Evolution of Astable Multivibrators \textit{in Silico}

Lorenz Huelsbergen\textsuperscript{1} and Edward Rietman\textsuperscript{1} and Robert Slous\textsuperscript{2}

\textsuperscript{1} Bell Laboratories, Lucent Technologies\textsuperscript{***}
\textsuperscript{2} Xilinx Inc.\textsuperscript{†}

Abstract. We use evolutionary search to find automatically electronic circuits that toggle an output line at, or close to, a given target frequency. Reconfigurable hardware in the form of field-programmable gate arrays—as opposed to circuit simulation—computes the fitness of a circuit which guides the evolutionary search. We find empirically that oscillating circuits can be evolved that closely approximate some of the supplied target frequencies. Our evolved oscillators alias a harmonic of the target frequency to satisfy the fitness goal. Frequencies of the evolved oscillators were sensitive to temperature and to the physical piece of silicon in which they operate. We posit that such sensitivities may have negative implications for demanding applications of reconfigurable hardware and positive implications for adaptive computing.

1 Introduction

Complex entities—biological and artificial, for example—are in part governed by systolic processes. In many animals, hearts beat at (perhaps variable) frequencies to distribute fluids. Circuits in machines coordinate information flow in step with a local or global clock. Biological oscillation arose from primitive components (molecules) in an environment (physics) via the process of natural selection. Mechanical oscillation arose in computers, and in many other machines, through human design. We seek to understand if evolution can be harnessed to design pieces of computing machines. Toward this end, we are conducting experiments to “evolve” circuits—oscillators (astable multivibrators) in particular—from primitive logic components. Our goals are twofold: The exploration of the capabilities of \textit{in Silico} evolution and the investigation of whether computational circuits based on oscillators can thus be constructed.

The genetic algorithm (GA) \cite{4} is a form of \textit{evolutionary search}. GAs have been shown to perform well as a general optimization technique across a broad range of domains (see Goldberg \cite{2} for examples). The GA maintains a population of individuals (bit strings) over a series of \textit{generations}. The initial population is random. Using an externally supplied \textit{fitness function} (environment), the GA selects promising individuals for the next generation. Some such selected individuals are then paired and, with random substrings interchanged, placed in the next generation.

Evolutionary search—most recently in the form of GA-based \textit{genetic programming} (GP)—has been used to evolve computer software (e.g., \cite{6}). In this

\textsuperscript{***}{lorenz,ear}@bell-labs.com
\textsuperscript{†}robert.slous@xilinx.com
context, the bit string comprising an individual is interpreted as a (perhaps variable length) sequence of instructions written in a computer language. Distance, in some metric space, between the result of evaluating a GP individual and the desired target result constitutes an individual’s fitness. It is well known that digital software and hardware are computationally equivalent. This suggests that application of software evolution techniques may also be fruitful in a hardware realm [8, 3, 1, 12]. With evolvable computational structures, the programming onus shifts from providing an algorithm (circuit or program) for solving the task at hand to crafting a function that assigns an accurate fitness measure to partial and complete solutions. Search—taking the form of a GA for this paper—can then automatically perform algorithm discovery.

The recent experiments of Thompson [8] in particular demonstrate that reconfigurable logic in the form of field-programmable gate arrays (FPGAs) can serve as a viable substrate for gate-level hardware evolution. Thompson evolved discriminator circuits that, when presented with one of two possible input tones (frequencies), would correctly classify their input. Our system for in Silico evolution is similar to Thompson’s. Our study however concerns circuits with fundamentally different characteristics than input-sensitive frequency discriminators—we are evolving computational components, namely oscillators, that function as stand-alone clocks.

Our result is the automatic generation of oscillators at specified target frequencies from primitive electronic components. Given only a target frequency $f$, our system can produce—from logic gates (such as “not”) and wires to connect them—a circuit whose single output oscillates between the logic states low and high with frequency $f$ (or a harmonic multiple thereof). Note that the circuit undergoing evolution receives no input in general and no clock signal in particular. It completely synthesizes oscillation.

Fig. 1.: A manually designed ring oscillator constructed from three inverters. The output oscillates due to gate and signal propagation delays. A genetic algorithm can find circuits with similar behavior—but not necessarily of similar structure—that oscillate at predefined target frequencies.

One route to such oscillation is exploitation of gate and signal propagation delays along with feedback. The manually designed ring oscillator of Figure 1 illustrates this basic principle. This oscillator’s frequency depends on the speed of the substrate’s implementation technology. Note that the oscillator’s frequency may be reduced by inserting additional inverters into the ring. It is difficult, even for skilled designers, to craft feedback circuits with specified characteristics (e.g., frequency) solely from logic gates. In practice, therefore, oscillators are usually constructed from (relatively expensive) analog components.

Gates in contemporaneous off-the-shelf FPGAs typically switch in nanoseconds.