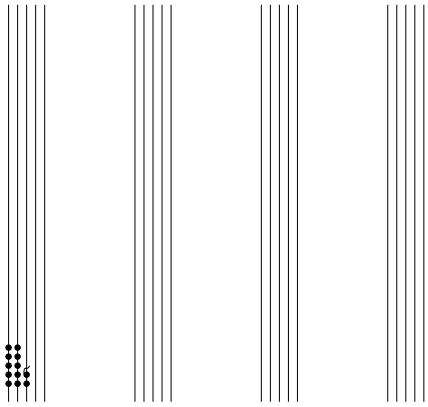


of the ideas put forward, by means of a simulation study using a highly abstract model of an FPGA. Finally, Section 4 describes a new evolvable hardware architecture, which is easy to build, but yet takes many of the important ideas on-board. The benefits of this architecture, and the underlying approach, are demonstrated by the real-world performance of this machine in controlling a robot.

2 Why Evolve in Hardware?

Under what conditions is the use of evolvable hardware beneficial, when compared with systems evolved in software simulation? This section identifies three areas: the first two are related to the raw speed of hardware, while the third is more profound and suggests powerful new kinds of system that have not been seen before.

2.1 Speed of Operation

behaviour: the clock is used so that the components are given time to reach a steady state before their condition is allowed to influence the rest of the system. I suggest (with the aid of the empirical evidence presented in the following sections) that such constraints should be abolished whenever they are a limitation on the potentially useful behaviour of an evolving hardware system. A designer carefully avoids "glitches," "cross-talk," "transients" and "meta-stability," but all of these things could be *put to use* by artificial evolution.

Evolution could also put to use properties of the hardware that the designer could *never* know about. For instance, a circuit may evolve to rely on some internal time-delays of an integrated circuit that are not externally observable. Even if there is a silicon defect, the system could evolve to use whatever function the "faulty" part happened to perform. This raises a fundamental problem: a circuit that is evolved for a particular evolvable hardware system (a certain FPGA chip, for example) may not work on a different system that is nominally identical — no two silicon chips are the same. There are several ways in which this could be avoided. The circuits could be evolved to be robust to perturbations in some properties that vary from chip to chip (by altering the chip's temperature or power supply during evolution, for example). Evolution could be forced to produce building blocks that are repeatedly used in the circuit, and would therefore be insensitive to characteristics that varied across one chip. Evolution could evaluate a configuration on more than one piece of reconfigurable hardware when judging its quality (this can also be done using a single reconfigurable chip that can instantiate the same circuit in several different ways, e.g. by using an FPGA's rotational symmetry). Finally, it could be accepted that further adaptation will have to take place each time a configuration is transferred from one reconfigurable device to another [14, 15, 16, 17].

The next two sections of this paper will provide experimental evidence for the ideas I have put forward here. Firstly, I present a simulation of the evolution of an FPGA configuration, which demonstrates that evolution can produce circuits optimised for a particular implementation, and in the absence of modularisation and clocking constraints. Then I describe a real evolved hardware control system that controls a real robot, and was produced according to the above rationale, demonstrating its benefits.

3 A Millisecond Oscillator from Nanosecond Logic Gates

Abandoning the external clock can reap even more rewards than were mentioned above. A clocked digital system is a finite-state machine, whereas an unclocked (asynchronous) digital system is not. To describe the state of an unclocked circuit, the temporal relationships between its parts must be included. These are continuously variable analogue quantities, so the machine is not finite-state. This theoretical point gives a clue to a practical advantage: in an unclocked digital system, it is possible to perform analogue operations using the time dimension, even when the logic gates assume only binary values (see for example, the *pulse stream* technique [21, 22]).

In the previous section, I argued that when producing circuits by evolution rather than design, the use of a clock is often an *unnecessary* limitation on the way in which the natural dynamics of the components can be used to mediate robot behaviour. This is not always the case — electronic components usually operate on time-scales much smaller than would be useful to a robot; unless the system can evolve such that the overall behaviour of the components (when integrated into the sensorimotor feedback loop of the robot) is much slower than the behaviour of individual components, then a clock (perhapsodar

(Node 0 was a special "ground" node, the output of which was always clamped at logic zero.) This encoding is based on that used in [2]. The source of each input was specified by counting forwards/backwards along the genotype (according to the 'Direction' bit) a certain number of segments (given by the 'Length' field), either starting from one end of the string, or starting from the current segment (dictated by the 'Addressing Mode' bit). When counting along the genotype, if one end was reached, then counting continued from the other.

Name	Symbol
BUFFER	
NOT	
AND	
OR	
XOR	
NAND	
NOR	
NOT-XOR	

(a)

Bits	Meaning
0-4	Junk
5-7	Node Function
	Pointer to First Input
8	Direction
9	Addressing Mode
10-15	Length
	Pointer to Second Input
16	Direction
17	Addressing Mode
18-23	Length

(b)

Table 1. (a) Node functions, (b) Genotype segment for one node.

At the start of the experiment, each node was assigned a real-valued propagation delay, selected uniformly randomly from the range 1.0 to 5.0 nanoseconds, and held to double precision accuracy. These delays were to be the input-output delays of the nodes during the entire experiment, no matter which functions the nodes performed. There were no delays on the interconnections. To commence a simulation of a network's behaviour, all of the outputs were set to logic zero. From that moment onwards, a standard asynchronous event-based logic simulation was performed [19], with real-valued time being held to double precision accuracy. An equivalent time-slicing simulation would have had a time-slice of 10^{-24} seconds, so the underlying synchrony of the simulating computer was only manifest at a time-scale 15 orders of magnitude smaller than the node delays, allowing the *asynchronous* dynamics of the network to be seen in the simulation. A low-pass filter mechanism meant that pulses shorter than 0.5ns never happened anywhere in the network.

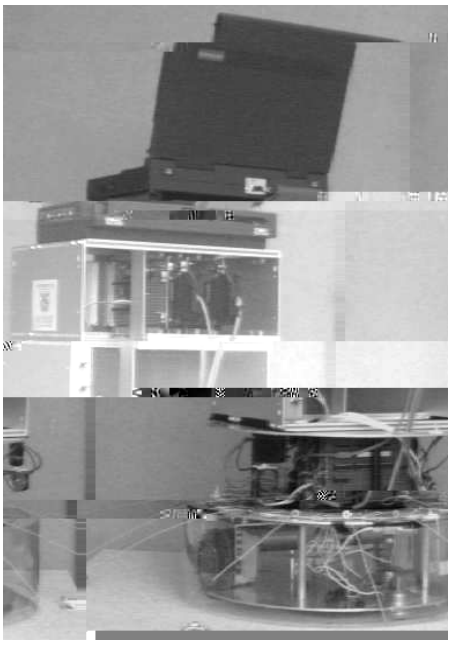
The objective was for node number 100 to produce a square wave oscillation of 1kHz, which means alternately spending 0.5×10^{-3} seconds at logic '1' and at logic '0'. If k logic transitions were observed on the output of node 100 during the simulation, with the

The experiment succeeded. Figure 2 shows that the output after 40 generations was approximately $4\frac{1}{2}$ thousand times slower than the best of the random initial population, and was six orders of magnitude slower than the propagation delays of the nodes. In fact, fitness was still

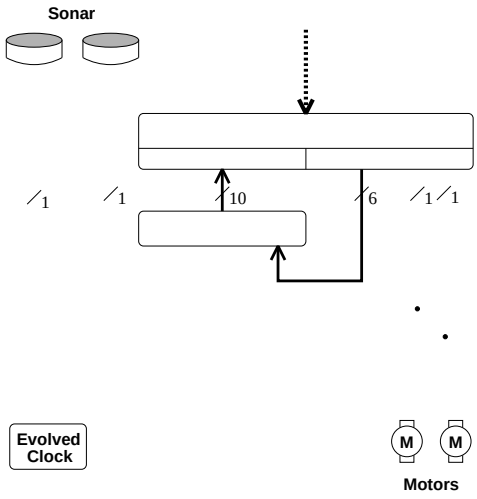
This simulation, although quite an unrealistic model of the evolution of a real FPGA configuration, has shown how evolution can assemble high speed components to produce behaviour on a time-scale that approaches that useful to a robot. It exploits the characteristics of the implementation, and does not require the imposition of spatial or temporal constraints such as modularisation or clocking. The style of solution adopted (the beating of spike trains) is an analogue operation over the time axis, and would have been more difficult in a discrete time system.

4 A Real Evolved Hardware Robot Controller

In this experiment, a real hardware robot control system was evolved for wall-avoidance behaviour in an empty 2.9m×4.2m rectangular arena, using sonar time-of-flight sensing. The two-wheeled robot ("Mr Chips," Figure 4(a)) has a diameter of 46cm, and a height of 63cm. For this scenario, its only sensors were a pair of fixed sonar heads pointing left and right.



(a)



contents would hold the next-state and output variables corresponding to each present-state and

wheels' angular velocities were measured, and used by a real time simulation of the motor charac-

When it is remembered that the DSM receives the raw echo signals from the sonars and directly drives the motors (one of which happens to be more powerful than the other), with only two internal state variables, then this performance in surprisingly good. It is not possible for the DSM directly to drive the motors from the sonar inputs (in the manner of Braitenberg's "Vehicle 2" [1]), because

References

1. Valentino Braitenberg. *Vehicles : Experiments in Synthetic Psychology.* MIT Press, 1984.
2.