bottom-left corner.

3.3 Statistical Fitting of FRET Data

The rate of subunit exchange can be extracted by fitting data generated from the experiment outlined in section 3.2 with an exponential function employed previously for subunit exchange in alpha-crystallin (16):

$$3.3.1: \quad F(t) = C_1 + C_2 e^{-kt}$$

Here, F(t) is the observed fluorescence signal, C_1 is fluorescence intensity at $t=\infty$, $C_2 = 1-C_1$. k is the shuffling rate constant, and t is time. This equation can be modified for use directly with FRET efficiencies "E," (as calculated by equations 3.2.1), yielding

$$3.3.2: \quad E = E_\infty * (1 - e^{-kt})$$

where E_∞ is the FRET efficiency at equilibrium. The rate constant k was determined by nonlinear regression fitting of the data using the Prism 4 (GraphPad Software, San Diego CA) statistical package program. Using this methodology, the following rate constants were determined: WT KaiC: 0.70±0.12 hr^{-1}, KaiC E487A: 0.66±0.18 hr^{-1}, and KaiC-AA: 1.08±0.10 hr^{-1}. Each KaiC rate constant was fit to data points from at least two separate experiments. Shown in figure 5 are three separate FRET experiments of KaiC WT, each performed on a unique batch of purified KaiC. The shaded lines are best-fit solutions to the data generated from the putative shuffling model with rate constants of 0.80, 0.59, and 0.71 hr^{-1}.

These rate constants indicate an average half life ($t_{1/2}$) of KaiC subunits to be approximately 1 hour for KaiC WT and E487A, and 40 minutes for that of KaiC-AA. Although KaiC WT and KaiC AA show the most significant difference, of primary interest are the similar rates of exchange demonstrated by dephosphorylated WT and the

hyperphosphorylated point mutant. Previous studies have noted the putative existence of several new trans-subunit contacts to the phosphoryl group at T432 (3). These interactions make up a small percentage of the total number of existing contacts that line the highly hydrophilic interface between KaiC monomers. S431, when phosphorylated, contributes even fewer intersubunit contacts than phosphorylated T432. Experimentally, ATP is required to maintain stability of KaiC in solution. Given ATP's presence at the interfaces of KaiC, their contribution to shuffling kinetics is potentially large. Indeed, as the S431A T432A mutant retains the ability to bind ATP, and E487A is perpetually phosphorylated, it is possible that ATP hydrolysis itself may regulate shuffling directly.

These analyses were done in the absence of other clock component proteins, so the effect of KaiA and KaiB on shuffling is unknown. In the previous study using FRET to examine KaiC shuffling (7), KaiA was added at the beginning of the experiment. As KaiA induces the autophosphorylation of KaiC over a similar timeframe as that of KaiC exchange itself, KaiA's effect on phosphorylated KaiC shuffling cannot be ascertained by the experimental protocol used in Mori et al. (7). Likewise, their addition of KaiB to dephosphorylated KaiC is inconclusive, as KaiB has been shown to only bind when KaiC is phosphorylated at S431 (5).

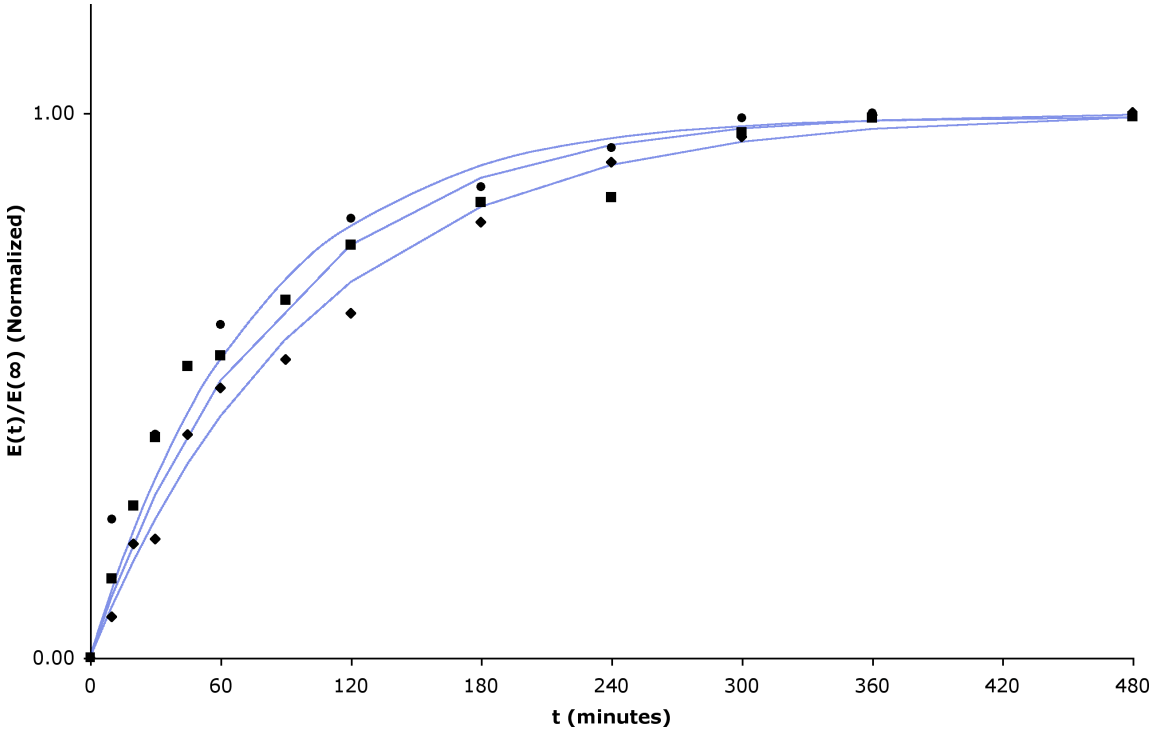

FIGURE 5: Shuffling Model Fit to Wild-Type Kaic Exchange Data

Figure 5 Caption: Plot showing results from three separate runs of KaiC WT, each fitted by nonlinear regression with equation 3.3.2 resulting in shuffling constants of 0.59 hr^{-1}, 0.66 hr^{-1}, and 0.78 hr^{-1}.Y axis is normalized energy transfer efficiency, generated by dividing energy transfer efficiency at time *t* by energy transfer efficiency at equilibrium.

3.4 Effects of Labeling Efficiency on Shuffling Constants

The presence of unlabeled or multiply labeled KaiC monomers in the shuffling population has the potential to generate a significant effect on observed FRET behavior. Furthermore, the fitting of shuffling rates to data assumes that labeling does not affect oligiomerization in any way and that all shuffling events are random. Incomplete labeling also assumes that the population of labeled subunits is large enough to accurately describe the distribution of the total population.

To investigate the effect of labeling efficiency on shuffling rate constants, k (the fitted shuffling rate constant) was plotted as a function of acceptor and donor labeling efficiencies, as seen in figure 6. Over the fairly narrow range of typical labeling efficiencies, no significant effect on shuffling values is apparent, for either the donor (5A) or acceptor (5B). To further verify these empirical results with a wider range of labeling percentages, a computer simulation of a set of monomers arranged into hexamers was designed. A dataset consisting of 10,000 hexamers, roughly the amount found in living *S. elongatus* cyanobacteria was generated containing a varying initial ratio of donor and acceptor labeled monomers. A shuffling subroutine was then used to randomly exchange subunits between hexamers, and at specific intervals a donor/acceptor proximity value analogous to FRET was generated. Results from this simulation (data not shown) revealed that labeling efficiency had no effect on the fitted shuffling rate constants.

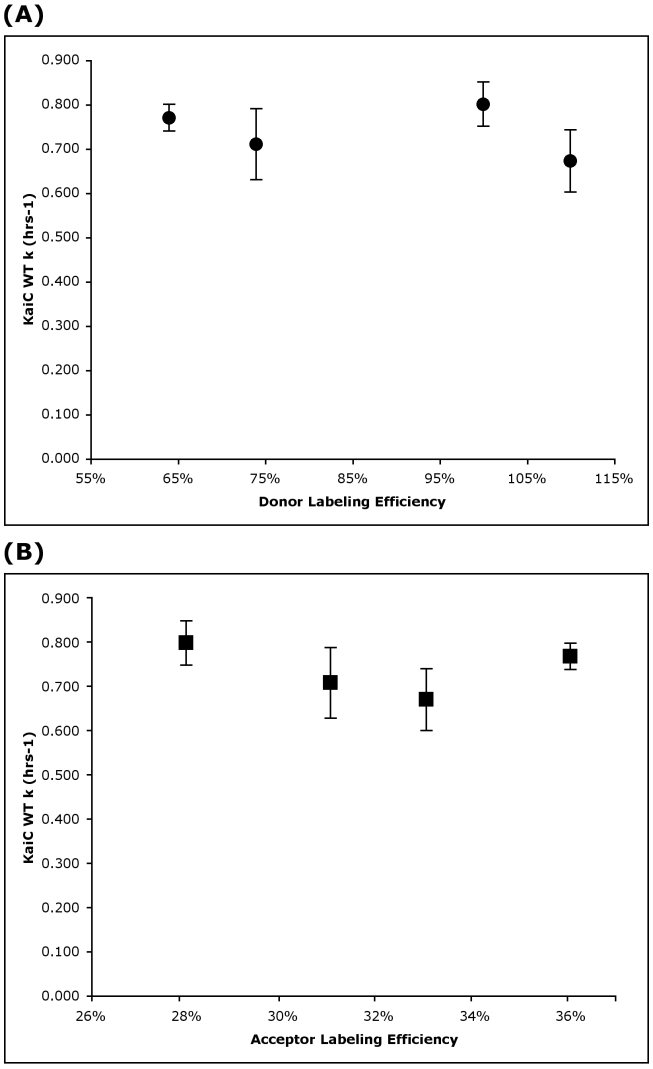

FIGURE 6: Effect of Labeling Efficiencies Upon Observed K

Figure 6 caption: Shuffling kinetic values as a function of donor IAEDANS (A) and acceptor 6-IAF (B) labeling efficiency. Error bars are standard error given from the statistical fit model.

3.5 Enzymatic Behavior of Labeled Kaic Hexamers

Because the C348 lies near the CII domain ATP-binding and phosphorylation sites, fluorophore labeling at this site may have interfered with enzymatic behavior. To compare the phosphorylation kinetics of labeled WT KaiC with that of unlabeled KaiC, samples of IAF-labeled KaiC as well as unlabeled KaiC were incubated with KaiA, and samples were taken out every hour. The phosphorylation kinetics and phosphorylation percentage of labeled KaiC were indistinguishable from those of unlabeled KaiC. Additionally, aliquots of labeled KaiC-AA and KaiC E487A incubated concurrently were taken. These samples were resolved via SDS-PAGE, which was then illuminated with long-wave UV light to observe the presence of fluorescent probes, and then stained with Coomassie brilliant blue.

As seen in Figure 7, KaiC retained its ability to be phosphorylated by KaiA over a period of four hours, and showed no preference of autophosphorylation towards either the labeled or unlabeled KaiC species. KaiC-AA fluorescence displayed a single band corresponding to dephosphorylated KaiC, whereas KaiC E487A appeared as a strong top band and weak dephosphorylated band, demonstrating the >90% phosphorylation characteristic of this mutant. This band did not decrease in intensity even after 12 hours incubation at 30°C. Unlabeled KaiC showed no fluorescence signal when irradiated.

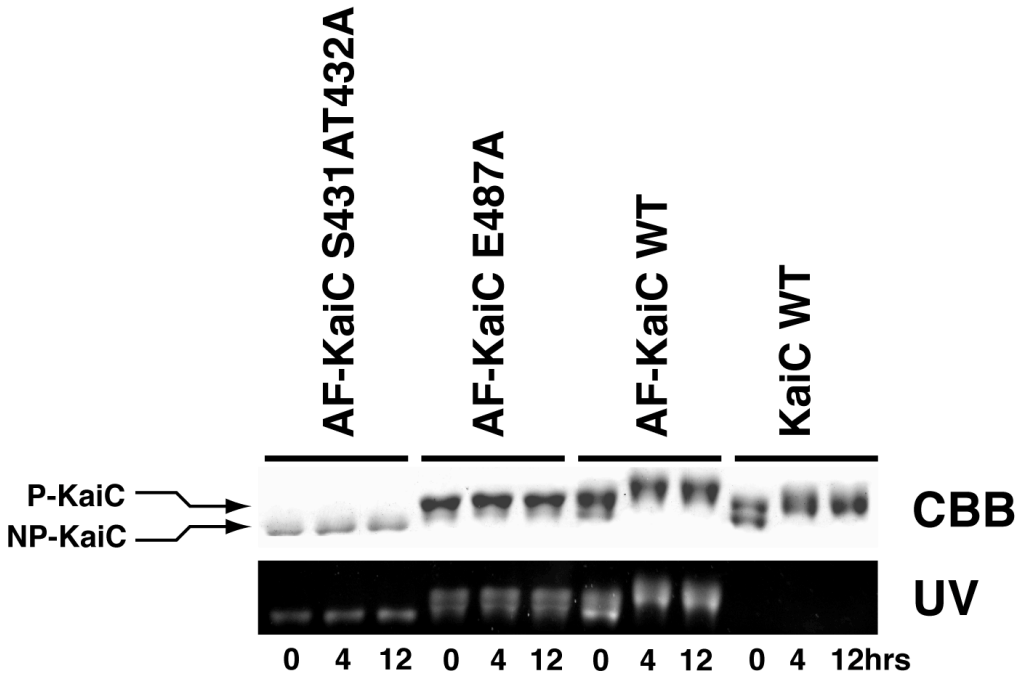

FIGURE 7: Phosphorylation Kinetics and Stability of Kaic Variants

Figure 7 Caption: 7.5% SDS-PAGE gel visualized under UV light and subsequent visible light illumination after Coomassie brilliant blue (CBB) staining. Each KaiC variant is represented by three time points corresponding to 0, 4 and 12 hours after addition of KaiA. Labeled KaiC species use acetamidofluroescein (AF) as fluorophore. Upper bands labeled “P-KaiC” correspond to hyperphosphorylated KaiC, while bands labeled with “NP-KaiC” correspond to dephosphorylated KaiC.

4. SUMMARY AND CONCLUSIONS

4.1 Summary

The Kai *in vitro* protein oscillator is unique in its elegant and concise construction, in addition to its ability to automatically correct for temperature and varying concentrations of its components. Its ability to maintain a constant rhythm over a span of several days is nothing short of amazing. Although high-resolution structures of these three components are known, their interplay and subsequent behavior remain a tantalizing puzzle.

As synchronization of the individual KaiC clocks has been identified as a crucial aspect to precise timekeeping, a significant amount of research has been directed at elucidating its mechanism. Our study indicates that the entire population of cellular timekeepers potentially stay synchronized not through periodic bursts of rapid shuffling, but through a gradual diffusion of monomers from complex to complex. As the rate of exchange is more than rapid enough to occur several times during each 24-hour oscillation, potentially disastrous timekeeper drift is effectively prevented.

4.2 Future Research

The current paradigm of the Kai oscillator is that a strictly-regimented program of phosphorylation events is driven, and itself drives recruitment and binding of clock components KaiA and KaiB. As this study did not investigate the effects of either

component on the mutants used, their effects (if any) on shuffling of specific phosphoforms has yet to be uncovered.

Of particular future interest are the behavior of specific phosphomimetic mutants, and the corresponding interaction of the other clock components. Future fluorescence spectroscopy-based work on phosphomimetic KaiC mutants complexed with KaiA and KaiC can lend more light on not only shuffling dynamics, but on potential structural rearrangments or perturbations as well.

Furthermore, a model developed in this lab places special importance on the KaiA-binding “A-loop” of KaiC which has several residues that apparently create trans-subunit hydrogen bonds. A comparison of the hyperphosphorylated truncation mutant KaiC 487 in which these interactions are abolished with the hyperphosphorylated point mutant KaiC E487A used in this study might shed light on the nature of these contacts.

REFERENCES

1.Ye, S., Vakonakis, I., Ioerger, T.R., LiWang, A.C. and Sacchettini, J.C. (2004) Crystal structure of circadian clock protein KaiA from Synechococcus elongatus, *J.Biol.Chem. 279*, 20511-20518,

2. Iwase, R., Imada, K., Hayashi, F., Uzumaki, T., Morishita, M., Onai, K., Furukawa, Y., Namba, K. and Ishiura, M. (2005) Functionally important substructures of circadian clock protein KaiB in a unique tetramer complex. *J. Biol. Chem. 280,* 43141-43149.

3. Pattanayek, R., Wang, J., Mori, T., Xu, Y., Johnson, C. H. and Egli, M. (2004) Visualizing a circadian clock protein: crystal structure of KaiC and functional insights *Mol. Cell* 15, 375–388

4. Iwasaki H, Taniguchi Y, Ishiura M, and Kondo, T. (1999) Physical interactions among circadian clock proteins KaiA, KaiB, and KaiC in cyanobacteria, *EMBO 18*, 1137-1145

5. Nishiwaki, T., Satomi, Y., Kitayama, Y., Terauchi, K., Kiyohara, R., Takao, T. and Kondo, T. (2007) A sequential program of dual phosphorylation of KaiC as a basis for circadian rhythm in cyanobacteria, *EMBO 26,* 4029-4037.

6. Rust, M. J., Markson, J. S., Lane, W. S., Fisher, D. S. and O'Shea, E. K. (2007) Ordered Phosphorylation Governs oscillation of a three-protein circadian clock. *Science* (in press)

7. Mori, T., Williams, D. R., Byrne, M. O., Qin, X., Egli, M., Mchaourab, H. S., Stewart, P. L. and Johnson, C. H. (2007) Elucidating the ticking of an *in vitro* circadian clockwork, *PLoS Biology 5,* 841-853.

8. Nakajima M, Imai K, Ito H, Nishiwaki T, Murayama Y. et al. (2005) Reconstitution of circadian oscillation of cyanobacterial KaiC phosphorylation *in vitro*, *Science 308*, 414-415.

9. Emberly E. and Wingreen, N. S. (2006) Hourglass model for a protein-based circadian oscillator, *Phys. Rev. Lett. 96,* 38303

10. Yoda, M., Eguchi, K., Terada, T. and Sasai, M. (2007) Monomer-shuffling and allosteric transition in KaiC circadian oscillation, *PLoS ONE 5,* 408.

11. Kageyama, H., Nishiwaki, T., Masato, N., Iwasaki H., Oyama, T. and Kondo, T. (2006) Cyanobacterial circadian pacemaker: Kai protein complex dynamics in the KaiC phosphorylation cycle *in vitro*. *Molecular Cell 23*, 161-171.

12. Williams, S., Vakonakis, I., Golden, S. and LiWang, A. (2002) Structure and function of circadian clock protein KaiA of Synechococcus elongatus: A potential clock input mechanism, *PNAS 99,*15357-15362.

13. Ito H., Kageyama H., Mutsuda M., Nakajima M., Oyama T. and Kondo T. (2007) Autonomous synchronization of the circadian KaiC phosphorylation rhythm. *Nat. Struct. Mol. Biol.* In press.

14. Nishiwaki, T., Iwasaki, H., Ishiura, M, and Kondo, T. (2000) Nucleotide binding and autophosphorylation of the clock protein KaiC as a circadian timing process of cyanobacteria. *PNAS 97*, 495-499.

15: Clegg, R. M. (1995) Fluorescence resonance energy transfer, *Curr. Opin. Chem. Biol. 6,* 103-110.

16. Bova, M. P., Ding, L., Horwitz, J. and Fung, B. K. (1997) Subunit exchange of αA-Crystallin, *J. Biol. Chem. 272,* 29511-29517.

APPENDIX A

SOURCE CODE OF SHUFFLING SIMULATION PROGRAM

Mklist.pl:

(Start of mklist.pl)

```
#!/usr/bin/perl

# This script creates a set of oligiomers fulfilling input parameters,
# suitable for input to shuffle

# get the number of arguments
$numargs = $#ARGV +1;
$argcounter = 0;

# go through the arguments and get their values
while ( $argcounter < $numargs )
{
 if ( $ARGV[$argcounter] eq "-h" ){
  &print_help;
 }
 elsif( $ARGV[$argcounter] eq "-n" ){
  $num_constructs = $ARGV[$argcounter +1];
 }
 elsif( $ARGV[$argcounter] eq "-m" ){
  $num_subunits = $ARGV[$argcounter +1];
 }
 elsif( $ARGV[$argcounter] eq "-a" ){
  $acceptor_label_percent = $ARGV[$argcounter +1];
 }
 elsif( $ARGV[$argcounter] eq "-d" ){
  $donor_label_percent = $ARGV[$argcounter +1];
 }
 elsif( $ARGV[$argcounter] eq "-out" ){
  $output_file = $ARGV[$argcounter +1];
 }

 $argcounter++;
}

if ($num_constructs < 10){
 die ("Number of constructs set too low (< 1). Aborting...\n");
}
if ($num_subunits < 2){
 die ("Number of subunits set too low (< 2). Aborting...\n");
}
if ($acceptor_label_percent > 1){
 die ("Acceptor label percentage greater than 1 (100%).
Aborting...\n");
}
```

```
if ($donor_label_percent > 1){
 die ("Donor label percentage greater than 1 (100%). Aborting...\n");
}
if (length($output_file) < 1){
 die ("Output file not specified. Aborting...\n");
}

# use current time for random number seed
srand(time() ^($$ + ($$ <<15)));

@constructs = ();

for($i = 0; $i < $num_constructs; $i++)
{
 $constructs[$i] = ''; # initialize

 for($j = 0; $j < $num_subunits; $j++)
 {
  # even constructs = donors
  # odd constructs = acceptors
  if (($i % 2) == 0)
  {
   if ((rand() + $donor_label_percent) > 1){
    $constructs[$i] = $constructs[$i] . 'D:';
   }
   else{
    $constructs[$i] = $constructs[$i] . 'N:';
   }

  }
  else
  {
   if ((rand() + $acceptor_label_percent) > 1){
    $constructs[$i] = $constructs[$i] . 'A:';
   }
   else{
    $constructs[$i] = $constructs[$i] . 'N:';
   }
  }
 }

 # pop off the last trailing semicolon
 chop($constructs[$i]);
}

open( OUTPUT, ">$output_file" ) || die "Couldn't open file for writing:
$!\n";

# write results to file
for($i = 0; $i < $num_constructs; $i++)
{
 print OUTPUT $constructs[$i] . "\n";
}

close( OUTPUT );
```

```
sub print_help
{
 print "-----------------------------------------------------------------
-----------------\n";
 print "   Help for mklist.pl\n";
 print "   (C)2007 Elihu Ihms, elihuihms(at)gmail.com\n";
 print "   Version: 1.0\n";
 print "-----------------------------------------------------------------
-----------------\n";
 print "mklist.pl creates an ASCII file containing a number of
multimers, each\n";
 print "consisting of a number of colon-delimited monomers, suitable
for input to either\n";
 print "move.pl or getfret.pl (although getfret.pl should always give a
FRET value of 0.\n";
 print "Both donor and acceptor labeling percentage may be
specified.\n";
 print "\n";
 print "Valid usage is: mklist.pl -h -n <X> -m <Y> -d <D> -a <A> -out
<OUTPUT FILE>\n";
 print "\n";
 print "Options:\n";
 print "-n <X>: Build X number of multimers. Must be > 1.\n";
 print "-m <Y>: Each multimer should be composed of <Y> monomers. Must
be > 1.\n";
 print "-d <D>: Donor labeling efficiency (0= 0%, 1=100%)\n";
 print "-a <A>: Acceptor labeling efficiency (0= 0%, 1=100%)\n";
 print "-h: Prints this help message.\n";
 print "-out <OUTPUT FILE>: File name to write generated list to.\n";
 print "\n";
 print "-----------------------------------------------------------------
-----------------\n";
}
```

(End of mklist.pl)

Move.pl:

(Start of move.pl)

```
#!/usr/bin/perl

# This script shuffles monomers between two randomly selected pairs

# get the number of arguments
$numargs = $#ARGV +1;
$argcounter = 0;

# default number of subunits
$num_subunits = 2;

# go through the arguments and get their values
while ( $argcounter < $numargs )
{
 if ( $ARGV[$argcounter] eq "-h" ){
  &print_help;
 }
 elsif( $ARGV[$argcounter] eq "-n" ){
  $shuffling_iterations  = $ARGV[$argcounter +1];
 }
 elsif( $ARGV[$argcounter] eq "-m" ){
  $num_subunits = $ARGV[$argcounter +1];
 }
 elsif( $ARGV[$argcounter] eq "-k" ){
  $shuffling_constant = $ARGV[$argcounter +1];
 }
 elsif( $ARGV[$argcounter] eq "-in" ){
  $input_file = $ARGV[$argcounter +1];
 }
 elsif( $ARGV[$argcounter] eq "-out" ){
  $output_file = $ARGV[$argcounter +1];
 }

 $argcounter++;
}

if ($shuffling_iterations < 1){
 die ("Number of iterations set too low (< 1). Aborting...\n");
}
if ($num_subunits < 2){
 die ("Number of subunits is set too low (< 2). Aborting...\n");
}
if ($shuffling_constant < 0){
 die ("Shuffling constant set too low (< 0). Aborting...\n");
}
if (length($input_file) < 1){
 die ("Input file not specified. Aborting...\n");
}
if (length($output_file) < 1){
 die ("Output file not specified. Aborting...\n");
}
```

```
# argument fixes
# this needs to be done because we start counting at 0
$num_subunits = $num_subunits -1;

# open the specified file
open(DATAFILE, $input_file) || die "Can't open $input_file: $!\n";

# read the entire file into @log_file_data
@constructs = <DATAFILE>;
$num_constructs = $#constructs;

close( DATAFILE );

# use current time for random number seed
srand(time() ^($$ + ($$ <<15)));

for($i = 0; $i < $shuffling_iterations; $i++)
{
 if (round(rand() * ($num_constructs * $shuffling_constant)) >= 1)
 {
  # get the random polymer positions
  $A_construct_pos = round($num_constructs * rand());
  $B_construct_pos = round($num_constructs * rand());

  # get the monomer positions
  $A_subunit_pos = round($num_subunits * rand());
  $B_subunit_pos = round($num_subunits * rand());
  if ($A_subunit_pos != $B_subunit_pos)
  {
   # remove trailing carriage return if necessary
   chomp($constructs[$A_construct_pos]);
   chomp($constructs[$B_construct_pos]);

   # split the monomer identities of the polymer into an array
   @A_construct = split(/:/,$constructs[$A_construct_pos]);
   @B_construct = split(/:/,$constructs[$B_construct_pos]);

   # get the monomers to be exchanged
   $A_subunit = $A_construct[$A_subunit_pos];
   $B_subunit = $B_construct[$B_subunit_pos];

   # now exchange the monomers
   $B_construct[$B_subunit_pos] = $A_subunit;
   $A_construct[$A_subunit_pos] = $B_subunit;

   # now save the structures to the original array
   $constructs[$A_construct_pos] = join(':',@A_construct);
   $constructs[$B_construct_pos] = join(':',@B_construct);
  }

#  print($i . "-Give: " . $A_construct_pos . "x" . $A_subunit_pos .
":'" . $A_subunit . "'\n");
#  print($i . "-Take: " . $B_construct_pos . "x" . $B_subunit_pos .
":'" . $B_subunit . "'\n");
```

```
 }
}

# write shuffled subunits to file
open(OUTPUT, ">$output_file") || die "Can't open $output_file: $!\n";

for($i = 0; $i < $num_constructs; $i++)
{
 chomp($constructs[$i]);
 print OUTPUT $constructs[$i] . "\n";
}

close( OUTPUT );

sub round
{
 my($number) = shift;
    return int($number + .5 * ($number <=> 0));
}
```

(End of move.pl)

Source code for getfret.pl:

(Start of getfret.pl)

```
#!/usr/bin/perl

# This script creates an arbitrary number based upon number of touching
"donor"
# "acceptor" pairs from a list

# get the number of arguments
$numargs = $#ARGV +1;
$argcounter = 0;

# default argument values
$bidirectional = 0;

# go through the arguments and get their values
while ( $argcounter < $numargs )
{
 if ( $ARGV[$argcounter] eq "-h" ){
  &print_help;
 }
 elsif( $ARGV[$argcounter] eq "-in" ){
  $input_file = $ARGV[$argcounter +1];
 }
 elsif( $ARGV[$argcounter] eq "-b" ){
  $bidirectional = 1;
 }

 $argcounter++;
}

if (length($input_file) < 1){
 die ("Input file not specified. Aborting...\n");
}

# open the specified file
open(DATAFILE, $input_file) || die "Can't open $input_file: $!\n";

# read the entire file into @log_file_data
@constructs = <DATAFILE>;

close(DATAFILE);

$fret_intensity = 0;
foreach(@constructs)
{
 # get rid of trailing newline
 chomp($_);
 # split into array
 @subunits = split(/:/,$_);

 # go through subunits and determine if D/A pair touch
 for ($i = 0; $i <= $#subunits; $i++)
```

```
 {
  if ($i != $#subunits){
   if (($subunits[$i] eq "D") && ($subunits[$i +1] eq "A")){
    $fret_intensity++;
   }
  }
  elsif($#subunits > 1){
   # check for wraparound, but not for dimers, which would be
bidirectional. natch
   if (($subunits[$i] eq "D") && ($subunits[0] eq "A")){
    $fret_intensity++;
   }
  }
 }
 if ($bidirectional)
 {
  # go through subunits and determine if A/D pair touch
  for ($i = 0; $i <= $#subunits; $i++)
  {
   if ($i != $#subunits){
    if (($subunits[$i] eq "A") && ($subunits[$i +1] eq "D")){
     $fret_intensity++;
    }
   }
   elsif($#subunits > 1){
    # check for wraparound, but not for dimers, which would be
bidirectional. natch
    if (($subunits[$i] eq "A") && ($subunits[0] eq "D")){
     $fret_intensity++;
    }
   }
  }
 }

}

print $fret_intensity . "\n";
```

(End of getfret.pl)

Source code for shuffle.pl:

(Start of shuffle.pl)

```
#!/usr/bin/perl

# This is a wrapper script that can execute an entire shuffle
experiment

# get the number of arguments
$numargs = $#ARGV +1;
$argcounter = 0;

$quiet_mode = 0;

# go through the arguments and get their values
while ( $argcounter < $numargs )
{
 if ( $ARGV[$argcounter] eq "-h" ){
  &print_help;
 }
 elsif( $ARGV[$argcounter] eq "-in" ){
  $input_file = $ARGV[$argcounter +1];
 }
 elsif( $ARGV[$argcounter] eq "-q" ){
  $quiet_mode = 1;
 }

 $argcounter++;
}

if (length($input_file) < 1){
 die ("Parameter file not specified. Aborting...\n");
}

# open the specified file
open(PARAMFILE, $input_file) || die "Can't open $input_file: $!\n";

# read the entire file into @log_file_data
@parameters = <PARAMFILE>;

# close the parameter file
close(PARAMFILE);

# go through the parameters line by line
for ($i = 0; $i <= $#parameters; $i++)
{
 # get line and remove leading/trailing whitespace
 $file_line = $parameters[ $i ];
 chomp( $file_line );

 #$file_line =~ /^(-?\d+\.?\d*)/;
 # split data field apart with regex

 $file_line =~ /^(\S*) = (\S*)/;
```

```
 if ((length($1) > 0) && (length($2) < 1))
 {
  die("Parameter '" . $1 . "' is not set.\n");
 }
 elsif(length($2) > 0)
 {
  if ($1 eq 'mklist_loc'){
   $mklist_script_loc = $2;
  }
  elsif($1 eq 'getfret_loc'){
   $getfret_script_loc = $2;
  }
  elsif($1 eq 'shuffle_loc'){
   $shuffle_script_loc = $2;
  }
  elsif($1 eq 'output_file'){
   $output_file = $2;
  }
  elsif($1 eq 'output_stamp'){
   $iteration_file_stamp = $2;
  }
  elsif($1 eq 'num_constructs'){
   $num_constructs = $2;
  }
  elsif($1 eq 'num_subunits'){
   $num_subunits = $2;
  }
  elsif($1 eq 'shuffle_k'){
   $shuffle_k = $2;
  }
  elsif($1 eq 'num_iterations'){
   $num_iterations = $2;
  }
  elsif($1 eq 'sample_iterations'){
   @sample_iterations = split(/,/,$2);
   # sort the sample iterations array in ascending order
   @sample_iterations = sort {$a <=> $b} (@sample_iterations);
  }
  elsif($1 eq 'donor_label'){
   $donor_label_percent = $2;
  }
  elsif($1 eq 'acceptor_label'){
   $acceptor_label_percent = $2;
  }
  elsif($1 eq 'bidirectional_fret'){
   $bidirectional_fret = $2;
  }
 }
}

# write starting parameters to output file

open(OUTPUT, ">$output_file") || die "Can't open $output_file: $!\n";

print OUTPUT "Experiment started:\t" . localtime() . "\n\n";
```

```
print OUTPUT "Number of multimers:\t" . $num_constructs . "\n";
print OUTPUT "Multimer polymeric state:\t" . $num_subunits . "\n";
print OUTPUT "Shuffling k value:\t" . $shuffle_k . "\n";
print OUTPUT "Donor labeling:\t\t" . $donor_label_percent . "\n";
print OUTPUT "Acceptor labeling:\t" . $acceptor_label_percent . "\n";
print OUTPUT "Bidirectional fret:\t";
if ($bidirectional_fret > 0){
 print OUTPUT "yes\n";
}
else{
 print OUTPUT "no\n";
}
print OUTPUT "Number of iterations:\t" . $num_iterations . "\n";
print OUTPUT "Iteration samples:\t" . join(',',@sample_iterations) .
"\n";

# setup the required shell commands
$example_mklist_command = "mklist.pl -n " . $num_constructs . " -m " .
$num_subunits . " -d " . $donor_label_percent . " -a " .
$acceptor_label_percent . " -out " . $iteration_file_stamp . "0.txt";
$example_shuffle_command = "move.pl -n <x> -m " . $num_subunits . " -k
" . $shuffle_k . " -in " . $iteration_file_stamp . "<y>.txt -out " .
$iteration_file_stamp . "<z>.txt";
$example_getfret_command = "getfret.pl -in " . $iteration_file_stamp .
"<x> ";
if ($bidirectional_fret > 0){
 $example_getfret_command = $example_getfret_command . "-b";
}

print OUTPUT "Example mklist.pl command:\t" . $example_mklist_command .
"\n";
print OUTPUT "Example getfret.pl command:\t" . $example_getfret_command
. "\n";
print OUTPUT "Example move.pl command:\t" . $example_shuffle_command .
"\n";

# run the mklist script
if ($quiet_mode == 0){
 print STDERR "Creating sample list...";
}

$mklist_command = $mklist_script_loc . " -n " . $num_constructs . " -m
" . $num_subunits . " -d " . $donor_label_percent . " -a " .
$acceptor_label_percent . " -out " . $iteration_file_stamp . "0.txt";
$mklist_return = qx($mklist_command);

if (length($mklist_return) > 0){
 print OUTPUT "Error creating list: " . $mklist_return . "\n";
 die("Error creating list: " . $mklist_return . "\n");
}

print OUTPUT "\nIteration FRET values:\n";
if ($quiet_mode == 0){
 print STDERR "Done.\n";
}
```

```
# run the shuffle subroutine and collect FRET values
foreach($i = 0; $i <= $#sample_iterations; $i++)
{
 if ($quiet_mode == 0){
  print STDERR "Getting FRET for iteration " . $sample_iterations[$i] .
"...";
 }
 $getfret_command = $getfret_script_loc . " -in " .
$iteration_file_stamp . $sample_iterations[$i] . ".txt";
 $getfret_return = qx($getfret_command);
 if ($quiet_mode == 0){
  print STDERR "Value is " . $getfret_return;
 }
 print OUTPUT $sample_iterations[$i] . "\t" . $getfret_return;

 if ($i == $#sample_iterations){
  last;
 }

 # shuffle
 if ($quiet_mode == 0){
  print STDERR "Shuffling iterations " . $sample_iterations[$i] . "
through " . $sample_iterations[$i +1] . "...";
 }
 $shuffle_iterations = $sample_iterations[$i +1] -
$sample_iterations[$i];
 $shuffle_command = $shuffle_script_loc . " -n " . $shuffle_iterations
. " -m " . $num_subunits . " -k " . $shuffle_k . " -in " .
$iteration_file_stamp . $sample_iterations[$i] . ".txt -out " .
$iteration_file_stamp . $sample_iterations[$i +1] . ".txt";
 $shuffle_return = qx($shuffle_command);
 if ($quiet_mode == 0){
  print STDERR "Done.\n";
 }
}
if ($quiet_mode == 0){
 print STDERR "Finished with experiment.\n";
}
print OUTPUT "\nFinished with experiment " . localtime() . "\n";

close(OUTPUT);

sub print_help
{
 print "-----------------------------------------------------------------
-----------------\n";
 print "   Help for shuffle.pl\n";
 print "   (C)2007 Elihu Ihms, elihuihms(at)gmail.com\n";
 print "   Version: 1.0\n";
 print "-----------------------------------------------------------------
-----------------\n";
 print "shuffle.pl is essentially a wrapper script for the rest of the
shuffle suite,\n";
```

```
 print "consisting of mklist.pl, move.pl, and getfret.pl. shuffle.pl
only takes a single\n";
 print "file, consisting of the parameters for the experiment, and
outputs the results\n";
 print "as well as the progress.\n";
 print "\n";
 print "Valid usage is: shuffle.pl -hq -in <PARAMS FILE> -out <RESULT
FILE>\n";
 print "\n";
 print "Options:\n";
 print "-q: Runs shuffle.pl in quite mode, suppressing output to
STDERR.";
 print "-h: Prints this help message.\n";
 print "-out <PARAMS FILE>: Parameter file to read configuration values
from.\n"
 print "-out <RESULT FILE>: Output file to which results and progress
are written.\n";
 print "\n";
 print "-------------------------------------------------------------
-----------------\n";
}
```

(End of shuffle.pl)

Example input file:

(Start of example input file)

```
#######################################
# Sample Input File for Shuffle Suite #
#######################################
# Conventions:
# Comments are preceded by a pound (#) sign
# Parameters followed by a space, equation (=) sign, and then the value
# (No spaces in value, please)
#
#######################################
# Execution and Environment Parameters #
#######################################
#
# Makelist (mklist.pl) script location:
mklist_loc = ~/Science/Utilities/shuffle/mklist.pl
#
# GetFRET (getfret.pl) script location:
getfret_loc = ~/Science/Utilities/shuffle/getfret.pl
#
# Move (move.pl) script location:
shuffle_loc = ~/Science/Utilities/shuffle/move.pl
#
# Number of subunits for each multimer
num_subunits = 6
#
# Number of constructs to use in experiment
num_constructs = 10000
#
# Shuffling k value
shuffle_k = 1
#
# Iterations to take FRET samples (comma delimited)
sample_iterations =
0,15000,30000,50000,75000,100000,150000,200000,250000,300000,350000,400
000,450000,500000
#
# Labeling efficiency for donor monomers
donor_label = 0.75
#
# Labeling efficiency for acceptor monomers
acceptor_label = 0.75
#
# Allow bidirectional FRET? (0 = no, 1 = yes)
bidirectional_fret = 1
#
# String to prepend to each iteration file
output_stamp = sample_iteration_
#
# Output file
output_file = sample_output.txt
```

(End of example input file)

VITA

Name:	Elihu Carl Ihms
Address:	Biochemistry and Biophysics, MS 2128 TAMU, College Station, TX 77843
Email Address:	ihms@tamu.edu
Education:	B.S., Chemistry, Indiana Wesleyan University, 2004