Evolvable Hardware: Evolution in FPGAs

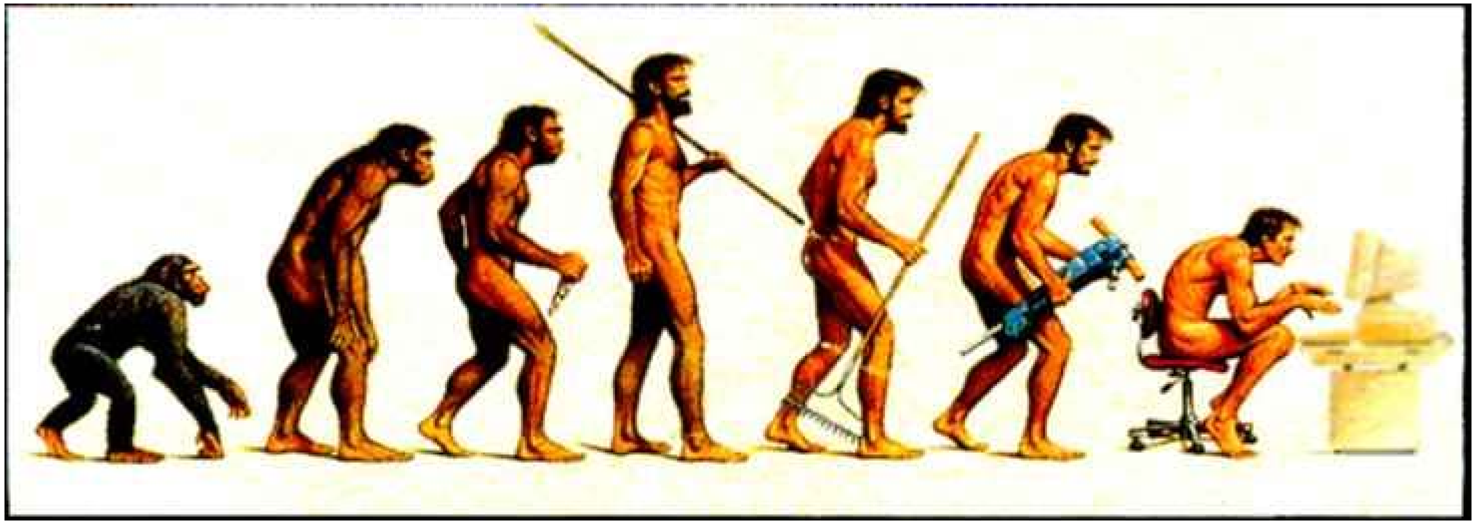
Adrian Thompson
Evolutionary & Adaptive Systems Group,
University of Sussex, Brighton UK

Mark L. Chang

ACME Seminar

December 3, 2002

Evolution



- A theory that the various types of animals and plants have their origin in other preexisting types and that the distinguishable differences are due to modifications in successive generations

Traditional Design

- Conventional *digital design* is just an abstraction for humans
- Simplifies designs by allowing reality to be ignored
  - Properties of real hardware are suppressed
  - Analog behavior of transistors reduced to ON/OFF switches
  - Transients suppressed by using a *synchronous* design
  - Modules are compartmentalized
  - Communicate only on *clock edges*
  - Transients are localized to modules and don’t influence other modules
- Spatial and dynamical behavior is constrained
- In any design methodology, we can’t get far without help from abstraction!

Why Evolvable Circuits?

- We can’t understand everything, so let Mother Nature do the tough stuff
- Increase the search space by eliminating constraints
- Let evolution organize the circuit spatially and dynamically
- There is a possibility for better solutions than in constrained designs

Why “In Silicon” Evolution?

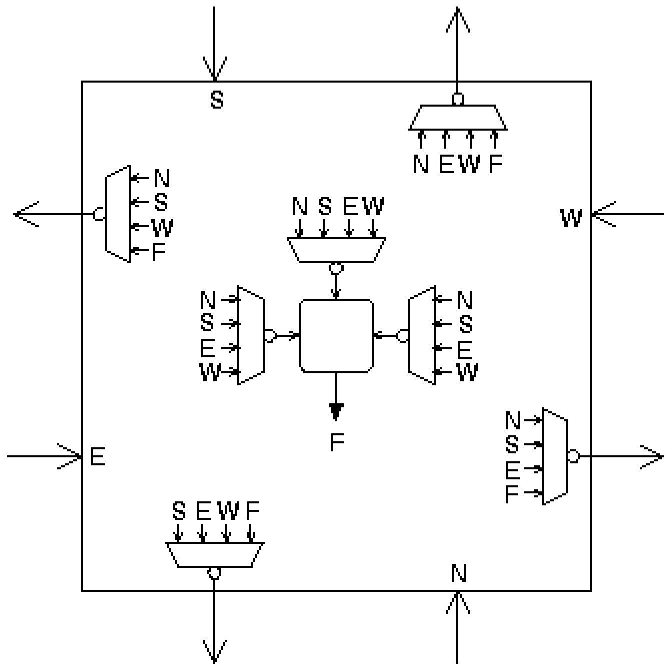
- Evolution normally takes place in software simulations
    - Easy to manipulate
    - Usually an Artificial Neural Network (ANN)—a constrained system
    - Not too slow that we need to accelerate using hardware
- Evolution in hardware gives us unique opportunities
    - Can exploit real-world physics that are often difficult or impossible to analyze or model in simulations—let’s us drop simplifying constraints.
    - Physical components have a size, shape and location, which are critical in determining the interactions between them. No longer artificial point-to-point interconnections as in simulations.
    - Characteristics of the components and their interactions are not exactly predictable or constant over time. Evolution must cope with this.

Artificial Evolution Basics

- Chromosome/Genotype
  - Bit string that represents a possible solution
- Population
  - Collection of chromosomes
- Fitness
  - The goodness of the solution expressed by a particular chromosome
- Crossover
  - A method of creating a child chromosome by taking sections from one parent or the other
- Mutation
  - Random inversion of bits in a chromosome

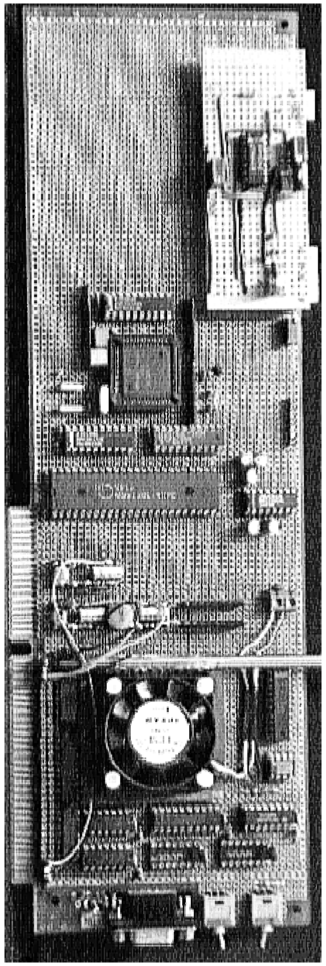
The Evolvable Hardware

- Xilinx XC6216 FPGA
  - 64x64 array of reconfigurable cells
  - NEWS connectivity
  - Functional block or route-through
  - 2-input Boolean / 3-input MUX
  - No illegal configurations possible
- Only nearest neighbor connections used
- Constrained to 10x10 corner
- One input and one output on IOBs
- 18 configuration bits per cell

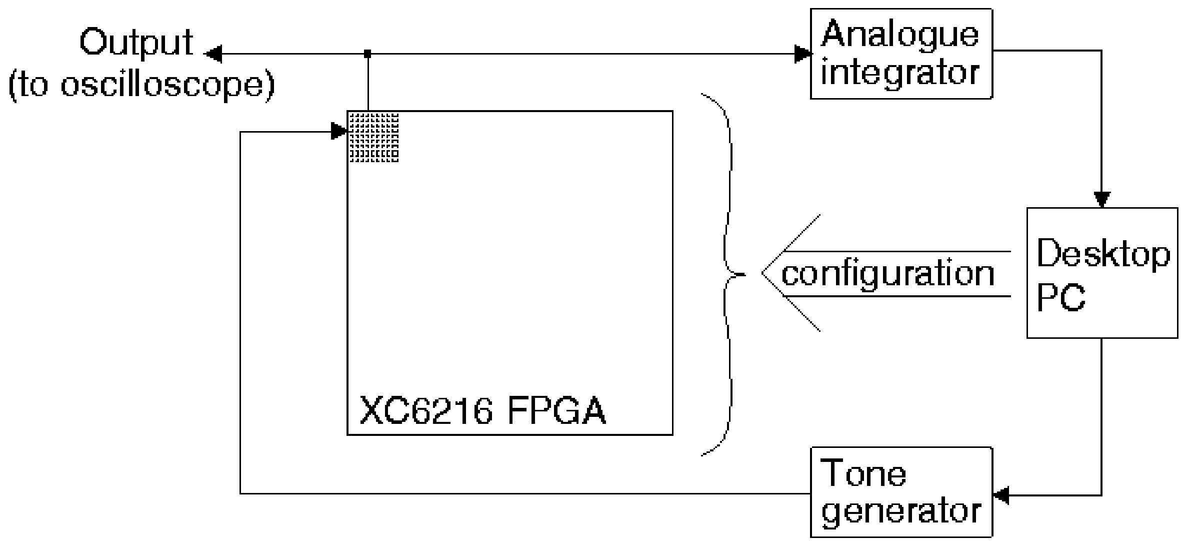


The Experiment

- Each cell has an 18-bit chromosome
  - Grouped into a linear bit-string genotype 1800 bits long
- Evolve a circuit to discriminate between square waves of 1kHz and 10kHz
  - Output +5V at one frequency, 0V at the other
- Genetic Algorithm (GA) parameters
  - Population size = 50 (randomly generated)
  - Elitism (single fittest individual survives)
  - Fittest individual expected twice the offspring as the median-ranked individual
  - Mutation rate = 2.7 mutations/genotype
- Configure and run on real hardware



Fitness Evaluation



- Five 500ms bursts of 1kHz square wave, five 500ms bursts of 10kHz square wave (random order, no gap)
- Reset integrator at beginning of each test tone
- Integrate output voltage over each test tone duration

Fitness Evaluation

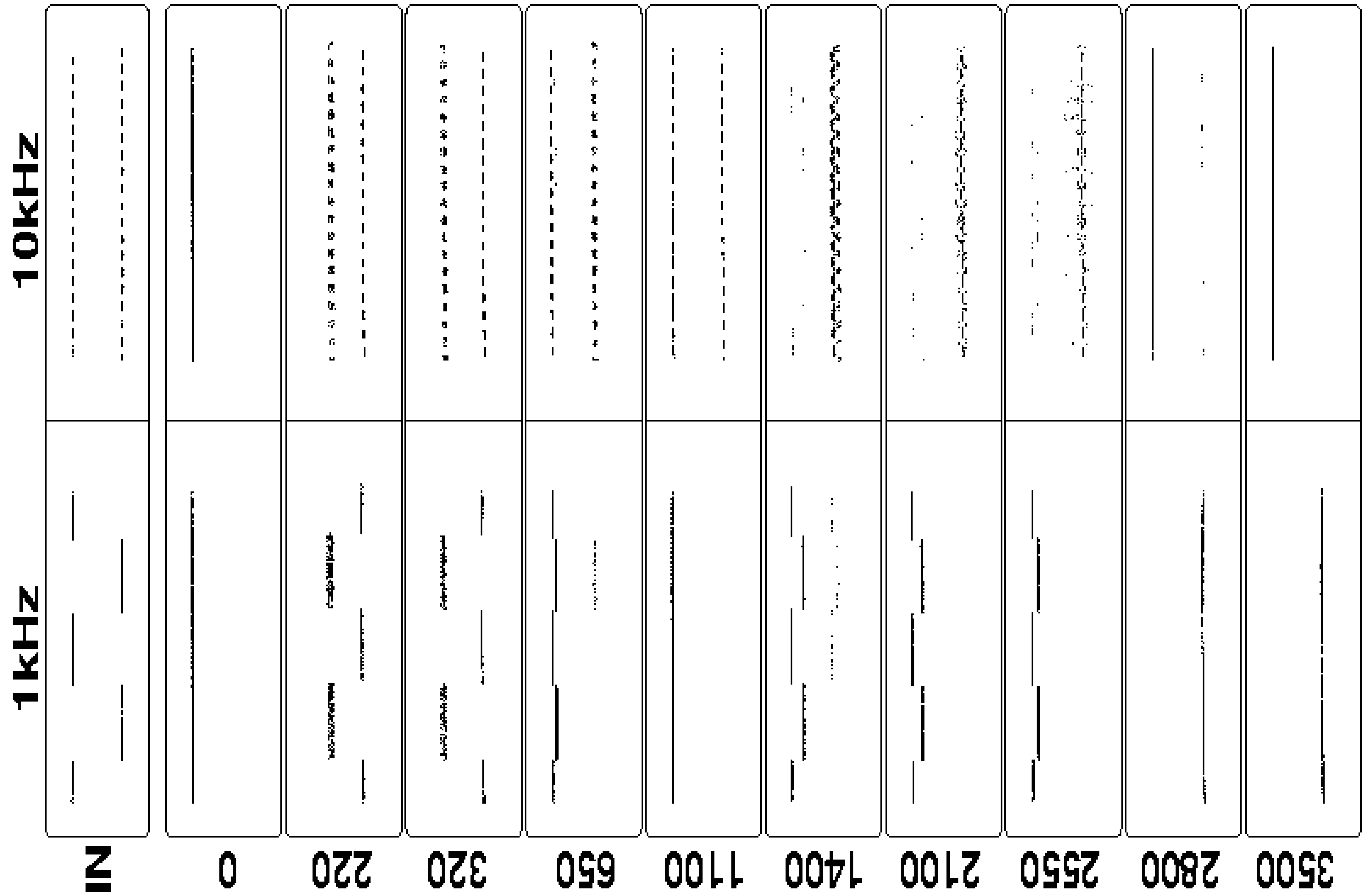
- i_t = integrator value at end of test tone *t*
- S_1 = set of five 1kHz test tones
- S_{10} = set of five 10kHz test tones

$$\text{fitness} = \frac{1}{10}\left|\left(k_1 \sum_{t \in S_1} i_t\right) - \left(k_2 \sum_{t \in S_{10}} i_t\right)\right|$$

$$\text{where}\{k_1 = 1/30730.746, k_2 = 1/30527.973\}$$

- Maximizes difference between the average output voltage during the two input test tones
  - Constants determined such that circuits that directly map input to output get zero fitness

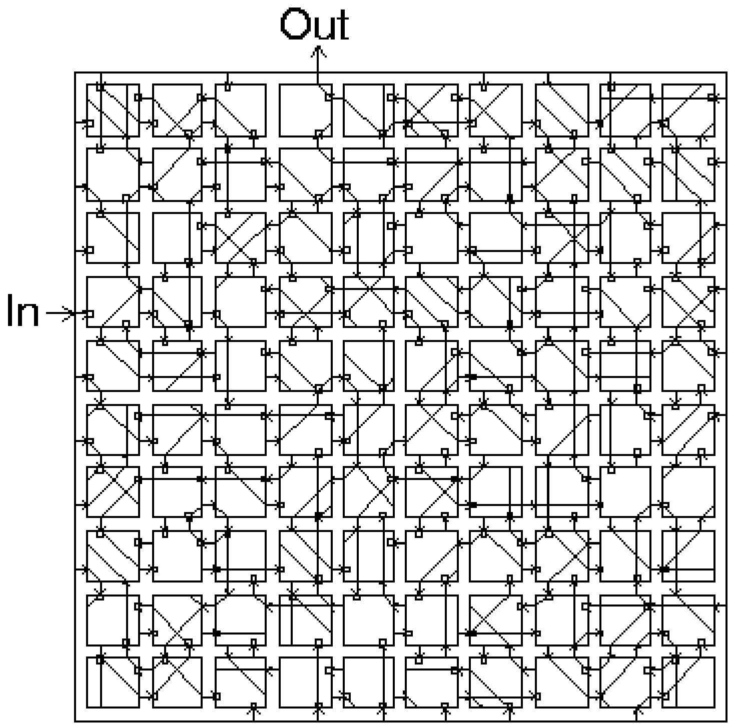
Results



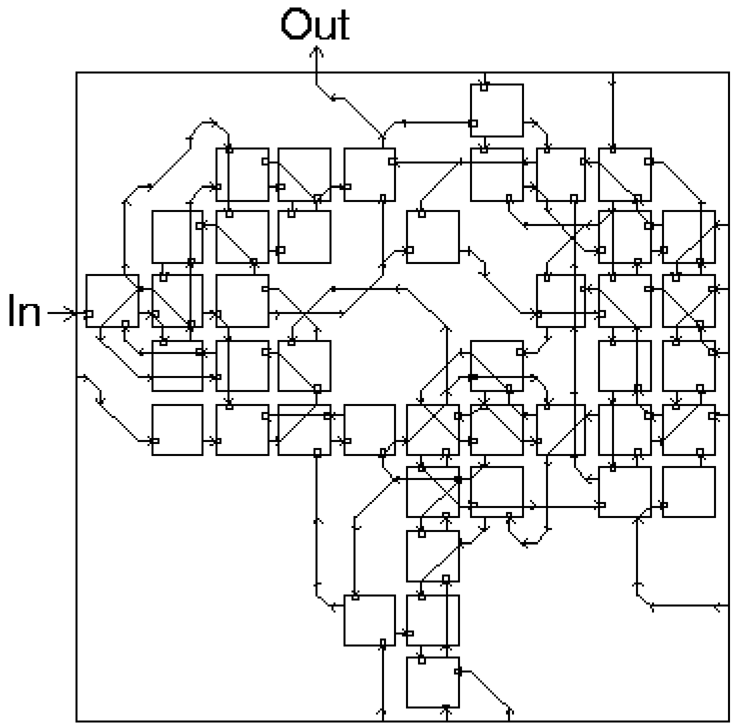
Caveats

- Analysis
  - No clear functional decomposition
  - Long-term stability is hard to prove
  - Possible that intermittent behavior has a long time-scale
  - Unknown failure modes
- Application
  - Must be insensitive to certain variations in implementation and environment
    - Thermal drift
    - Noise
    - Aging effects at the semiconductor level
    - Variations at the semiconductor level

Resulting Circuit



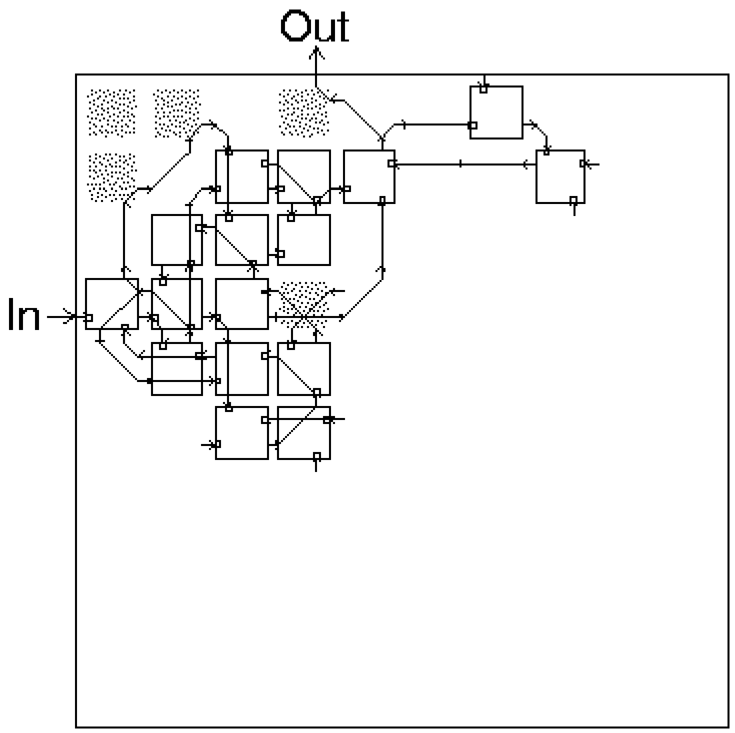
Full evolved circuit



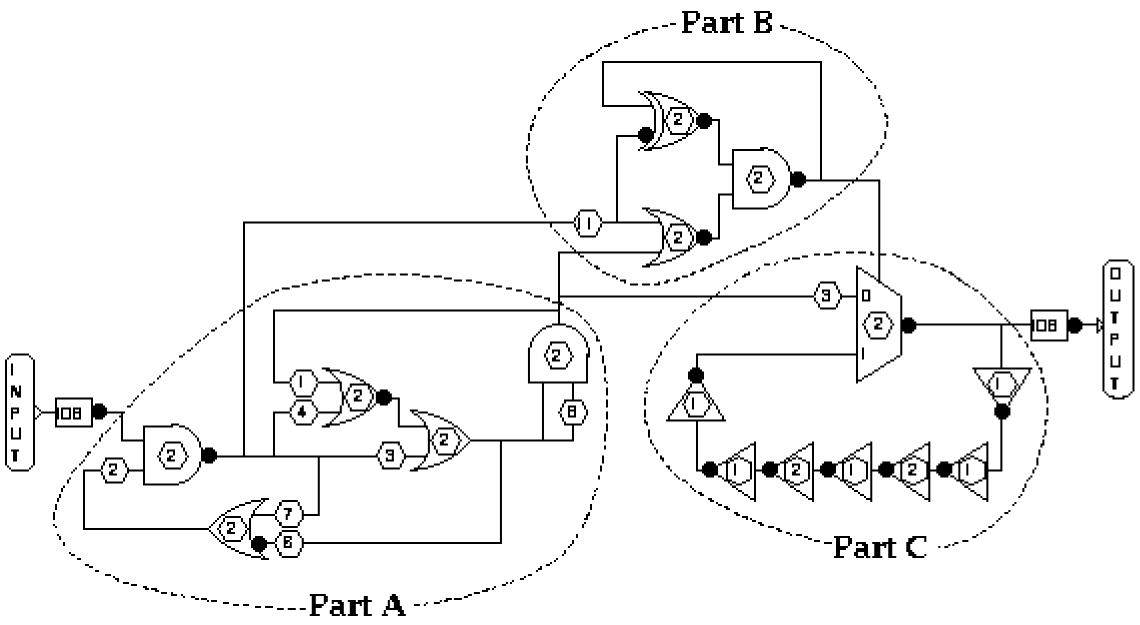
Pruned circuit

Functional Circuit

- **Random "unused" cells were clamped to random constant values**
- **Fitness was re-evaluated**
  - **Cell unclamped if performance degraded**
  - **Cell left clamped otherwise**
- **Iterated to build a maximal set of clamped cells without altering fitness**
- **Grey cells *cannot* be clamped even though there is no path to the output!**

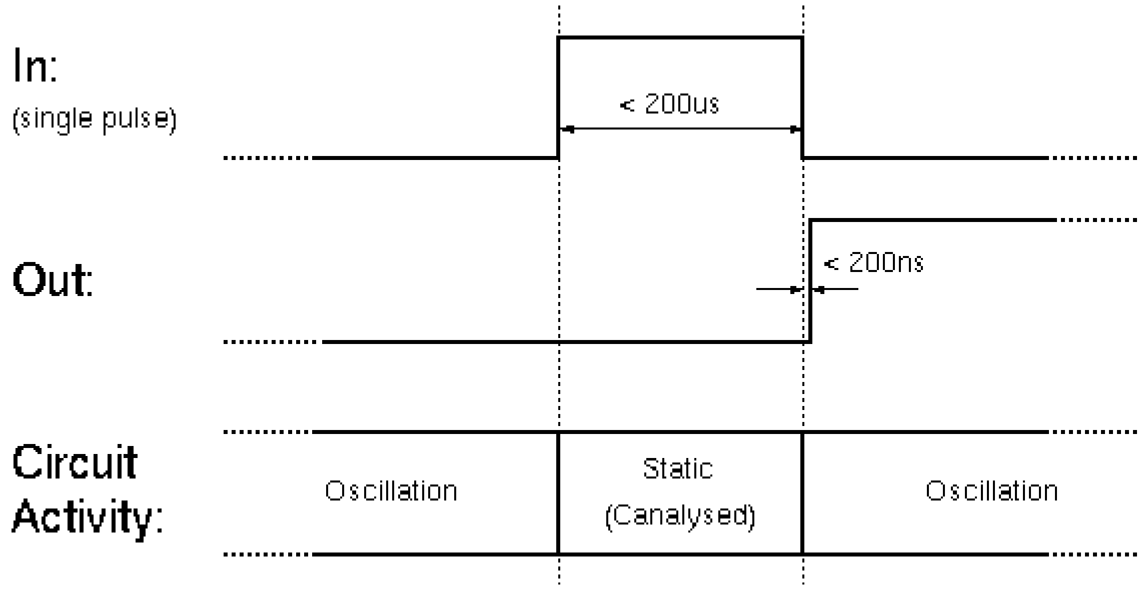


Circuit Diagram



- Within 20ns of an input ‘1’, gates in A and B switch to fixed states until input goes to ‘0’
- Part C has 9ns inversion delay

Circuit Behavior



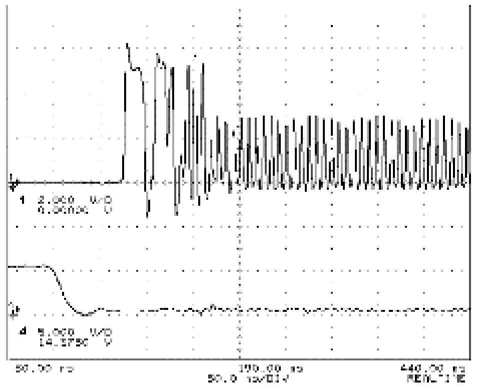
- During pulse A and B parts are static within 20ns
- Part C is also static (through observation), yet “knows” within 200ns of the end of the pulse, how long the pulse was
- Short pulses keep output high, longer pulses change output low

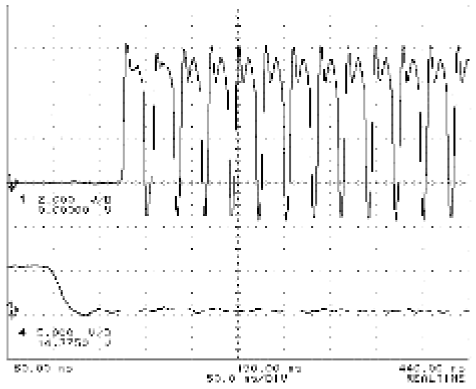
Analysis

- Probing
  - Power consumption at quiescent levels during pulse
  - Observed signals are in a stable state
- Simulation
  - PSPICE simulation verified transient upon entering static state
- Synthesis
  - Built circuit out of separate CMOS multiplexer chips--failed

Analysis

- Evolutionary history



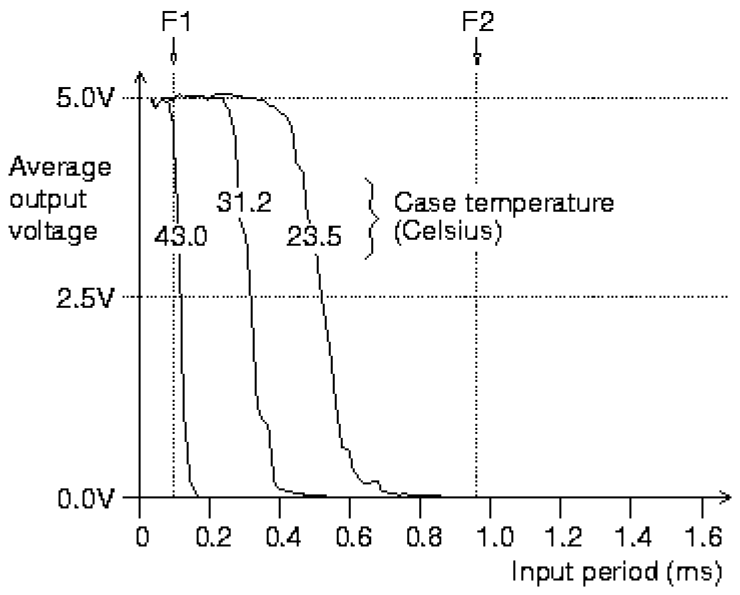


  - After pulse, output oscillates at one of two frequencies, depending on the length of the pulse (left: long, right: short)
  - Bistable oscillations present in Part A of final circuit
  - These oscillations are used by Parts B &C to derive a steady output according to the pulse width
  - Maybe due to charging/discharging of unknown parasitic capacitance?

Robustness

- Discriminate between >4.5kHz and <1.6kHz
- Vary frequency from 31.25kHz to .625kHz
- Vary temperature

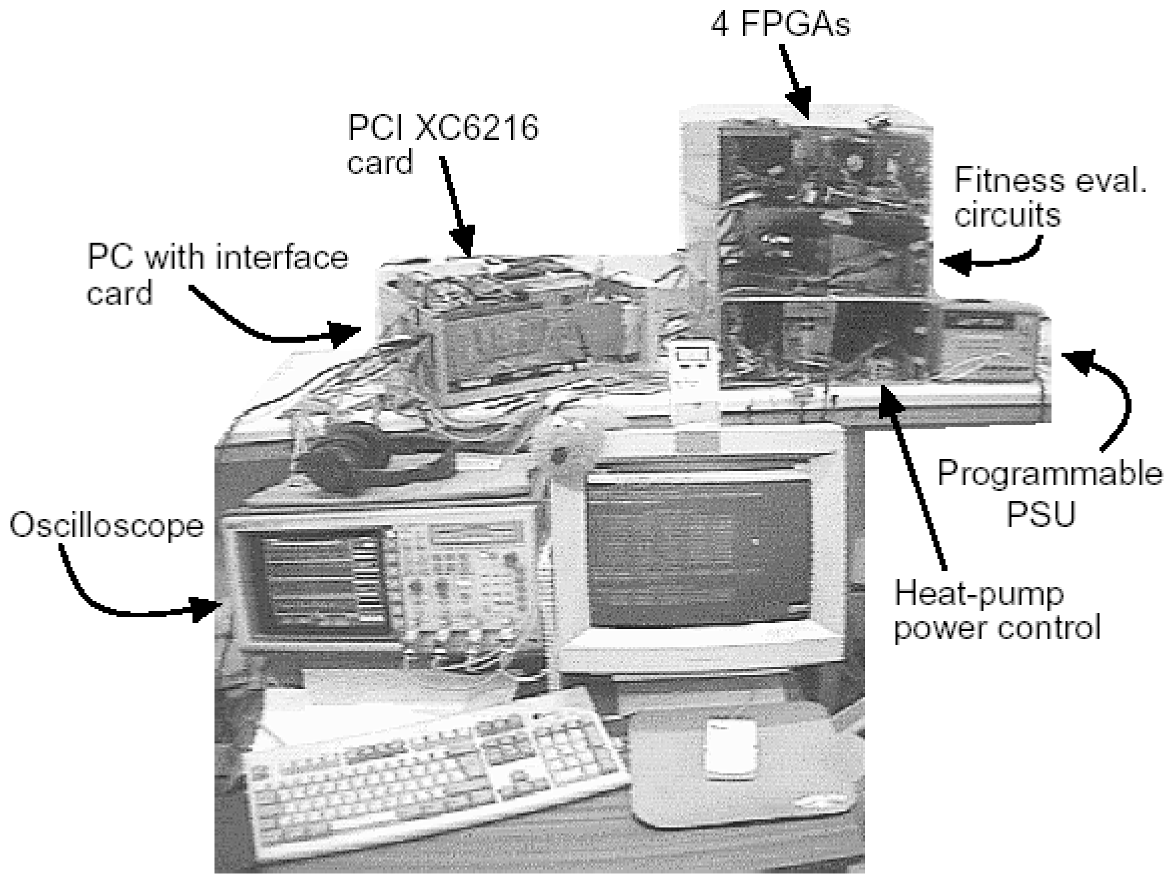
- Can discriminate well at nominal temperature
- Temperature has adverse effect on performance



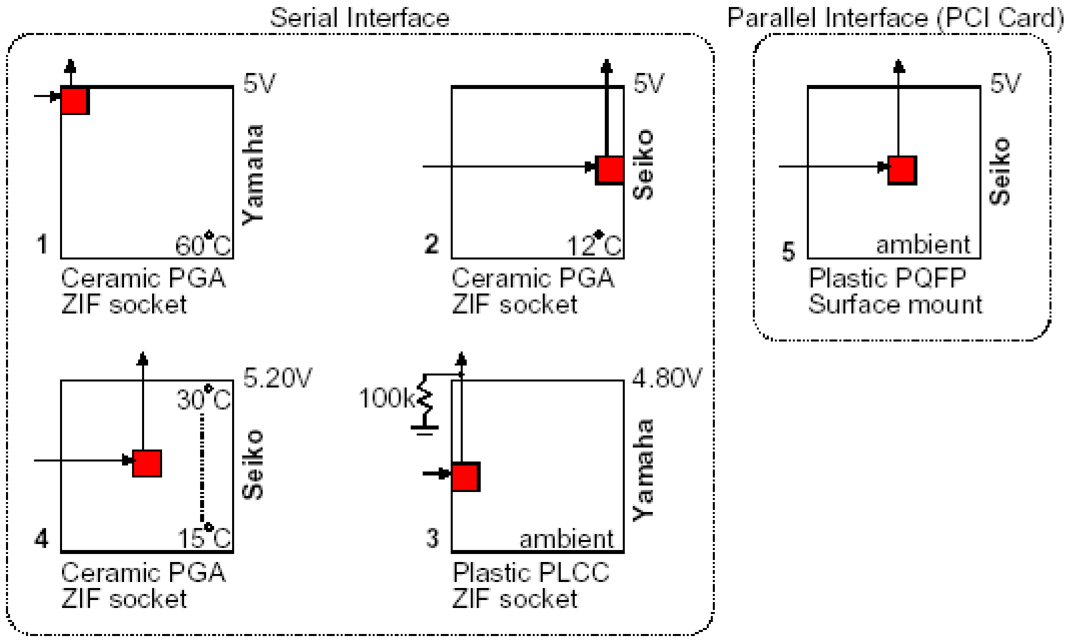
Evolving Robustness

- Can we evolve a circuit to work within a defined *operational envelope?*
- The Experimental Operational Envelope
  - Use five FPGAs in similar configuration to last experiment
  - Electronic surroundings
    - Shielding, pin connectivity
    - Circuit position
    - FPGA fabrication variations (Yamaha vs. Seiko)
    - FPGA Packaging
    - Temperature (ambient, constant cold, constant hot, gradient)
    - Power supply
    - Output pin resistive loading

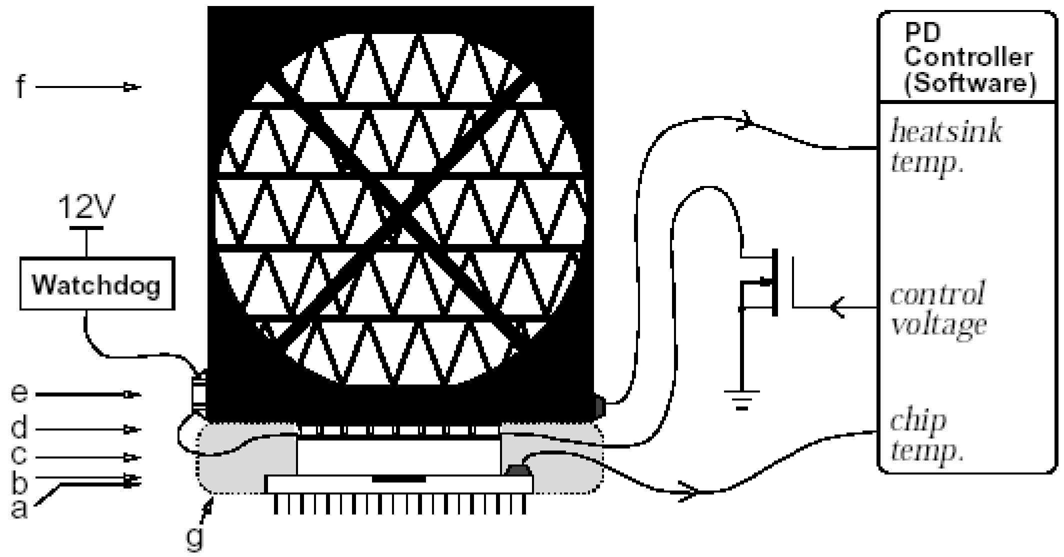
The Evolvatron Mk. I



Per FPGA Variations

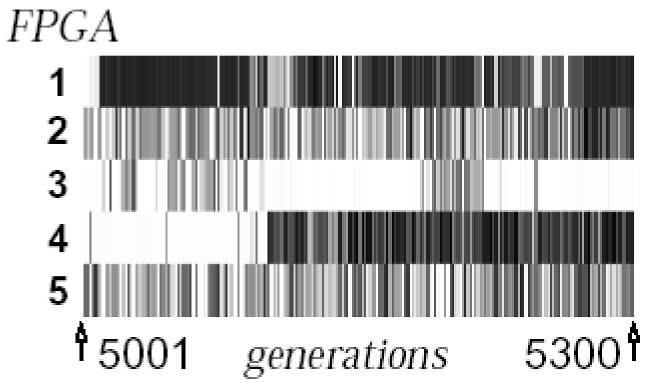


Hot and Cold FPGAs



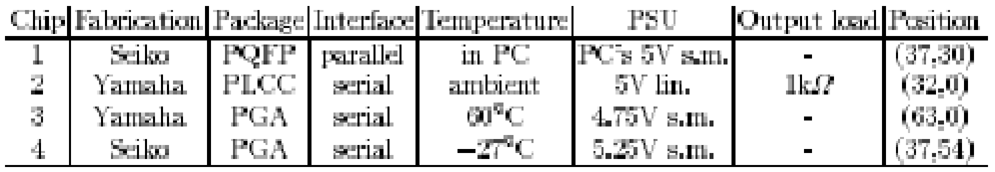
Results Mk. I

- Take generation 5000 population from previous experiment
- Continue evolution with overall fitness now being the mean of the individual fitnesses measured on the five different FPGAs
- This measures the population subject to new selection pressures
- Trying to find a *single* individual scoring well in *all* conditions



Evolvatron Mk. II

- Uses similar hardware setup, but different evolutionary algorithm

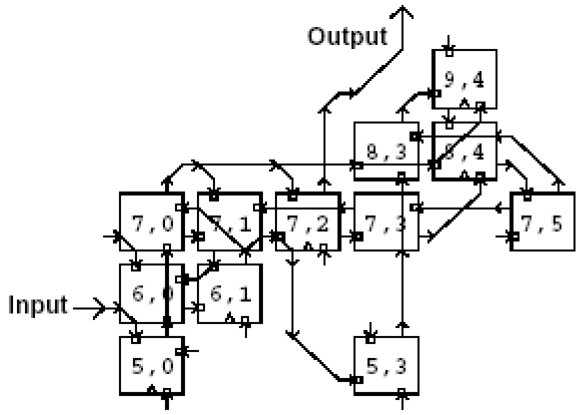


Chip	Fabrication	Package	Interface	Temperature	PSU	Output load	Position
1	Seiko	PQFP	parallel	in PC	PC's 5V s.m.	-	(37,30)
2	Yamaha	PLCC	serial	ambient	5V lin.	1kΩ	(32,0)
3	Yamaha	PGA	serial	60°C	4.75V s.m.	-	(63,0)
4	Seiko	PGA	serial	−27°C	5.25V s.m.	-	(37,54)

- Clock supplied as an environmental resource
- Randomly pick cell and randomly select mux within cell to reconfigure with mutation – do three times per parent

Results Mk. II

- Final circuit works near-perfectly on all of the FPGAs in all conditions
- Tested on six new FPGAs first frozen to –50C – worked perfectly all the way to room temperature
- Simulated perfectly in PSPICE
  - Does not rely on analog effects
  - Much more robust than earlier experiements



Conclusions

- Pros
  - Evolved circuits can be much more compact and power-efficient than human-created circuits
  - Intrinsic evolution allows us to explore and exploit physical dynamics of hardware without the need to understand the exact dynamic behavior
  - Evolving circuits can be exposed to robustness parameters and can evolve tolerances and adaptability
  - Evolution can take place without human intervention
  - Evolved circuits can teach designers new tricks
- Cons
  - Fitness function creation can be a non-trivial task
  - Intrinsic fitness evaluation can be time consuming
  - Very little understanding of fundamentals