Design Tools for Oscillator-based Computing Systems

Tianshi Wang[‡] and Jaijeet Roychowdhury
[*]Department of Electrical Engineering and Computer Science, The University of California, Berkeley, CA, USA
[‡]Contact author. Email: tianshi@berkeley.edu

ABSTRACT

Recently, general-purpose computing schemes have been proposed that use phase relationships to represent Boolean logic levels and employ self-sustaining nonlinear oscillators as latches and registers. Such phase-based systems have superior noise immunity relative to traditional level-encoded logic, hence are of interest for next-generation computing using nanodevices. However, the design of such systems poses special challenges for existing tools. We present a suite of techniques and tools that provide designers with efficient simulation and convenient visualization facilities at all stages of phase logic system design. We demonstrate our tools through a case study of the design of a phase logic finite state machine (FSM). We build this FSM and validate our design tools and processes against measurements. Our plan is to release our tools to the community in open source form.

1. INTRODUCTION

For decades, Moore's law [13] has been the driving force behind the growth of the semiconductor industry and the rapid progress of computing power. However, serious design and manufacturing roadblocks are starting to slow down this trend [22]. For example, noise and variability are having an ever-greater impact on system performance as transistors are further miniaturized; power consumption has also been of serious concern for many years. With such obstacles impeding progress in computing, there has been a search for alternative computing paradigms.

One broad direction being explored is alternatives to conventional planar silicon CMOS transistors, e.g., new device structures such as multigate transistors [10], new materials including graphene [14] and carbon nanotubes [16], and new devices in alternative domains such as optics and spintronics [2, 3]. In another direction, novel system-level designs are being explored to alleviate power consumption concerns, e.g., improved micro-architectures, advanced power management, multi-core chip designs with multi-threaded software [8], *etc.* Most explorations of alternatives have concentrated on the device and architectural levels.

Another direction, alternative low-level physical representations of abstract logic, has also attracted attention recently. In conventional computing systems, binary logic is typically encoded as two stable voltage levels; CMOS technology provides an excellent substrate for such level-based computation. However, an alternative low-level mechanism for encoding logic, using relative phase differences from a reference signal (REF), has been shown to hold considerable promise.

The notion of phase-encoded computing dates back to decades ago. In the 1950s, Eiichi Goto and John von Neumann proposed schemes for implementing computation with phase logic as alternatives to the vacuum tube technology then used in computers [19, 21, 9]. They devised resonant circuits, driven by AC power sources, that oscillated at integer sub-multiples of the AC driving frequency and featured multiple stable phase states, which they used for logic encoding. Goto's resonant circuit was termed the Parametron; in 1958, he used it to build the PC-1, a computer based on the Parametron. Computers based on Parametrons were once popular in Japan but quickly gave way to transistor-based computers with level-based logic encoding. Later versions of Goto's Parametron employed superconducting Josephson junctions to improve speed and energy consumption; however, they still remained unsuitable for large-scale integration and have been limited in application.

Recently, the phase logic idea has been revisited with new mechanisms and schemes proposed in [18, 15]. The new phase logic framework is termed PHLOGON: PHase-based LOGic using Oscillatory Nano-systems. Instead of passive resonant circuits driven by AC power, PHLOGON uses DC-powered self-sustaining nonlinear oscillators as phase logic components. When perturbed with a synchronization signal (SYNC) that oscillates at integer multiples of its natural frequency, such oscillators feature Sub-Harmonic Injection Locking (SHIL), a phenomenon via which the oscillator's phase can lock to SYNC's in multiple ways. These multiple sub-harmonic phase locks can then be used to encode logic bits. With SHIL as the central mechanism, a broad choice of nonlinear oscillators become potential candidates for implementing PHLOGON systems, including many that can be integrated and consume very little power. Moreover, phase encoding offers better noise immunity compared with level-based encoding [18, 15]. With noise and power acting as today's central barriers impeding the progress of computing, these features of PHLOGON have made it an interesting and promising alternative to conventional level-based logic computation.

However, the design of PHLOGON systems poses new challenges to existing tools. PHLOGON systems use nonlinear oscillators as underlying logic components. The simulation of even a single oscillator is often non-trivial, to say nothing of many of them connected together for logical computation. In particular, standard SPICE transient simulation is often not well suited for capturing oscillator phases, central to phase-based logic encoding, accurately and efficiently. These challenges are detailed in Section 2; no existing tools with emphasis on phase-based computation are conveniently available to designers.

In this paper, we present a set of design tools that can overcome these difficulties. The tools use phase-macromodel-based techniques (detailed in Section 2) that enable direct simulation and visualization of the phases of oscillatory signals in a PHLOGON system. In particular, the tools can predict whether SHIL will happen, and when it does, estimate the locking range and locking phase error — key properties to assess the latch's ability to store a bit. They also provide convenient simulation and visualization of the logic bit's flipping behaviour under the control of logic inputs. These features are of key importance during the design of oscillator latches; our tools have made their characterization convenient. Once individual oscillator latches have been designed, the tools perform full system simulation using phase macromodels for efficiency and improved visualization.

We demonstrate the capabilities of our tools by using them to design example ring-oscillator-based latches and a "proof-of-concept" FSM, showing how the tools are useful at every stage in the design of a PHLOGON system. Using the design tools, we have built a phase logic FSM on breadboards and verified its logical operation through oscilloscope measurements. This state machine, to our knowledge, is the first one built with DC-powered oscillators and phase-based logic encoding. Without the design tools, designing, building and debugging even this simple state machine would have been much more difficult, perhaps impossible.

The rest of this paper is organized as follows. In Section 2, we detail the challenges posed by the design of PHLOGON systems, and the contributions of our design tools in overcoming these difficulties. In

DAC '15, June 07 - 11, 2015, San Francisco, California, USA.

DOI: http://dx.doi.org/10.1145/2744769.2744818

Section 3, we provide background on oscillator PPVs and the Adler equation, concepts central to our tools. Then, in Section 4, we present our tools and demonstrate their use for the design of a ring-oscillator-based phase logic computation system, illustrating their mechanisms and capabilities along the way. Using insights and results from the design tools, we have built a ring-oscillator-based PHLOGON state machine; in Section 5, we validate the accuracy and efficiency of the design tools against both standard transient simulations and measurements of actual circuit implementations.

2. PHLOGON DESIGN CHALLENGES AND OUR CONTRIBUTIONS

The general design flow of a PHLOGON system is shown in Fig. 1. Designers start from a self-sustaining nonlinear oscillator, attach SYNC and logic control inputs to make it a phase logic latch, then construct state machines with latches and logic gates [1]. The effective and efficient design of such a system requires simulation tools that can conveniently assist designers in the following design stages:

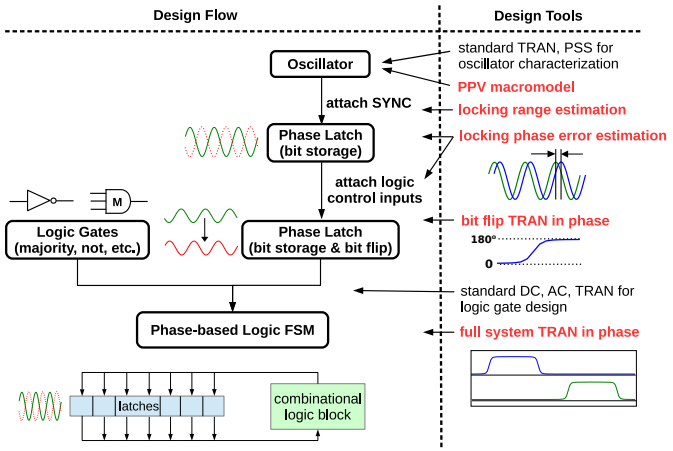

Figure 1: Design flow of PHLOGON systems and design tools required. Tools in red bold fonts are ones currently not conveniently available to designers.

1. Attaching SYNC for bit storage: simulation should be able to predict whether SHIL will occur given a design. It should estimate the locking range with respect to system parameters and inputs.
2. Attaching logic inputs for bit flip: simulation should be able to capture the transient behaviour of phases shifting from one locked state to another. This is central to the timing analysis of phase-based logic latches.
3. Full system simulation for PHLOGON: efficient simulation that directly captures phase transitions is needed to simulate the logic operation as well as timing of the final system.

At these design stages, standard SPICE-level transient simulation is not suitable due to accuracy and efficiency reasons [5, 20]. Oscillators feature phase stability, causing numerical errors in phase to accumulate without bound. This renders the SPICE-level transient simulation results meaningless for systems that operate based on phase logic. For acceptable accuracy, tiny time steps are needed in each oscillation cycle, making computation very expensive for PHLOGON systems, where thousands of cycles are often simulated for phase transitions to perform logic operation. Also, the occurrence of IL/SHIL is often hard to estimate by eye from SPICE-level transient results. Apart from all these, even if such a long transient simulation is performed accurately, it doesn't provide much insight into the improvements of the system, so the design will be largely based on trial-and-error. Therefore, as shown in Fig. 1, more efficient and convenient tools that can directly address the needs of PHLOGON designers are highly desired.

In comparison, our design tools implement techniques based on Perturbation Projection Vector (PPV) macromodels [6, 7, 23, 17] and the Generalized Adler's Equation (GAE) [4]. They provide designers with all the capabilities shown in Fig. 1, including many that are not currently conveniently available.

1. For bit storage in latches: the design tools can automatically extract PPV macromodels from an oscillator using both time-domain [7, 23] and frequency-domain [17] methods, derive the GAE from the PPV and calculate its equilibrium state to provide convenient visualization of locking range when sweeping system parameters.
2. For bit flip in latches: with the presence of logic control inputs, through the calculation of the GAE's equilibrium state, the design tools can directly return the phase error of a locked state with respect to a reference phase. This can be used to visualize a latch's logic function, i.e., verify its truth table with logic inputs. Also, the transient solutions of GAE macromodel are also available with the design tools to visualize the bit flip transient behaviour directly in phase domain.
3. Full system transient simulation with phase macromodels: after constructing PPV macromodels, the design tools can directly simulate the transient behaviour of the phases of the oscillators in a PHLOGON system. The results can be used not only to reproduce SPICE-level transient results more efficiently, but also to provide visualization of the system's logic operation in the phase domain.

The techniques used in the design tools, although the theory of them has been established ([6, 7, 23, 17, 4]), do not have open-source or commercial implementations conveniently available to designers, nor have they ever been applied to the design and analysis of phase logic computation systems before. Our design tools implement the simulation techniques, and also include visualization modules associated with them that are specially designed for phase logic operations. We expect to release the simulation and visualization tools as well as examples of PHLOGON systems implemented using them freely as open-source software to facilitate the exploration of phase-based logic computation.

3. PPV MACROMODEL AND GAE

A nonlinear self-sustaining oscillator under perturbation can be described mathematically as a set of Differential Algebraic Equations (DAEs):

$$\frac{d}{dt}\vec{q}(\vec{x}) + \vec{f}(\vec{x}) + \vec{b}(t) = \vec{0} \tag{1}$$

where $\vec{x} \in \mathbf{R}^n$ are the unknowns in the system, $\vec{b}(t)$ is a small time-varying input. The oscillator's response can be approximated well as

$$\vec{x}(t) = \vec{x}_s(t + \alpha(t)) \tag{2}$$

where $x_s(t)$ is the oscillator's steady state response without perturbation ($\vec{b}(t) \equiv \vec{0}$); $\alpha(t)$ is the phase shift caused by the external input and is governed by the following differential equation:

$$\frac{d}{dt}\alpha(t) = \vec{v}^T(t + \alpha(t)) \cdot \vec{b}(t) \tag{3}$$

where the vector $\vec{v}(t)$ is known as the Perturbation Projection Vector (PPV) [6] of the oscillator. Assume the oscillator's natural frequency is $f_0 = 1/T_0$. Then $\vec{v}(t)$ is a T_0-periodic vector that can be extracted numerically from the DAEs of the oscillator without knowing any information about the input $\vec{b}(t)$. Put in other words, it is a property intrinsic to the oscillator that captures its phase response to small external inputs.

PPV can be used to model and predict injection locking effectively [11]. Based on it people have developed a simplified approximation of (3) known as the Generalized Adler's Equation (GAE) [4]. It governs the dynamics of the oscillator's phase with specific periodic inputs $\vec{b}(t)$ and its equilibrium states provide good approximations to injection-locked solutions of (3). GAE has the following form:

$$\frac{d\Delta\phi(t)}{dt} = -(f_1 - f_0) + f_0 \cdot g(\Delta\phi(t)) \tag{4}$$

where $\Delta\phi(t) = f_0 t + f_0\alpha(t) - f_1 t$ is phase difference between the oscillator and the perturbation signal; f_1 is the fundamental frequency of periodic input $\vec{b}(t)$ and $f_1 \approx f_0$; function g is derived from (3) and can be evaluated numerically based on the formulation in [4].

[1] Majority and Not operations constitute a logically complete set [18] and they are used in PHLOGON to build combinational logic blocks.

When injection locking happens, the phase difference $\Delta\phi(t)$ between oscillator and injected signal becomes constant, i.e.

$$\frac{f_1 - f_0}{f_0} = g(\Delta\phi^*) \tag{5}$$

By plotting the LHS and RHS of (5) and looking for intersections, designers can predict whether IL or SHIL will happen given a design [1] without running long and expensive transient simulations.

4. DESIGN TOOLS FOR OSCILLATOR-BASED COMPUTING SYSTEMS

As an overview, Fig. 2 illustrates the mechanisms and usage of our design tools. All the operations shown in Fig. 2 are automatically performed to guide the design of oscillator latches and the full PHLOGON system.

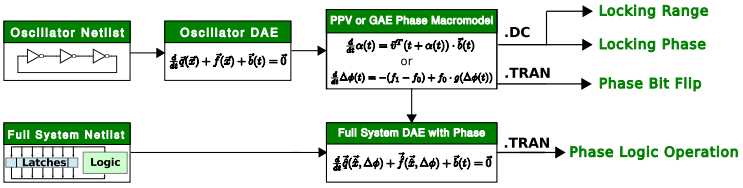

Figure 2: Overview of the mechanism of the design tools.

The key of the operation of the design tools is to transform the oscillator models to phase domain. Using phase macromodels, instead of simulating oscillating voltages and currents in the system, we can directly calculate the responses of phases.

In the rest of the section, we follow the design procedures of a ring oscillator phase logic FSM example, and show how the simulation and visualization provided by the design tools can be helpful in every stage in the design.

4.1 Simulation of Bit Storage in Oscillator Latches

As described in Section 1, bit storage in an oscillator latch is achieved through Sub-harmonic Injection Locking (SHIL). But what are the conditions for SHIL to happen? How the adjustments on circuit structures affect SHIL's locking range? And when SHIL happens, how to predict the exact phase values of the locked states such that they can be used as references in phase-based logic encoding?

These questions are hard to answer by trial-and-error approaches with only the traditional SPICE-level transient simulation tools. In comparison, our design tools can automatically extract the ring oscillator's PPV macromodel as in (3), perform Generalized Adlerization and derive the oscillator's GAE equilibrium formula shown in (5). By plotting the LHS and RHS of (5), the solution of it, predicting the occurrence of SHIL, can be easily visualized.

As illustration, we apply our tools on a ring oscillator example shown in Fig. 3 and show how they can be used to answer these questions from designers. To store a phase-based binary bit, we perturb the ring oscillator with an external current source SYNC with $I_{\text{SYNC}} = A \cdot \cos(2\pi \cdot 2f_1 \cdot t)$, where f_1 is close to its natural frequency f_0. In this oscillator, we choose C to be 4.7nF and assume that each CMOS inverter consists of 1 ALD1106 NMOS and 1 ALD1107 PMOS for the convenience of breadboard prototyping later on. In Fig. 5 we show the plot of the LHS and RHS of (5) at $f_1 = 9.6$kHz with various magnitudes A of SYNC produced by our design tools on this oscillator. From Fig. 5 we can see that when A is larger than 70μA, there are four intersections of LHS and RHS, two of them representing stable DC solutions of (4) ([4]).

Although Fig. 5 provides a good visualization of the existence and stability of the solutions of (5), it is not necessary to plot LHS and RHS across the entire range of $\Delta\phi$ in order to predict the occurrence of SHIL numerically. Instead, after formulating GAE in our design tools, DC operating point solving will be attempted with different initial guesses. If no solution is found, the locking condition in (5) is not satisfied with any $\Delta\phi^*$ and SHIL won't happen; if a solution $\Delta\phi^*$ is found, the derivative of $g(\Delta\phi^*)$ is calculated. Based on Lyapunov's

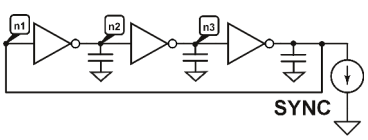

Figure 3: Diagram of a 3-stage ring oscillator.

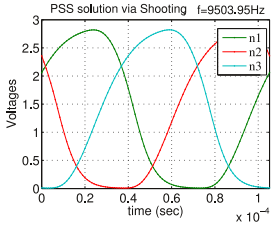

Figure 4: PSS reponse of the free-running ring oscillator.

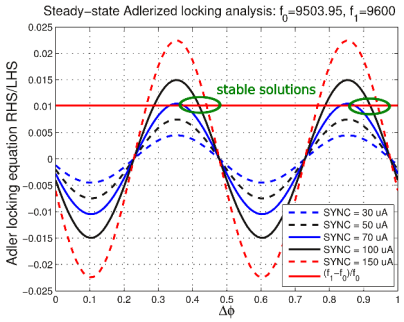

Figure 5: Graphical solutions of (5) in the ring oscillators with sinusoidal SYNC of various magnitudes.

Stability Theorem [12] in the scalar case, if the derivative is negative, the solution is stable and SHIL will happen. Such operation points can be calculated in a DC sweep on parameters A and f_1 to give designers a more comprehensive idea of the locking range. Considering that GAE is a scalar ODE, this method is robust and efficient.

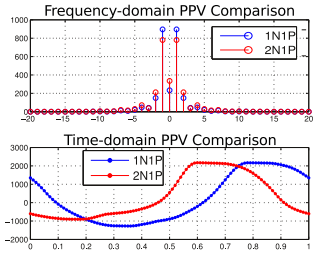

Figure 6: PPVs extracted by the design tools from ring oscillator latches with 2N1P and 1N1P inverters.

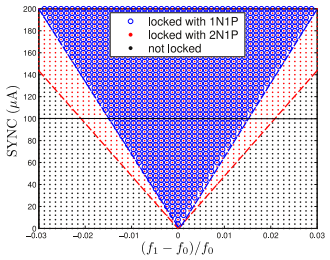

Figure 7: Locking range returned by the design tools on ring oscillator latches with 2N1P and 1N1P inverters.

Applying this DC-sweep-based method on the GAE of the ring oscillator, the locking range for SHIL can be visualized as in Fig. 7. As a comparison, we perform the same analysis on another ring oscillator design with inverters made of 2 NMOS devices and 1 PMOS (2N1P). We overlay their PPVs and locking ranges in Fig. 6 and Fig. 7. The comparison reveals that after asymmetrizing the inverters (changing to 2N1P), the second harmonic component of the PPV becomes larger, resulting in a larger locking range for SHIL. Such insight in design is difficult to get without facilities made available by our tools based on PPV/GAE macromodels.

Apart from the locking range, in order to use the bi-stable phases to encode logic bits in an oscillator latch, designers also need to know the exact phase value of the oscillator's response under SHIL, i.e., where the peak of the oscillatory signal lies within a cycle. To analyze this property, we first normalize the time axis in the oscillator latch's T_0-periodic PSS response $\vec{x}_s(t)$ and define a 1-periodic function:

$$\vec{x}_{s(1)}(t) = \vec{x}_s(T_0 t) \tag{6}$$

Assume that the peak of the output $V(n_1)$ is at position $\Delta\phi_{\text{peak}}$ in this 1-periodic function, e.g., $\Delta\phi_{\text{peak}} \approx 0.21$ in Fig. 4. Then a cos function shifted by $\Delta\phi_{\text{peak}}$

$$V_{\text{peak}}(t) = \cos\left(2\pi \cdot (t - \Delta\phi_{\text{peak}})\right) \tag{7}$$

will have peaks aligned with those in the 1-periodic PSS $\vec{x}_{s(1)}(t)$.

When SHIL happens in the latch, assume the two stable solutions of the GAE in equilibrium (5) are $\Delta\phi_0$ and $\Delta\phi_1$, separated by 0.5. To

represent 1 and 0 in phase-encoding, we define reference signals:

$$V_{\text{REF}}(t) = \frac{V_{dd}}{2} + \frac{V_{dd}}{2}\cos\left(2\pi(f_1 t - \Delta\phi_{\text{peak}} - \Delta\phi_0)\right) \quad (8)$$

$$V_{\overline{\text{REF}}}(t) = \frac{V_{dd}}{2} + \frac{V_{dd}}{2}\cos\left(2\pi(f_1 t - \Delta\phi_{\text{peak}} - \Delta\phi_1)\right) \quad (9)$$

such that when SHIL happens in the latch, the peaks of the output $V(n_1)$ will be bi-stably aligned with those of either V_{REF} or $V_{\overline{\text{REF}}}$, representing either phase-based 1 or 0.

Ideally, designers would like to operate the latch with $f_1 = f_0$ such that the locking range is maximized (Fig. 7). However, when detuning happens, i.e., $f_1 \neq f_0$, even though the latch may still stay in lock with a large enough SYNC, its locking phases $\Delta\hat{\phi}_i$ may deviate from references $\Delta\phi_i$, and their "quality" as logic bits may degenerate. For this design spec, our tools offer a good visualization by calculating the DC solutions of GAE and plotting the magnitudes of locking phase errors $|\Delta\hat{\phi}_i - \Delta\phi_i|$ across the locking range.

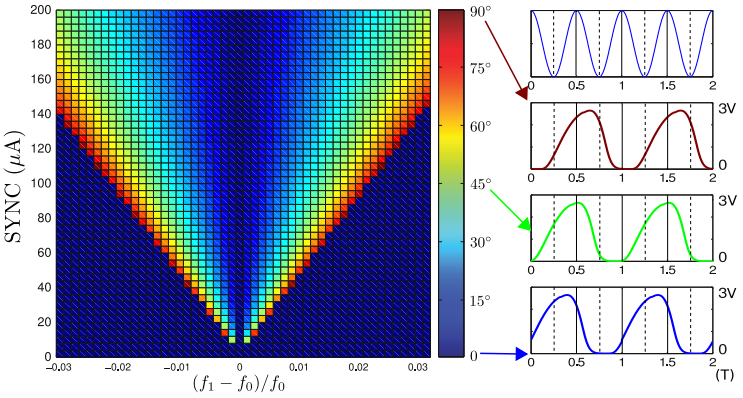

Figure 8: Locking phases errors $|\Delta\hat{\phi}_i - \Delta\phi_i|$ within the locking range. 0 means the signal under injection locking has the same phase as the reference.

The same procedures that calculate and plot GAE's DC solutions can be applied to any oscillator for the analysis of bit storage in phase logic latches. With their help, designers are able to gain comprehensive knowledge on the locking range, locking phases as well as the effects of circuit structure and parameters on them.

4.2 Simulation of Bit Flip in Oscillator Latches

After attaching SYNC to the latch and defining phase-logic references based on its SHIL properties, designers then attach logic inputs to the latch to control the bit stored in it. But where to attach these inputs? And given a certain topology, what are the appropriate input magnitudes? Similar to the bit storage scenario, the visualization of GAE solutions in our tools can be useful in determining these design specs.

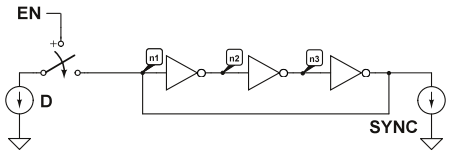

Figure 9: Diagram of a ring-oscillator D latch with phase-based D and level-based EN.

As an example, Fig. 9 shows the diagram of a simple D latch where EN is a level-based logic input that controls the on or off state of a switch, D input is an oscillatory current source with a certain output impedance:

$$I_{\text{D}}(t) = -A_D\cos\left(2\pi(f_1 t - \Delta\phi_{\text{peak}} - \Delta\phi_i)\right), i = 0, 1 \quad (10)$$

such that when EN = 1, $i = 0$, it enforces $V(n_1)$ to be aligned with V_{REF}; when EN = 1, $i = 1$, it enforces $V(n_1)$ to be aligned with $V_{\overline{\text{REF}}}$; when EN = 0, no matter what I_{D}'s phase is, $V(n_1)$ should hold on to its original logic value. In this way, it achieves the functionality of a D latch.

Assume that the switch is implemented using a transmission gate made of ALD1106/7 long-channel MOSFET devices with a typical on-resistance of 1kΩ and off-resistance of 100GΩ. The current source is implemented with ALD1106 NMOS at the output stage with typical output impedance of 10MΩ. Under these conditions, what's an appropriate value for A_D such that the logic bit in the latch can be flipped by I_D when EN=1, but retains its value when EN=0?

To answer this question, designers can again ask our tools to visualize the solutions of GAE, but this time with both SYNC and D as inputs. For example, Fig. 10 plots the LHS and RHS of GAE equilibrium equation (5) with EN=1, SYNC=100μA and various A_D values. From it, we observe that when A_D is larger than approximately 50μA, one of the stable phase states in SHIL vanishes and the output's phase is controlled by input D only.

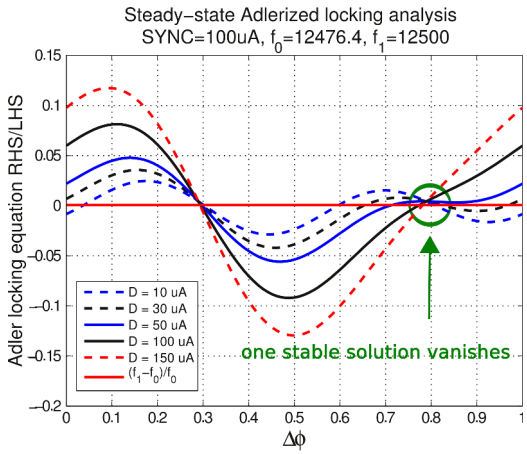

Figure 10: Graphical solutions of (5) in the D latch with 100μA SYNC and various magnitudes of D signals when EN=1. As D increases, one stable solution vanishes.

Similar to the calculation of locked phases in bit storage scenario, instead of visualizing LHS and RHS at each input value, the design tools can sweep the magnitude of D and return all stable solutions of (5) that predict the stable phases under lock. Fig. 11 shows all these locked phase solutions predicted by the tools with various magnitudes of D when EN=0 and 1. From it, designers are able to get a comprehensive idea on how the choice of parameter A_D can affect the logic operation of the latch.

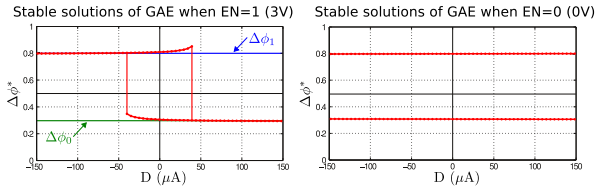

Figure 11: GAE equilibrium solutions with different D magnitudes when EN=1 and EN=0.

Furthermore, after adjusting parameters to make the circuit behave as a D latch under steady state, designer also have to meet certain timing specifications, i.e., they have to make sure that the phase can flip to these desired steady states quickly enough. To simulate this bit flipping behaviour efficiently, our design tools can directly run transient simulation on the latch's GAE macromodel in (4) and the waveform of $\Delta\phi(t)$ transiting from $\Delta\phi_0$ to $\Delta\phi_1$ offers direct phase-domain visualization of the latch's timing properties.

We use the D latch example in Fig. 9 again as illustration. Based on just the GAE DC solutions shown in Fig. 11, any D signal with a magnitude larger than 50μA seems suitable in the design. However, results from transient simulations on the GAE (in Fig. 12) reveal more valuable information. They not only confirm that 30μA is not sufficient for the latch's response to catch up with D for logic operation, but also predict that even though 50μA D signal will flip the phase eventually, the bit flipping speed is probably not desirable. It is much slower than the flipping with 100μA, and the difference in timing is much larger than that between 100μA and 150μA. It is worth mentioning that transient simulations run on a scalar phase domain differential equation as in (4) are much more efficient than those run directly on the DAEs of oscillators. Their results are also more straightforward in visualizing the latch's operation. With such

facility made available by our design tools, designers can then adjust their phase-based latch design accordingly to meet timing specs needed for logic computation.

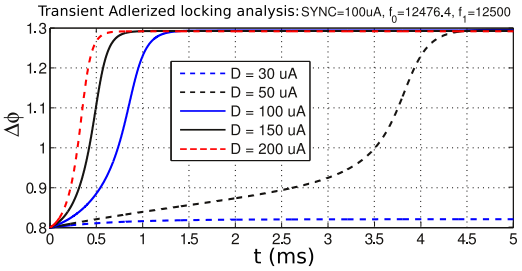

Figure 12: Transient simulations on GAE predict the bit flipping timing behaviours.

While Fig. 9 demonstrates a D latch with level-based EN, with the help of the design tools, more complicated oscillator latches with logic inputs completely encoded in phases can be designed. Fig. 13 shows our design of an SR latch and a D latch. Although their logic operations (truth tables in Fig. 13) are straightforward to understand from their structures, their practical implementations rely heavily on the design tools.

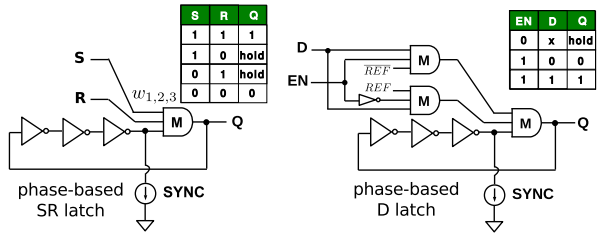

Figure 13: Diagrams of completely phase-based SR and D latches.

One design consideration in such an SR latch is that the latch should tolerate certain amount of mismatch between S and R. More specifically, when S and R have opposite phases but slightly different magnitudes, the logic value in the latch should still hold. Put in other words, the residue after cancelling opposite S and R shouldn't flip the bit stored in the latch. Similar to the example in Fig. 11, we use the design tools to sweep the magnitudes of S and R and plot all the stable solutions of (5) in Fig. 14. From it we conclude that a conventional majority gate that adds its inputs with equal weights $w_1 = w_2 = w_3 = 1$ (black line in Fig. 14) is not suitable for SR latch designs like Fig. 13. As an improvement, changing the weights of the majority gate to $w_1 = w_2 = 0.01, w_3 = 1$ (red lines) will make the latch tolerant more mismatch between S and R while still securely flip to desired logic state when S and R have the same phase with the magnitude of $Vdd/2 = 1.5V$.

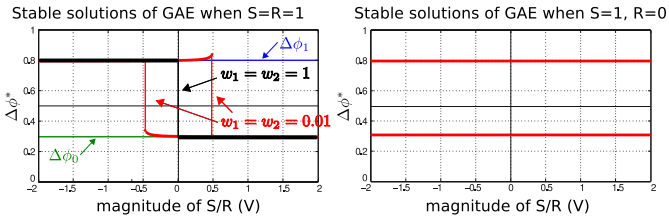

Figure 14: GAE equilibrium solutions of an SR latch as in Fig. 13. The left and right subfigures plot solutions with different magnitudes of S and R signals, encoding the same and opposite phase logic values respectively.

4.3 Full System Transient Simulation with Phase Macromodels

With oscillator latches and logic gates, phase-based general-purpose computing can be implemented by assembling them into FSMs. In the operation of a phase logic FSM, when the logic values of inputs change over time, i.e., their phase differences with respect to the reference signal shift back and forth between 0 and 180°, designers need to know whether the system is indeed performing computation properly with phase-based logic.

For this purpose, traditional transient simulation can be performed

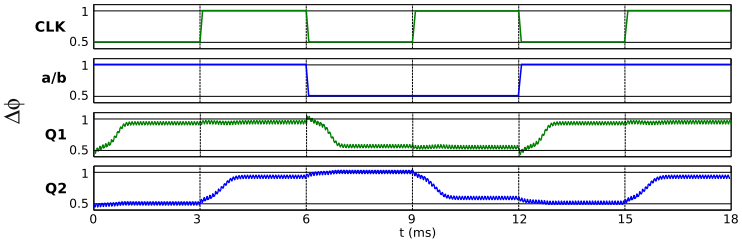

Figure 16: Transient simulation of the serial adder in Fig. 15 with oscillator latches replaced by their PPV macromodels. $\Delta\phi = 0.5$ indicates opposite phase w.r.t. REF, thus encoding phase-based 0, while $\Delta\phi = 1$ encodes 1.

on the DAE of the full system:

$$\frac{d}{dt}\vec{q}_{\text{full}}(\vec{x}_{\text{full}}) + \vec{f}_{\text{full}}(\vec{x}_{\text{full}}) + \vec{b}_{\text{full}}(t) = \vec{0} \tag{11}$$

However, if we separate system unknowns $\vec{x}_{\text{full}}$ into $\vec{x}_{\text{osc}_i}$ and $\vec{x}_{\text{other}}$, where $\vec{x}_{\text{osc}_i}, i = 1, 2, \cdots, k$ represents unknowns inside each of the k oscillator latches, we know $\vec{x}_{\text{osc}_i}$ can be approximated well with its steady state response $\vec{x}_{\text{osc}_i(s)}$ and its phase α_i as in (2). As α_i is unbounded in simulation, similar to the formulation of GAE, we use the locking phase error $\Delta\phi_i = f_0 t + f_0\alpha_i(t) - f_1 t$ instead, such that

$$\vec{x}_{\text{osc}_i}(t) = \vec{x}_{\text{osc}_i(s)}((f_1 t + \Delta\phi_i(t))/f_0) \tag{12}$$

where $\Delta\phi_i$ is governed by

$$\frac{d}{dt}\Delta\phi_i(t) = f_0 - f_1 + f_0 \cdot \vec{v}_i^T(T_1\Delta\phi_i(t) + t) \cdot \vec{b}_i(t) \tag{13}$$

where external $\vec{b}_i(t)$ can be calculated from $\vec{x}_{\text{other}}$ based on circuit connections.

The formulation in (13) is slightly different from the GAE in (4) in that the fast varying mode is preserved instead of averaged out [4]. In this way, the full system can be formulated as

$$\frac{d}{dt}\vec{q}'(\vec{x}_{\text{other}}, \Delta\vec{\phi}) + \vec{f}'(\vec{x}_{\text{other}}, \Delta\vec{\phi}) + \vec{b}(t) = \vec{0} \tag{14}$$

where $\Delta\vec{\phi}$ represents all $\Delta\phi_i$s. For each oscillator latch, all its voltage and current unknowns are now represented by only one scalar. Therefore, the system size is reduced.

Instead of formulating (14) analytically, our design tools allow users to identify subcircuits that describe oscillator latches and replace them with their PPV macromodels as in (13). Since the PPV of the latch has already been calculated in previous stages of design, little computation is introduced in reformulating the full system into (14). Such reformulation with PPV macromodels results in an equation system not only with less unknowns, but also with unknowns that are more directly related to the phase operation of the system.

As illustration, in Fig. 15 we show a simple FSM with 1-bit state: a serial adder with inputs a, b and CLK all encoded in phase logic. The D latches in Fig. 15 are the same as in Fig. 13 and their PPV macromodels have already been constructed in the previous stage of design in Section 4.2. Our design tools then use the macromodels to replace the latches and run transient simulation on the resulting system. From the transient results in Fig. 16, we observe the phase transition of the two latches ($\Delta\phi$ of Q1 and Q2) when adding a=b=101 sequentially, in particular how Q2 follows Q1's phase in the master-slave flip-flop.

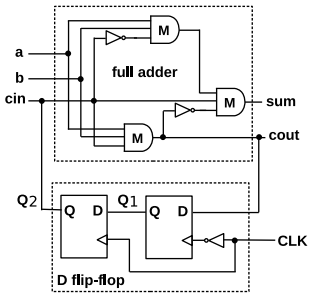

Figure 15: Serial adder.

5. EXPERIMENTS AND VALIDATION

The design and simulation of oscillator latches as well as the FSM made from them have already demonstrated the mechanisms and capabilities of our design tools in every stage of the implementation of oscillator-based computing systems. In this section, using standard transient simulation and breadboard circuit experiments, we further validate that the system designed with our tools can behave as pre-

dicted.

5.1 Bit Flip Transient Simulation

Firstly, we simulate the transient behaviour of a phase-based logic bit switching states in the oscillator latch as in Fig. 9. As discussed in Section 4.2, our design tools can efficiently predict this key feature of a given oscillator latch. As an experiment, we run SPICE-level transient simulation with the same magnitude of SYNC (100μA) as chosen in Section 4.2 and flip D signal's phase (with magnitude 150μA) when EN=1. To study the phase transition of the latch, we plot the output of the latch in Fig. 17. We also measure its zero crossings [2], calculate their differences between those of the reference signal, and plot the differences also in Fig. 17. Then we overlay the predictions by GAE from our design tools (black curve) to compare against the phase transition represented by zero crossings. From the comparison, we observe that even though they don't exactly overlap due to different definitions of phase, they show similar results in the amount of time needed for the latch to settle at the new phase state, verifying the capability of the design tools in efficiently analyzing the timing of oscillator latches.

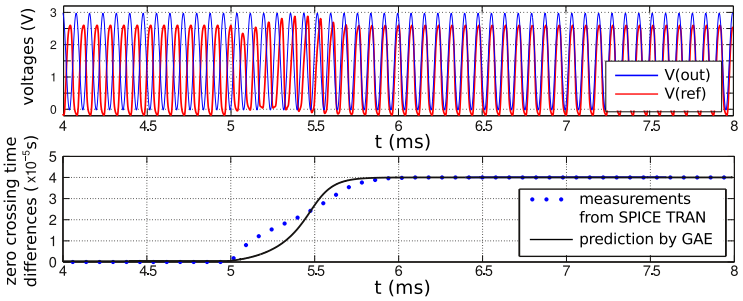

Figure 17: SPICE-level transient simulation results of bit flipping in D latch shown in Fig. 9: waveforms (top subfigure) and zero crossing differences between V(out) and V(ref) compared with prediction from GAE in Fig. 12 (bottom subfigure).

5.2 Breadboard Experiments Validating FSM's Operation

With the help of the design tools, we have followed the design procedures of a phase-based FSM in Section 4. It's logic operation as a serial adder has been predicted to be valid based on full system simulations with PPV macromodels. To demonstrate that such a computing system predicted to be working in our design tools will also work in reality, we build an FSM as in Fig. 15 on breadboards (Fig. 18). In the actual implementation, the majority and not gates are implemented with op-amps with resistive feedbacks.

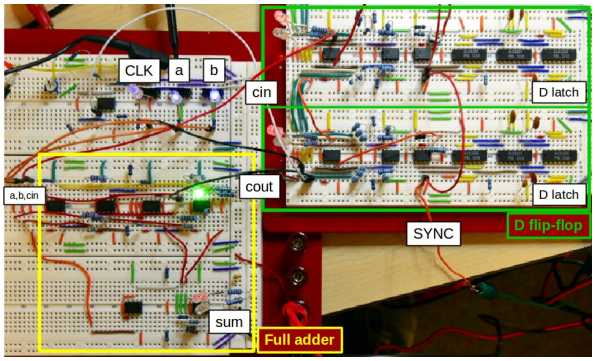

Figure 18: Breadboard implementation of the serial adder in Fig. 15. The green blocks highlight the flip-flop made from two D latches; the yellow block is the combinational logic block that has the functionality of a full adder.

Fig. 19 and Fig. 20 show snapshots of test results seen from an oscilloscope. With thorough testing with various input combinations and sequences, we conclude that the circuit in Fig. 18 implements a phase-based serial adder. It is worth mentioning that to our knowledge, this is the first phase-based state machine made with DC-powered oscillators, and a lot more trial and error would have been required without the help of the design tools described in this paper.

[2]With single-ended power supply we consider crossings with the offset $Vdd/2$ on the rising slopes as zero crossings.

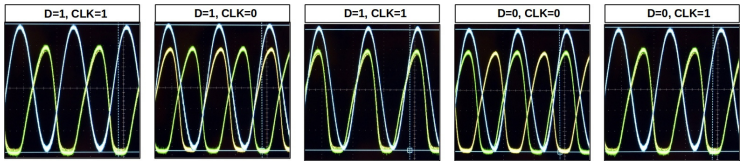

Figure 19: Test results of the master-slave D flip-flop as shown in the green block of Fig. 18. The blue, green, yellow signals represent REF, Q1, Q2 respectively. As CLK shifts between logic 1 and 0, Q1 (green) always follows input D at falling edges of CLK, while Q2 (yellow) follows Q1 at rising edges of CLK. This validates the simulation results shown in Fig. 16.

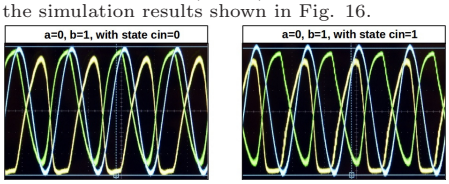

Figure 20: Selected test results of the serial adder as in Fig. 18. REF, sum, cout are displayed in blue, green, yellow respectively. With the same inputs: a=0 and b=1, the left subfigure shows sum=1, cout=0 when the state machine is at carry=0 state; the right one shows sum=0, cout=1 when the state machine is at carry=1 state.

6. CONCLUSIONS

In this paper we introduced design tools based on PPV and GAE macromodels of oscillators for implementing oscillator-based computing systems. Following the design procedures of an example phase-based FSM, we demonstrated the usage and capabilities of the tools in all the stages of design, from predicting an oscillator latch's bit storage and flipping properties to the efficient simulation of the full FSM system. With the help of the design tools we were able to implement a phase-based state machine on breadboards. We plan to release the design tools as open-source software to facilitate the designers in the exploration of oscillator-based general-purpose computing.

7. REFERENCES

[1] A. Neogy and J. Roychowdhury. Analysis and Design of Sub-harmonically Injection Locked Oscillators. In Proc. IEEE DATE, Mar 2012.

[2] H. Arsenault. Optical processing and computing. Elsevier, 2012.

[3] B. Behin-Aein, D. Datta, S. Salahuddin, and S. Datta. Proposal for an all-spin logic device with built-in memory. Nature nanotechnology, 5(4):266–270, 2010.

[4] P. Bhansali and J. Roychowdhury. Gen-Adler: The generalized Adler's equation for injection locking analysis in oscillators. In Proc. IEEE ASP-DAC, pages 522–227, January 2009.

[5] P. Bhansali and J. Roychowdhury. Injection locking analysis and simulation ofÂăweakly coupled oscillator networks. In P. Li, L. M. Silveira, and P. Feldmann, editors, Simulation and Verification of Electronic and Biological Systems, pages 71–93. Springer Netherlands, 2011.

[6] A. Demir, A. Mehrotra, and J. Roychowdhury. Phase Noise in Oscillators: a Unifying Theory and Numerical Methods for Characterization. IEEE Trans. Ckts. Syst. – I: Fund. Th. Appl., 47:655–674, May 2000.

[7] A. Demir and J. Roychowdhury. A Reliable and Efficient Procedure for Oscillator PPV Computation, with Phase Noise Macromodelling Applications. IEEE Trans. on Computer-Aided Design, pages 188–197, February 2003.

[8] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. SIGARCH Comput. Archit. News, 39(3):365–376, June 2011.

[9] E. Goto. New Parametron circuit element using nonlinear reactance. KDD Kenyku Shiryo, October 1954.

[10] D. Hisamoto, W.-C. Lee, J. Kedzierski, H. Takeuchi, K. Asano, C. Kuo, E. Anderson, T.-J. King, J. Bokor, and C. Hu. Finfet-a self-aligned double-gate mosfet scalable to 20 nm. Electron Devices, IEEE Transactions on, 47(12):2320–2325, 2000.

[11] X. Lai and J. Roychowdhury. Capturing injection locking via nonlinear phase domain macromodels. IEEE Trans. Microwave Theory Tech., 52(9):2251–2261, September 2004.

[12] A. M. Lyapunov. The general problem of the stability of motion. International Journal of Control, 55(3):531–534, 1992.

[13] G. E. Moore et al. Cramming more components onto integrated circuits, 1965.

[14] K. S. Novoselov, V. Fal, L. Colombo, P. Gellert, M. Schwab, K. Kim, et al. A roadmap for graphene. Nature, 490(7419):192–200, 2012.

[15] J. Roychowdhury. Boolean Computation Using Self-Sustaining Nonlinear Oscillators. arXiv:1410.5016v1 [cs.ET], Oct 2014. arXiv:1410.5016v1http://arxiv.org/abs/1410.5016v1/.

[16] M. M. Shulaker, G. Hills, N. Patil, H. Wei, H.-Y. Chen, H.-S. P. Wong, and S. Mitra. Carbon nanotube computer. Nature, 501(7468):526–530, 2013.

[17] T. Mei and J. Roychowdhury. PPV-HB: Harmonic Balance for Oscillator/PLL Phase Macromodels. In Proc. ICCAD, pages 283–288, Nov. 2006.

[18] T. Wang and J. Roychowdhury. PHLOGON: PHase-based LOGic using Oscillatory Nanosystems. In Proc. Unconventional Computation and Natural Computation: 13th International Conference, UCNC 2014, London, ON, Canada, July 14-18, 2014, LNCS sublibrary: Theoretical computer science and general issues. Springer, 2014.

[19] J. von Neumann. Non-linear capacitance or inductance switching, amplifying and memory devices. 1954.

[20] Z. Wang, X. Lai, and J. Roychowdhury. PV-PPV: Parameter Variability Aware, Automatically Extracted, Nonlinear Time-Shifted Oscillator Macromodels. In Proc. IEEE DAC, San Diego, CA, June 2007.

[21] R. L. Wigington. A New Concept in Computing. Proceedings of the Institute of Radio Engineers, 47:516–523, April 1959.

[22] L. Wilson. International technology roadmap for semiconductors (itrs). 2013.

[23] X. Lai and J. Roychowdhury. TP-PPV: Piecewise Nonlinear, Time-Shifted Oscillator Macromodel Extraction For Fast, Accurate PLL Simulation. In Proc. ICCAD, pages 269–274, November 2006.