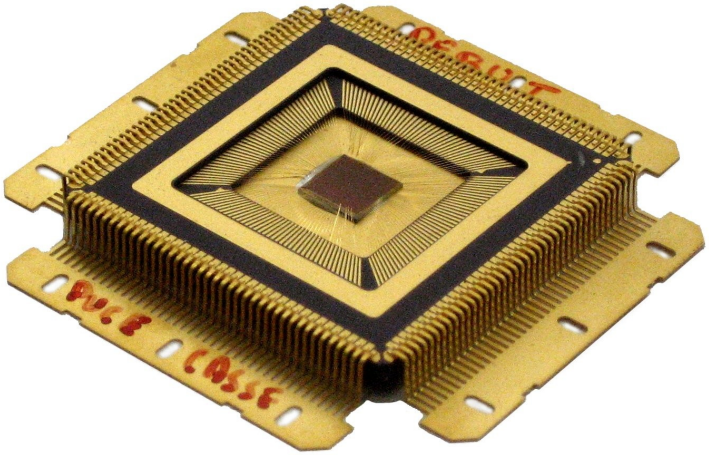
Evolving Digital Hardware
EHW Module 2008



Andy Greensted
Department of Electronics
ajg112@ohm.york.ac.uk

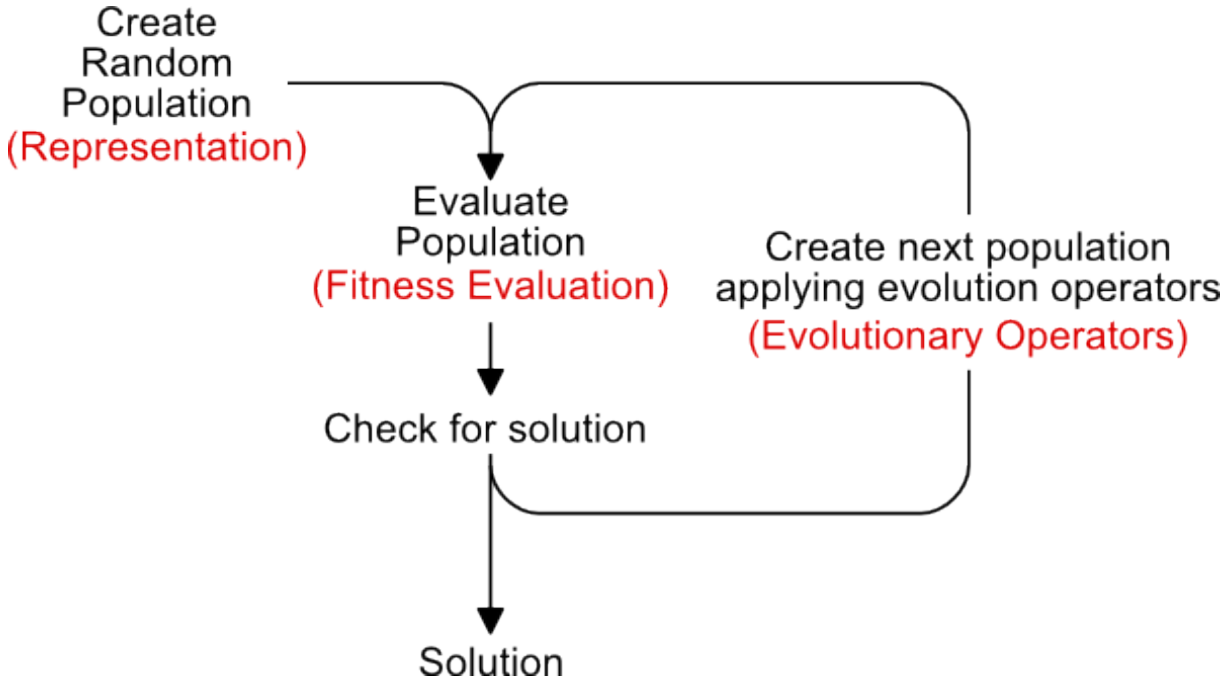
www.bioinspired.com/users/ajg112/teaching/evoHW.shtml

Lecture 1

Lecture Overview

- Lecture 1
  - An introduction to evolving digital circuits
- Lecture 2
  - Overview of the labs
  - FPGAs and their role in EHW
- Lecture 3
  - CGP
  - Case studies 1
- Lecture 4
  - Case studies 2

Evolution Overview

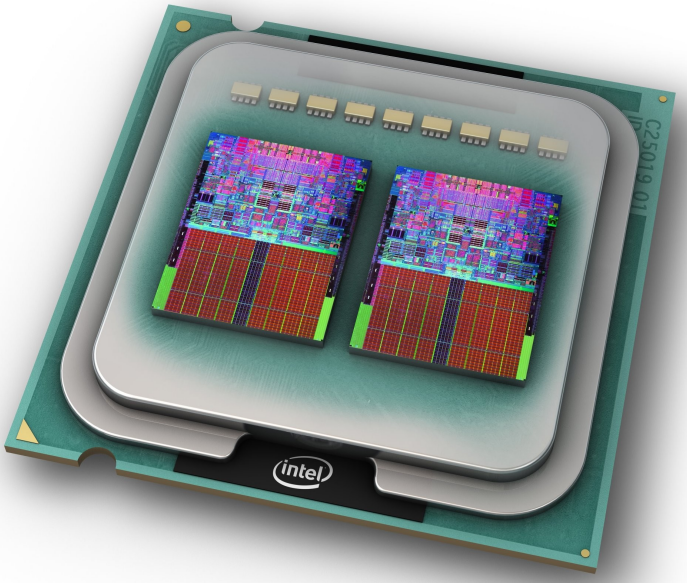


- Evolvable hardware uses the principle of Darwinian Evolution to create electronic circuits.
- The important factors are:
  - Representation
  - Fitness Evaluation
  - Evolutionary Operators

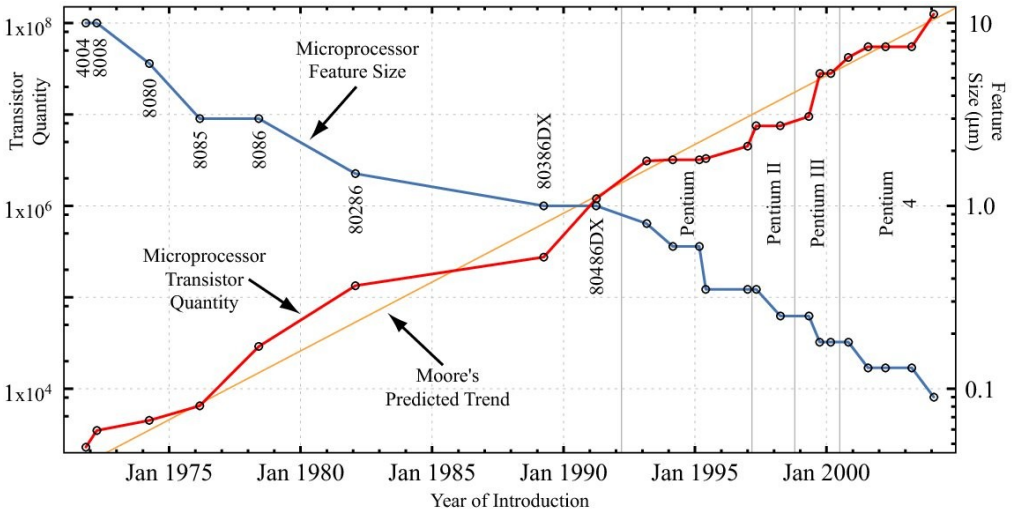
Motivation - Why use EHW

- The development of electronic circuits has so far closely followed Moore's Law
- Increasing complexity
  - More transistors
  - Greater Density

Can evolution help with the complexity problem?



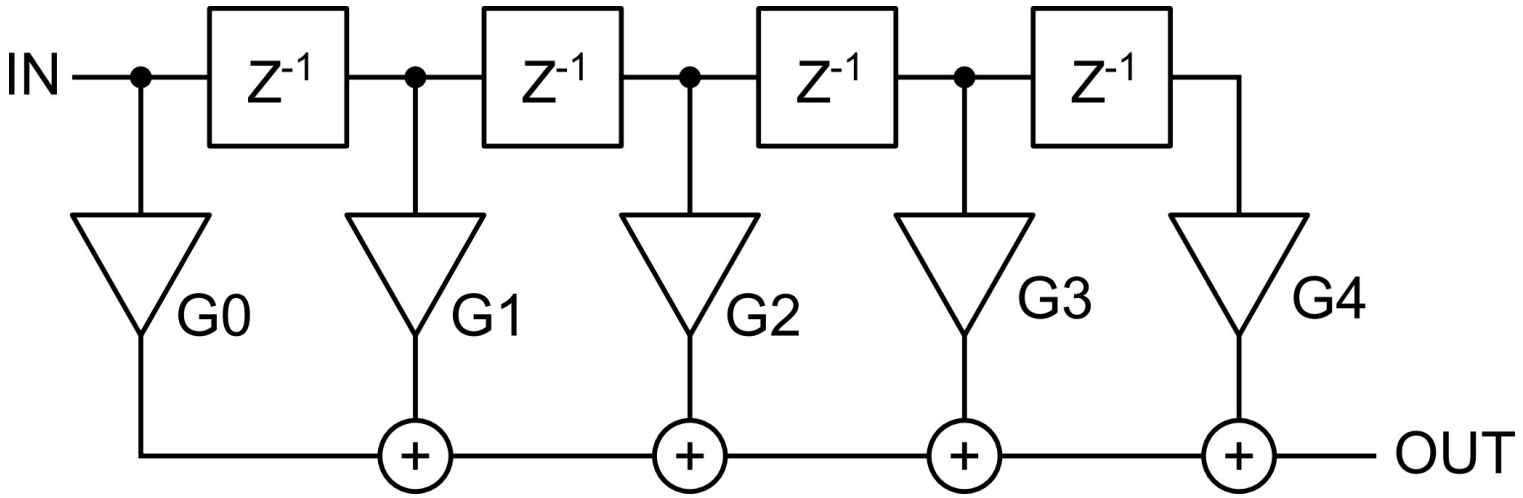
Intel® Core™2 Extreme
Quad-core Processor
(≈ 580 Million Transistors)



Moore's Law

What do we mean by Evolving Digital Hardware 1

You can use evolution to search the parameter space of digital circuits.



IIR Digital Filter

The gain of the multiplier stages provide a set of parameters that can be adjusted using an evolutionary process.

However, evolution just using parameter optimisation is limited by the scope of the fixed circuit structure.

What do we mean by Evolving Digital Hardware 1

• Evolution is quite good at performing parameter optimisation. Usually the search space is quite small.

However

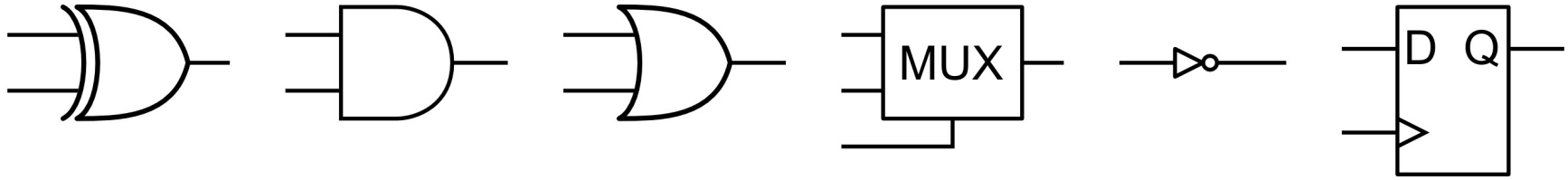
• Normally the operation of a human designed circuit is understood.

• Formulae are available to determine the required parameters for a particular set of circuit characteristics.

• Therefore, do we need evolution to find new parameter configurations?

What do we mean by Evolving Digital Hardware 2

Evolution can also be used to create whole circuits, controlling the selection and connectivity of electronic components.



In digital circuits, this is normally logic gates. Evolution can find novel circuits, from which new design techniques can be learnt.

It is also possible to evolve the parameters of these gates:

- Output drive strength
- Output switching speed

Although, these aren't normally taken into consideration.

As the evolution of component choice & connectivity of digital circuits can potentially create novel circuits it is the more widely researched approach.

Intrinsic & Extrinsic Evolution

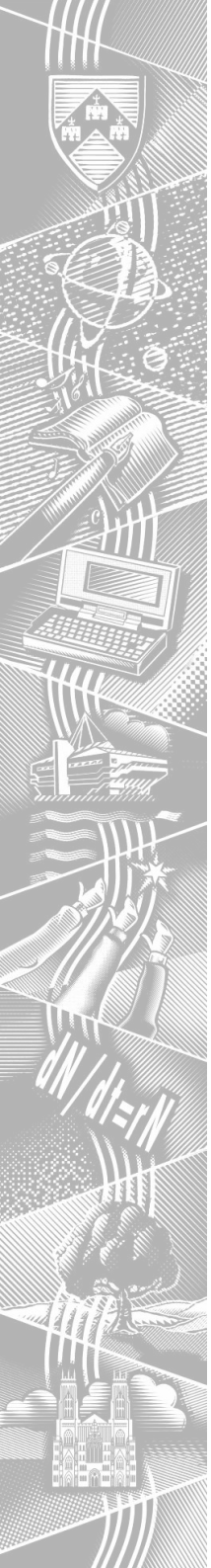
- A major choice in EHW is whether fitness evaluation is performed using a real circuit or simulation:
  - Intrinsic - Fitness evaluation using the target circuitry
  - Extrinsic - Fitness evaluation in simulation

Extrinsic (Simulation) Evolution

- Can be simpler to add the fitness evaluation stage in the evolution loop
- Can be more flexible, easier to try different circuit configurations
- Safer, no chance of destroying hardware by testing 'bad' candidate solutions

Intrinsic (In Hardware) Evolution

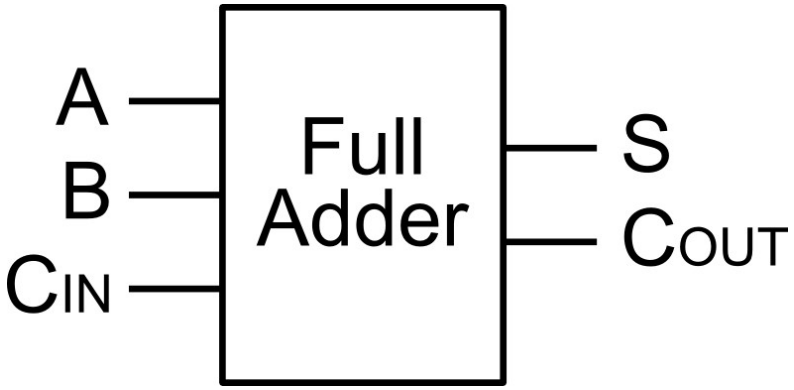
- No problems with moving a simulated solution into hardware.
- Evolution can be allowed to take advantage of the 'physics' of the target circuitry.
- For larger circuits, it could be faster



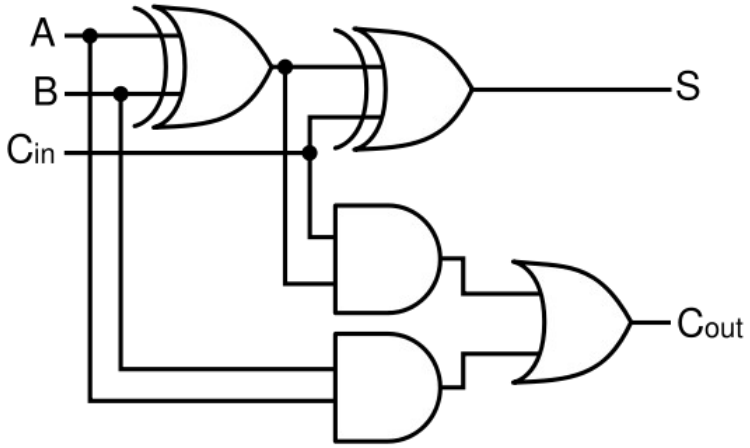
Part 2

Examples of Evolved Circuits

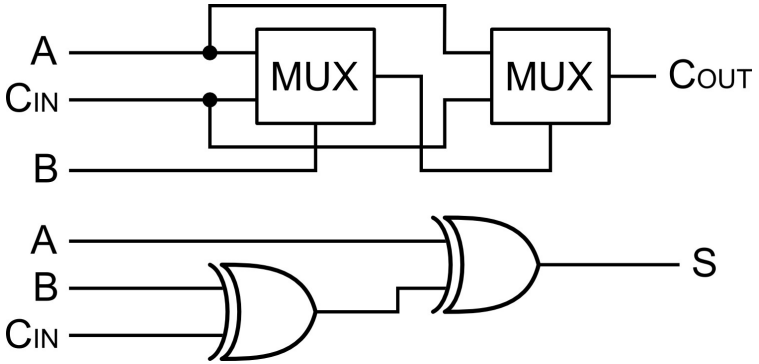
Evolution of a Simple Circuit

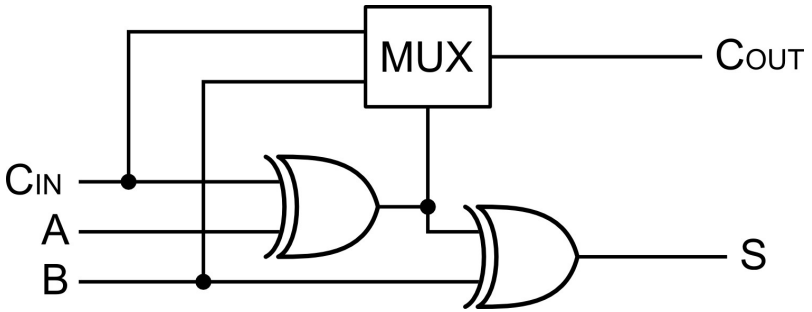


Full Adder Symbol



Conventional Full Adder Circuit

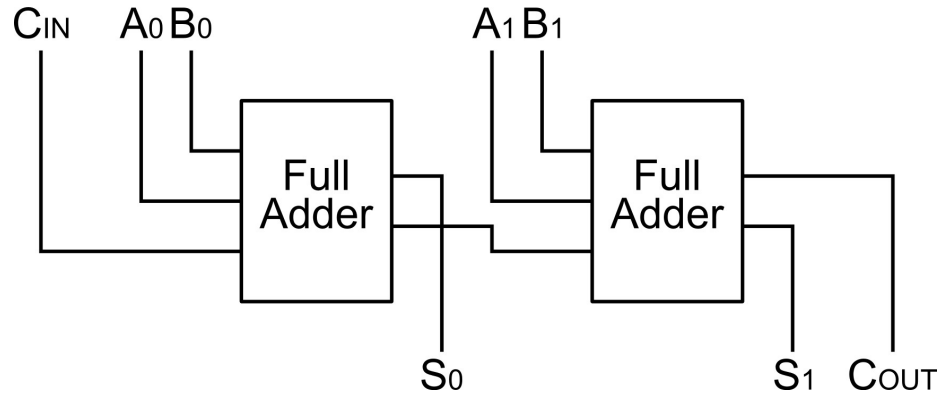




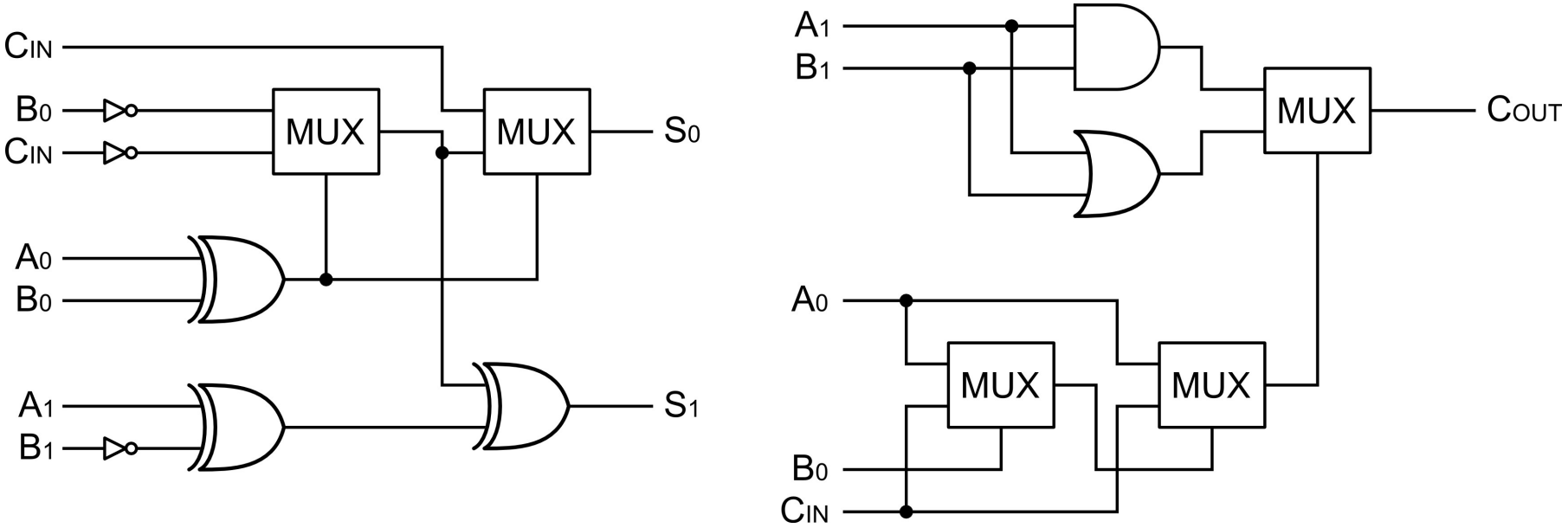
Evolved Full Adders

The evolved circuits are of equal complexity as traditional full adders. There is no real advantage in their use.

An Evolved 2bit Adder



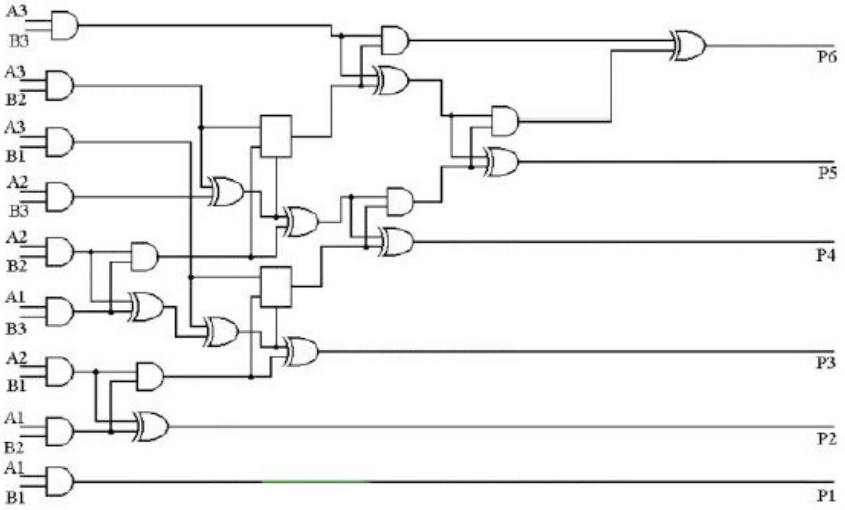
Conventional 2 bit Adder
10 Gates

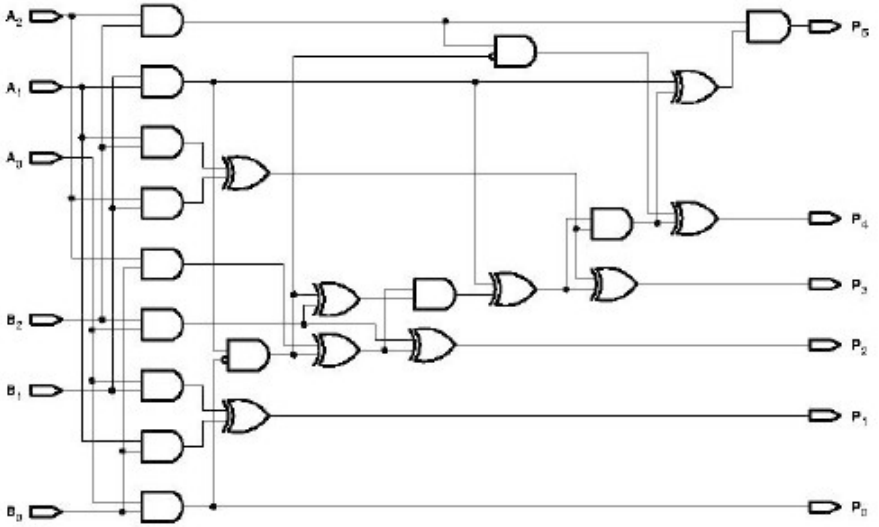


Evolved 2 bit Adder: 10 Gates

Depending on the implementation, multiplexers can contain more transistors, making the evolved circuits larger.

3 bit Multiplier





Conventional

26 Gates
Maximum of 6 gate delays

Faster Circuit, but larger

Evolved

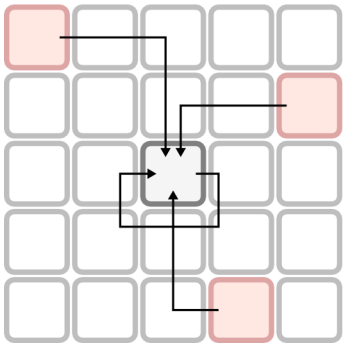
23 Gates
Maximum of 8 gate delays

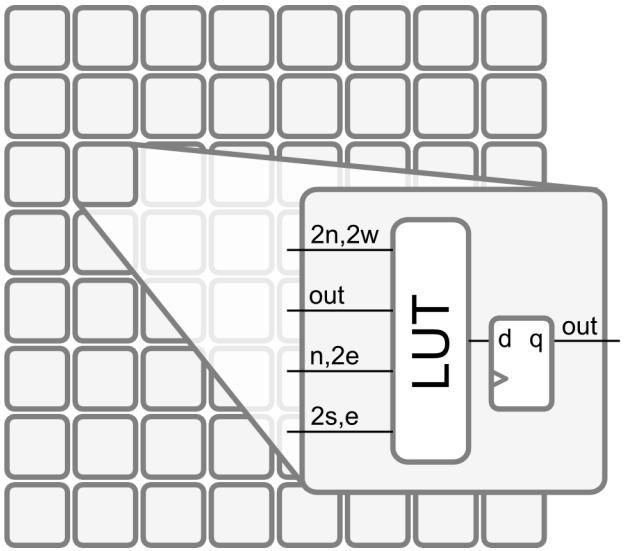
Smaller Circuit, but slower

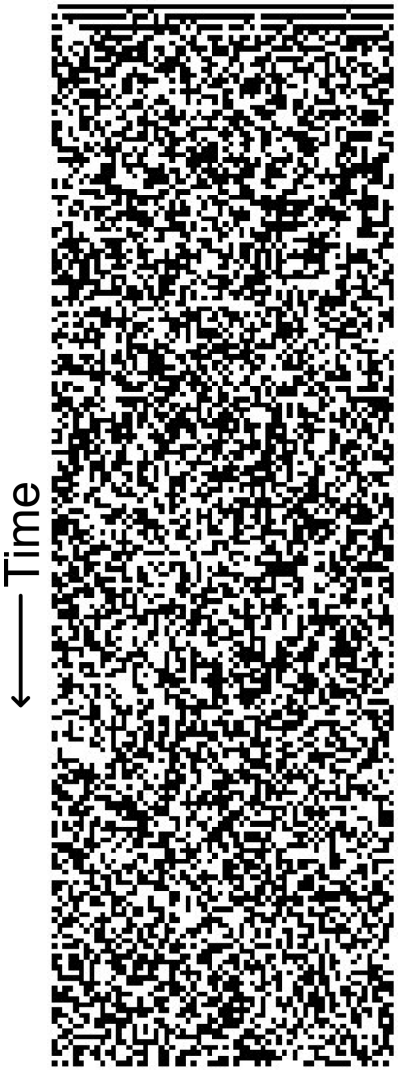
In this case evolution has found a solution that is potentially useful. **However,** the largest evolved multiplier contains only 100s of transistors, no way near the 580 Million of a human designed processor.

Pseudo Random Number Generator

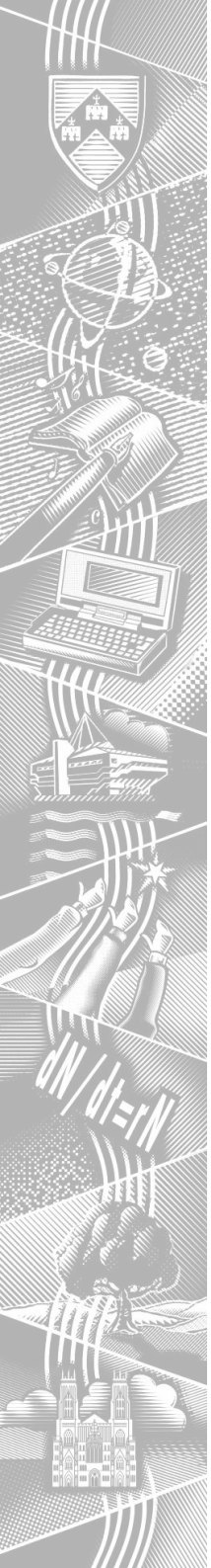
- Cellular automata based random number generator
- 8x8 cell array
- Each cell is a 4 Input registered lookup table
- Evolved next state logic and cell connections







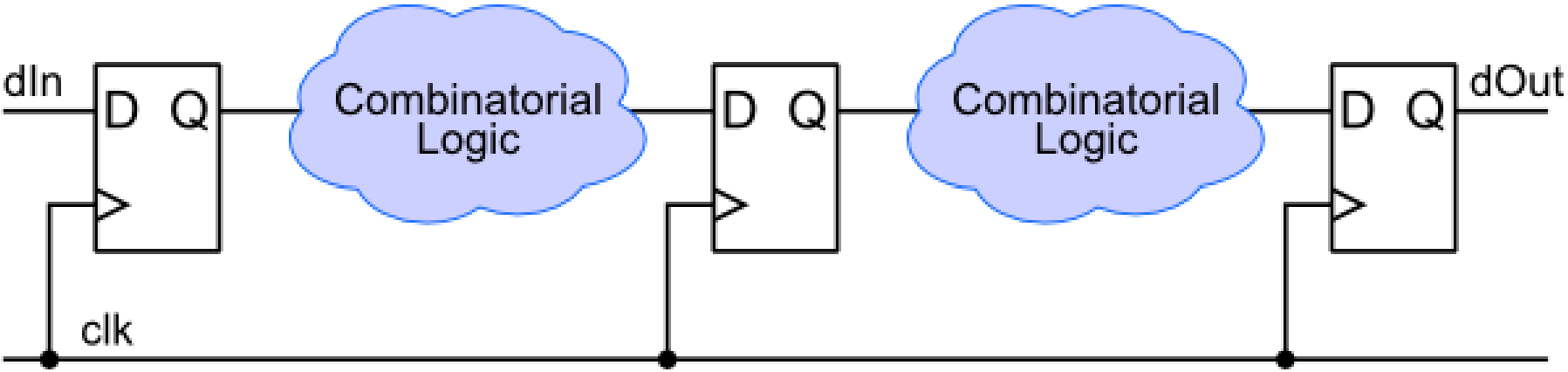
FPGA Implementation of Neighborhood-of-Four Cellular Automata Random Number Generators
Barry Shackleford, Motoo Tanaka, Richard Carter, Greg Snider
10th International Symposium on Field Programmable Gate Arrays, Monterey, California, USA, 2002



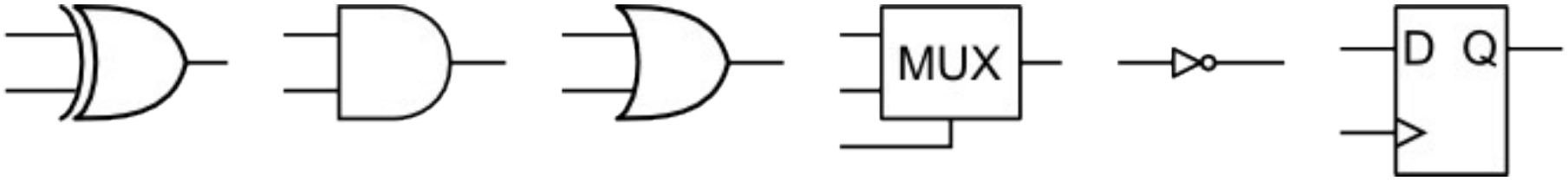
Part 3

An Architecture for Evolving Digital Circuits

Digital Basics

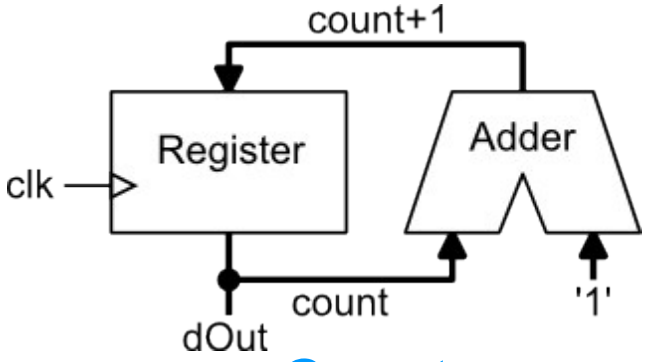


Sequential circuits use a clock to drive data between registers. Logic gates between registers operate on the data.

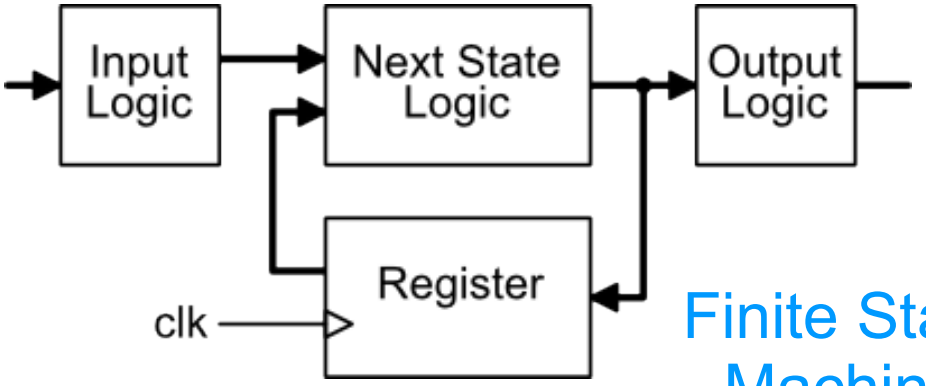


Most digital circuits are constructed using this format.

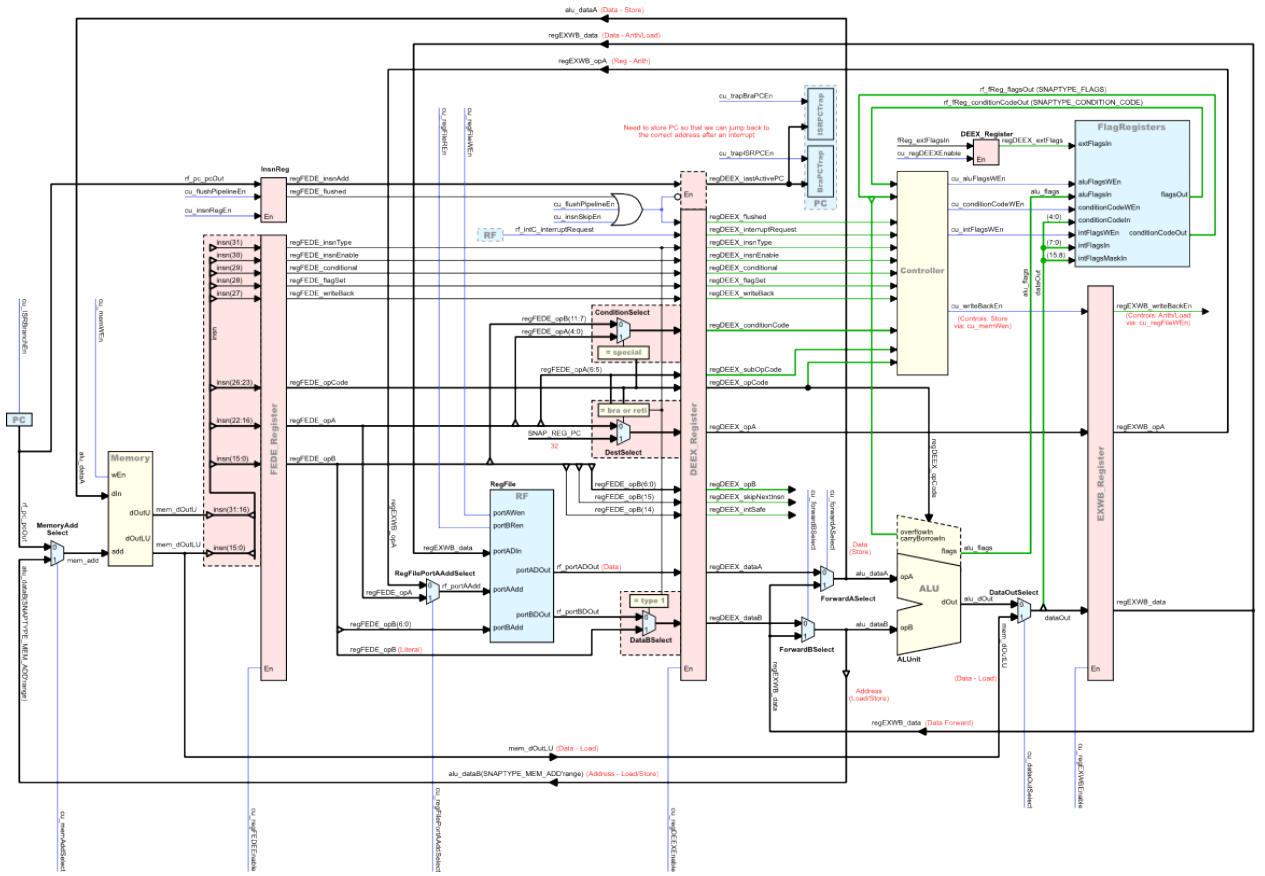
Example Digital Circuits



Counter



Finite State Machine



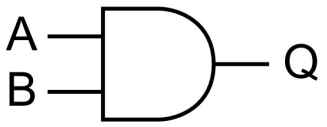
Microprocessor

These all follow the register and logic structure

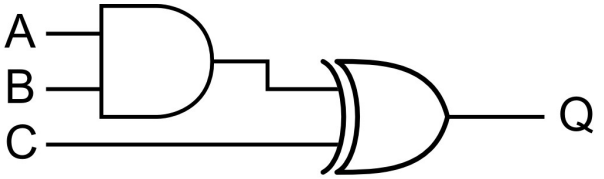
RAM, LUTs and Logic Tables

How can we make configurable combinatorial circuits?

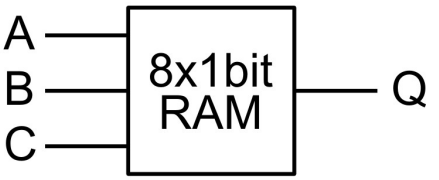
First, how can we make configurable gates?



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

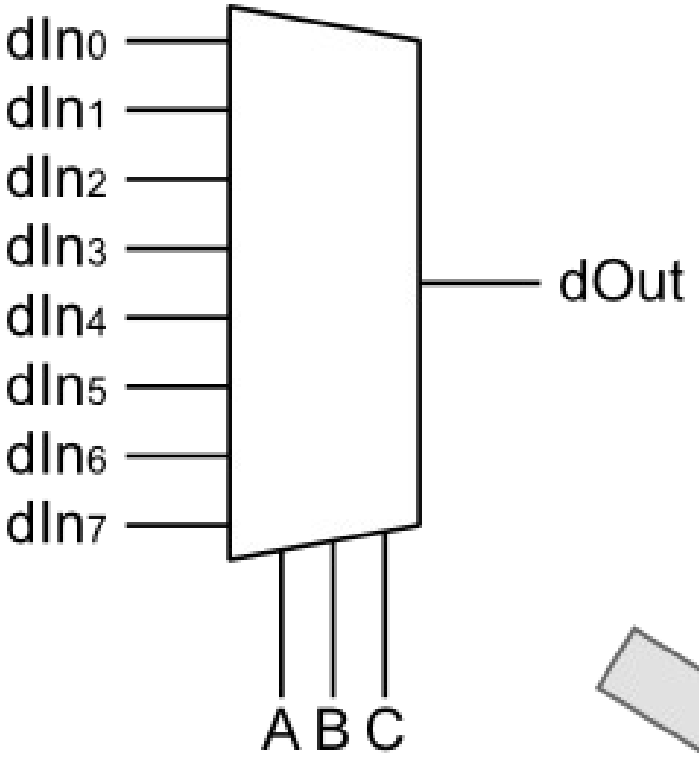


A	B	C	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



The same functionality of gates can be constructed using small blocks of RAM. The contents of the RAM defines the logic function. RAM used in this way is often referred to as a **Lookup Table (LUT).**

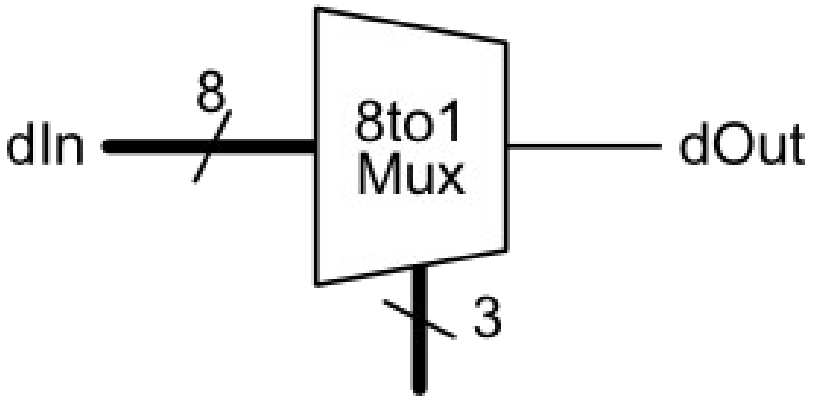
Circuit Connections



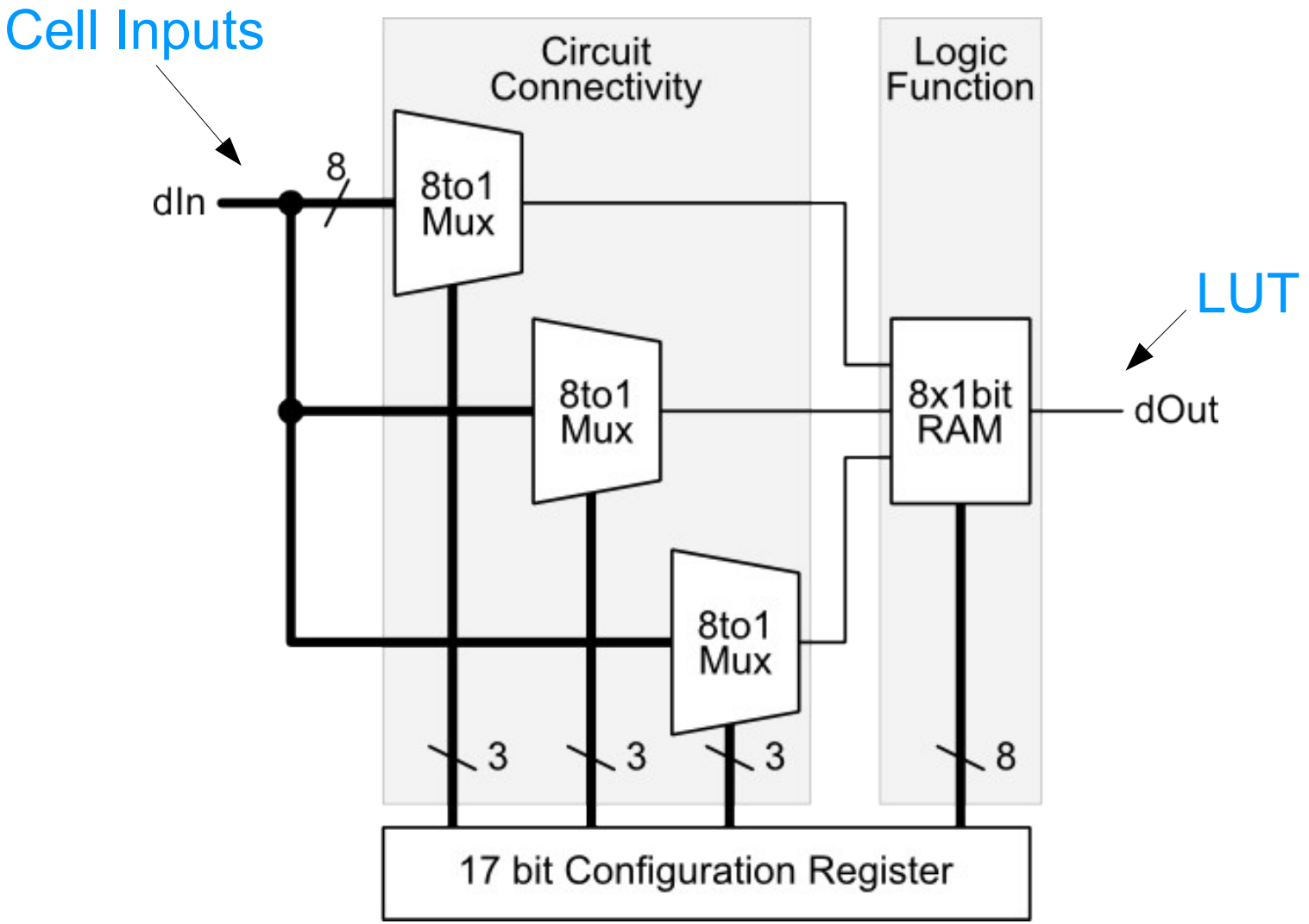
A	B	C	dOut
0	0	0	dIn0
0	0	1	dIn1
0	1	0	dIn2
0	1	1	dIn3
1	0	0	dIn4
1	0	1	dIn5
1	1	0	dIn6
1	1	1	dIn7

An 8 to 1 multiplexer

This component allows one of a selection of inputs to be connected to the output.

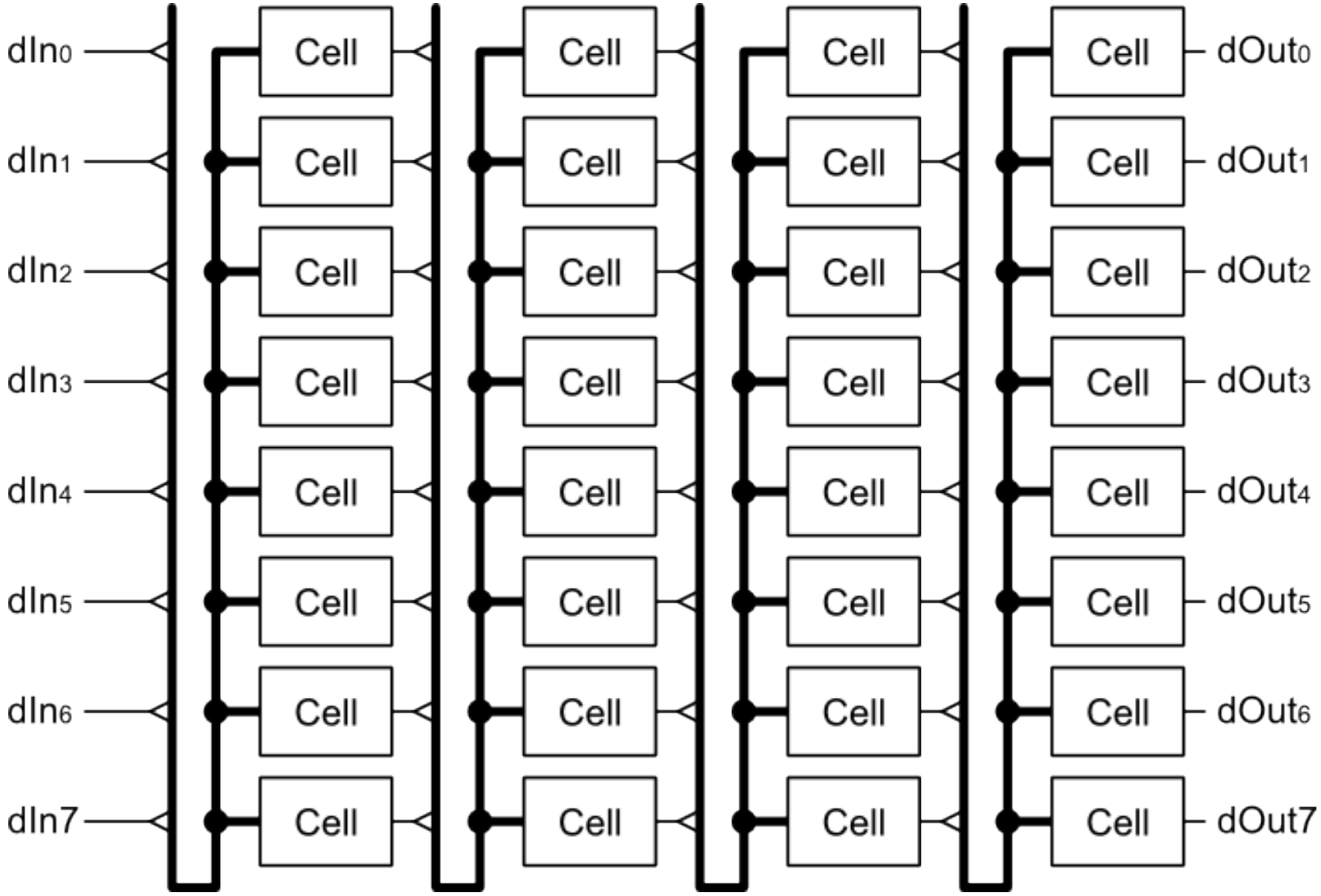


A Configurable Cell



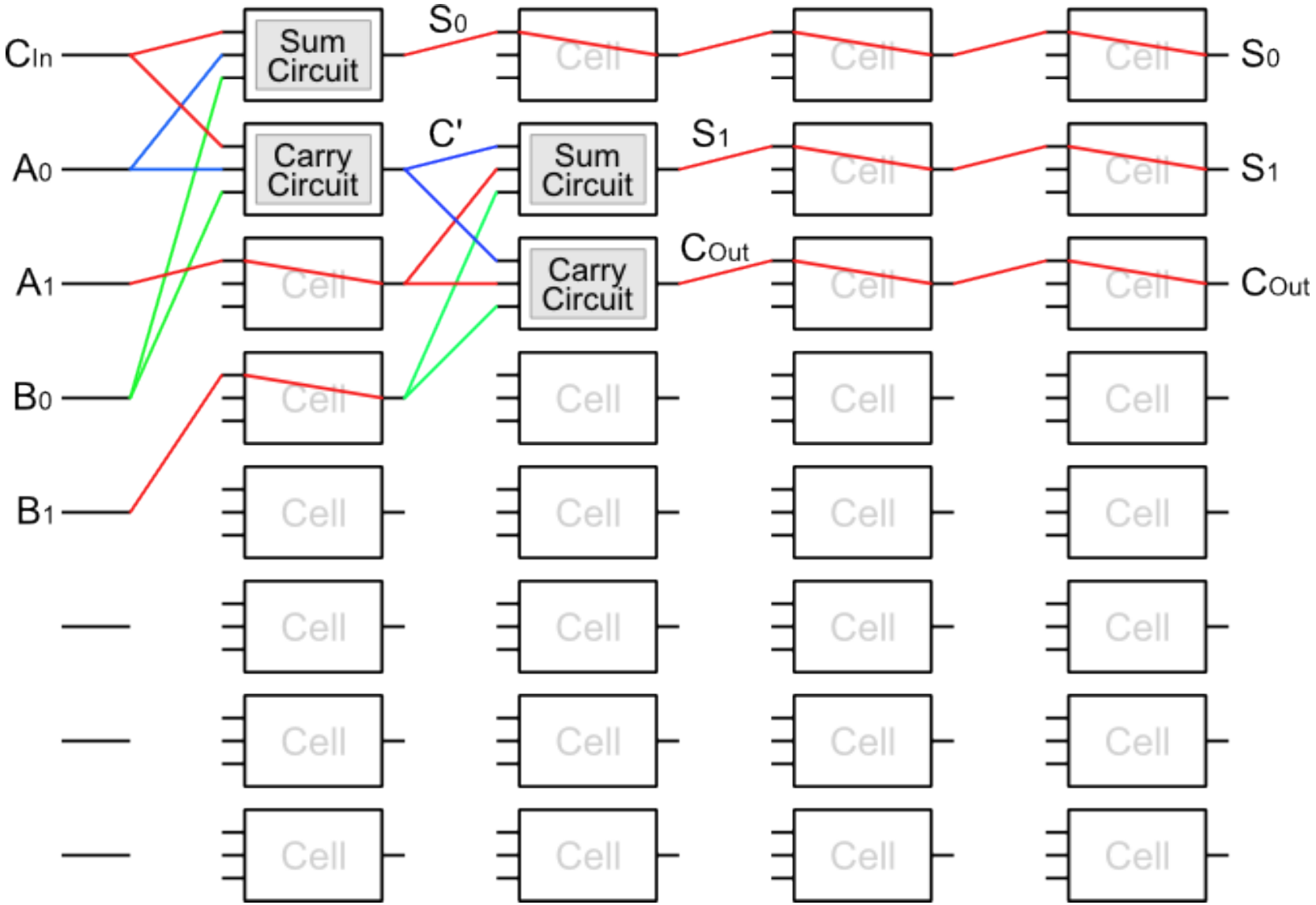
This configurable cell has a 3 input LUT, each LUT input can be connected to one of a selection of 8 cell inputs.

A Configurable Combinatorial Circuit



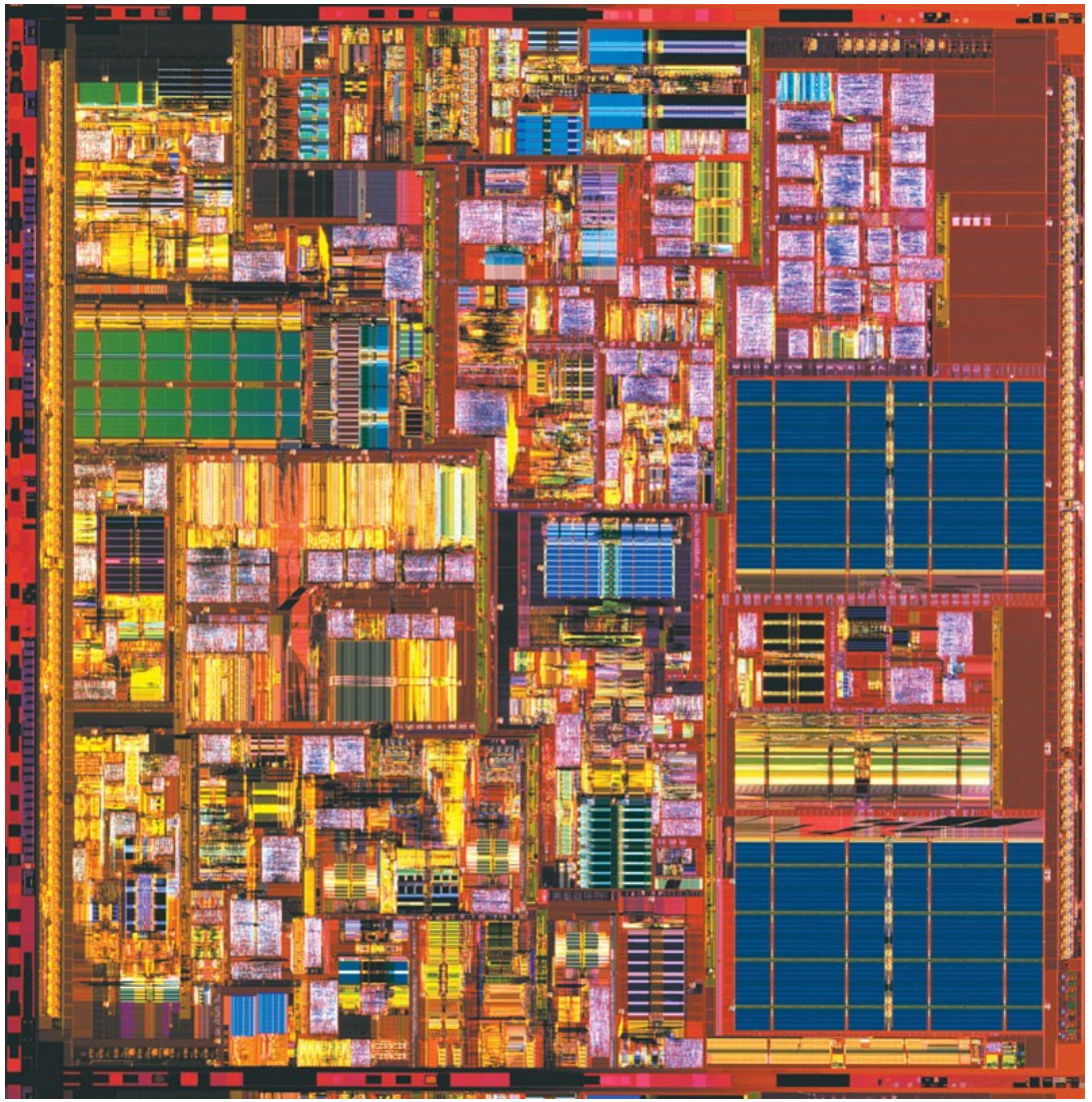
An array of configurable cells, 8 rows, 4 columns (32 Cells)
32 * 17 = 544 configurable bits

Evolution of a 2 bit adder



The 2 bit adder uses just 4 cells for the adding circuit, 9 others being used for routing.

Is EHW up to the Task?



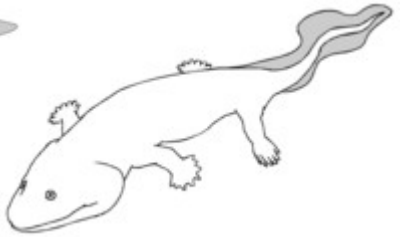
Pentium 4 die (about 50 million transistors)

However, how easy would it be to evolve a circuit as complex as this?

Could evolution take care of:

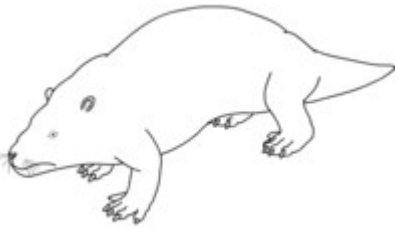
- Connectivity
- Component choice
- Power Consumption
- Clock Distribution
- Fan out/in
- Power Distribution
- Cross talk
- Fault Tolerance

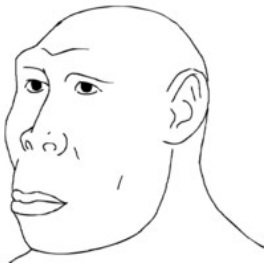
The Difficulty of Evolving Large Circuits

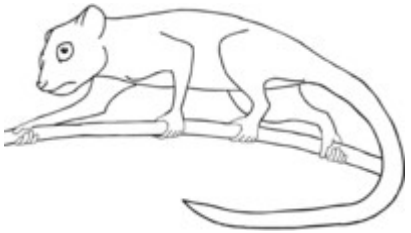


2 Bit Adder?

- Evolution of complex things takes a long time, with a lot of intermediate stages.
- The goal of evolution wasn't to create a human.
- You can't go from start to finish in one step.







Pentium 4?