A Memory Design in QCAs using the SQUARES Formalism.

D Berzon, T J Fountain
Dept. of Physics and Astronomy, University College London, UK
d.berzon@ucl.ac.uk, t.fountain@ucl.ac.uk

Abstract

We present a formalism for implementing circuits with Quantum-dot Cellular Automata (QCA), comprising a set of standard circuit elements with uniform layout rules. The formalism simplifies circuit design from an engineering perspective and overcomes an observed sensitivity of QCA systems to input delays. A design for an addressable shift register is implemented, and promises considerable density gains over conventional CMOS.

1. Introduction

The Quantum-dot Cellular Automaton (QCA) [1] has been proposed as an alternative architecture to CMOS and in principle should permit the implementation of lower-power logic circuitry at higher packing densities. It uses a new paradigm for encoding binary logic into electronic circuitry, where the binary 1s and 0s are mapped to spatial configurations of electrons rather than the magnitudes of electronic currents.

The interaction rules for QCA cells are sensitively dependent on geometric layout. In this paper we present the SQUARES formalism, which is an initial attempt at standardizing the process of QCA circuit design and simplifying the engineering layout rules.

The first section describes the details of the QCA architecture. We then give a description of the basic components of the SQUARES formalism with detailed circuit layouts. The process of circuit implementation is then demonstrated using an addressable shift register as an example. In the final section we evaluate the architecture in terms of logic density and timing considerations.

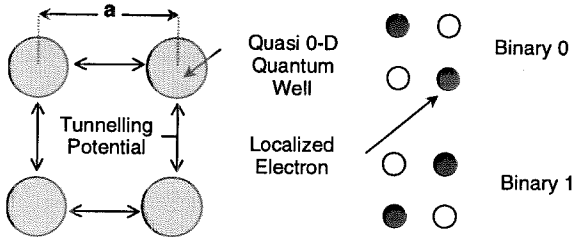

Figure 1. The layout of a QCA cell.

1.1. The QCA architecture

The detailed theory of the QCA architecture and some proposals for manufacture are set out in [2]. The ideal QCA cell comprises four quasi zero-dimensional quantum dots arranged on the corners of a square of side **a**. There is a finite tunneling potential between any two dots, which can be gated to either inhibit or encourage electronic localization. The cell is occupied by two electrons, which when localized will take up diagonally opposite dots. The two possible values of the electronic configuration, which are termed polarisations, can be mapped to binary data states of 1 and 0.

The polarization of a cell can be influenced by either an off-axis external potential, or the coulombic influence of neighboring cells. Cells that are adjacent will match polarisations, and diagonally aligned cells will take on opposite values (anti-voting).

For any arrangement of cells there is a unique ground state which can be mapped to the computational output of a circuit. Simple geometric arrangements of cells can produce basic circuit elements and a full logic family has been developed and simulated [3]. Computation can be achieved in a network by driving the polarizations of input cells, allowing the system to settle to ground state, and subsequently sensing the output polarizations.

1.2. Adiabatic four phase clocking

To guarantee a network's evolution to the ground state, and avoid thermodynamically excited states, it is desirable to implement adiabatic four phase clocking [2]. The circuit is split into phase regions, each with independent control of the tunneling potentials of the QCAs within it. These potentials are gated with a trapezoid waveform, where a low value represents localized electrons and a high value corresponds to de-localization (Figure 2).

There are four transitions corresponding to four computational states: *computing*, where electrons are localizing and influenced by the surrounding potential landscape; *locked*, where the current polarization is held;

0-7695-0104-4/99 $10.00 © 1999 IEEE

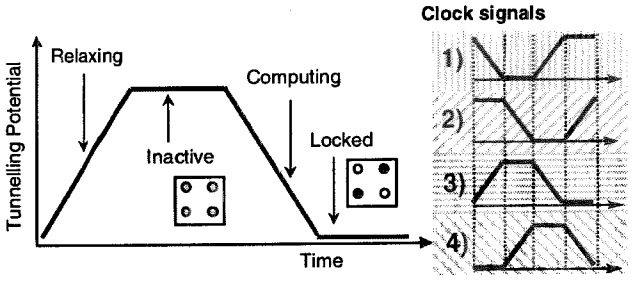

Figure 2. The adiabatic clock signals

relaxing where the electrons are allowed to de-localize; *inactive,* where the electrons are fully de-localized and the cells have no polarizing influence on neighboring regions. If the transitions are slow compared to the characteristic switching time of the cells, the system will remain in a constant ground state.

The waveform is successively phase shifted by $\pi/2$ to create four clock signals (Figure 2). These are distributed to the different phase regions, forcing a data flow in the direction of successive phase. When a region is in the computing phase, the previous region is locked and acting as input with the following region inactive, preventing back processing. In the figure, and in the designs that follow, successive clock signals are shaded with lines rotated clockwise through 45°.

2. The SQUARES formalism

It is desirable for a circuit engineer to have a series of components that can be connected arbitrarily and considered to work independently of the geometric environment. In this paper we present an initial attempt to create such a standard set of QCA components. We have named this formalism Standard QUantum cellular automata Array Elements (SQUARES).

Figure 3 shows the SQUARES' layout. Each tile performs either a single logic operation or connection. The tiles can be tessellated to form any desired binary logic function. Each SQUARE is held inside an area envelope with space for 5×5 QCA cells, with the inputs and outputs on the center of each edge. The cells themselves are of uniform dimension and separation.

The top SQUARES provide the basic connection mechanisms and are adaptations of the designs laid out in [3]. The crossover SQUARE represents the most compact design we were able to develop and necessitated the 5×5 cell envelope for the tiles. The crossover is carried out by the four rotated QCA cells.

The inverter can be combined with corners or fan-outs by the addition of input or output cells with the restriction that only one output can be inverted. A corner NOT is shown as an example.

The majority gate provides the basic combinational logic of the QCA architecture. The SQUARES implementation can also be used as a three way fan-out by inverting the input and output wires. The AND gate is

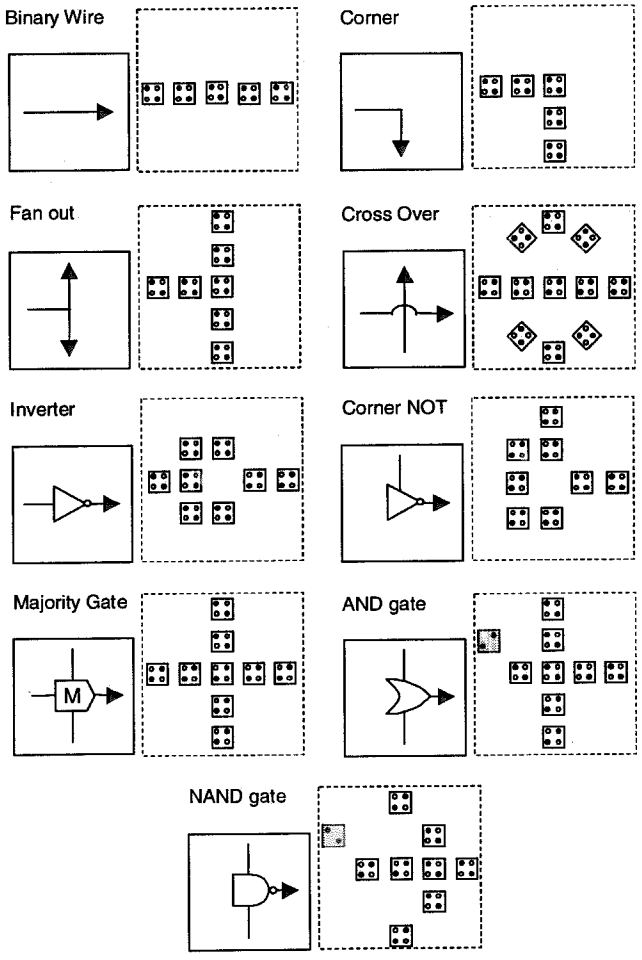

Figure 3. The layout of the SQUARES

implemented by driving the horizontal input to a majority gate with a 0 (although the actual driver cell is a 1) . It was necessary to initially invert the driver signal to maintain the SQUARE's independent operation. Any of the three non-driven wires can be used as the output. The NAND gate is implemented by inverting the inputs of an OR gate. However this design has no flexibility in choice of output wire. OR and NOR gates would use the same designs but with opposite drivers.

The SQUARES have been simulated at a quantum mechanical level in AQUINAS [4], both in isolation and combination with each other, and the correct and independent operation of each has been confirmed. The simulation results were independent of **a**, but the center to center separation of the cells was set at **4a**.

2.1. Clock phase considerations

The cells in each SQUARE are under the same clock phase and when the SQUARES are combined into circuitry the allocation of the clock signals determines the input and output directions. In this way, information is effectively micro-pipelined throughout the system.

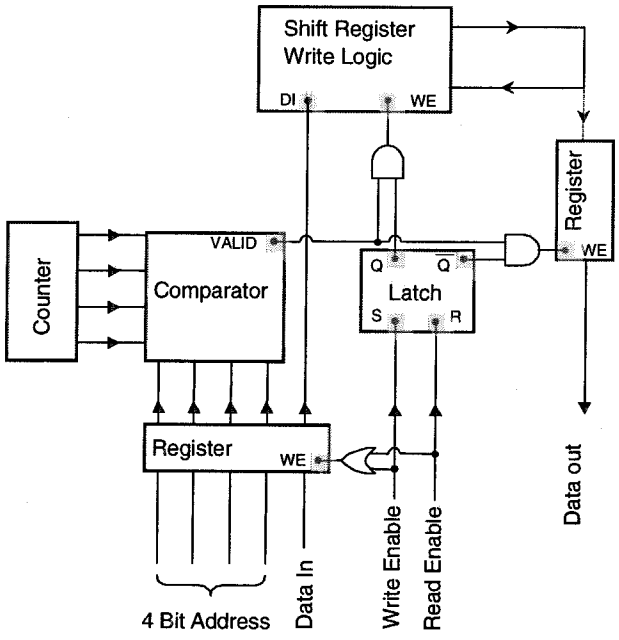

Figure 4. Circuit diagram for the shift register

Early circuit simulations in AQUINAS uncovered a dependence of the operation of majority gates on the length of input wires [6]. This behavior has serious consequences for the size of phase regions, in that input lengths to logic gates have to be equalized and there can only be one gate per phase region.

The SQUARES architecture was developed partly in response to these considerations, so that all input lengths are effectively equal. As long as each gate is assigned a different clock phase, the SQUARES have been shown to be resistant to this source of error. Although it is only necessary to separately clock the gate SQUARES and not the interconnects, timing considerations make it more efficient to do both. The time needed for a clock transition is determined by the intrinsic switching time of a QCA and the number of cells in the phase region. Therefore, the maximum connection length under one phase region determines the transition time and it becomes more efficient to clock each SQUARE separately. However this will increase circuit size as discussed below.

There is currently major doubt as to whether the input delay dependence of gates is a real phenomenon, as it appears to contradict the adiabatic theorem [5]. The matter remains unsettled, although current work at Notre Dame suggests that the behavior is a consequence of the AQUINAS simulation model. In this paper we have assumed the dependence of circuits on input delays and the designs that follow use full micro-pipelining. If the behavior proves to be false, the modularity benefits of the SQUARES are not lost but clock phases could be shared between SQUARES.

3. Implementation of memory circuitry

We decided to focus on memory implementation as an initial application of the SQUARES architecture. The first device implemented was an addressable S-RAM cell [6]. The design occupied 8×12 SQUARES for one bit of RAM, representing a device area of approximately 40,000$\mathbf{a}^2$. For molecular implementations of QCAs this signifies a vast improvement over CMOS but for solid state implementations it is likely that this is within the bounds of the SIA roadmap.

To increase the density of QCA memory, we investigated alternative architectures, more suited to QCAs. Here we present a memory system based on addressable shift registers. CMOS shift registers require logic to implement bit shifting between elements of the register, but clocked QCA circuits have an inherent micro-pipelining, with an extended binary wire carrying one bit of information every four SQUARES. A shift loop can be implemented easily as a closed loop of binary wire SQUARES.

Figure 4 shows the circuit. A 16 bit shift loop is accessed with a four bit address which is compared with a 4 bit counter. When writing, this allows the "data in" value to be written to the correct element. The output of the loop is continually piped to a register, and a read signal will load this register, writing the required bit to the output line. The registers on the address and data line, as well as the S-R latch on the WE/RE lines are necessary to hold the input state while the 4 bit counter cycles. The capacity of the design can be extended by increasing the register length and number of address bits. In addition, n-bit words can be implemented using parallel shift loops.

In Figure 5 we show a SQUARES implementation of the basic elements for the addressing circuitry. The write logic uses a majority gate to write to the loop in one clock phase, in a manner similar to an S-RAM. The 1 bit register uses a simple S-RAM logic and the S-R latch is a translation of a CMOS latch, with different handling of race conditions.

The comparator can be implemented as a string of XNOR gates in series, each comparing one address bit with one counter bit. The outputs can then be combined with an AND gate to produce the final valid signal. Figure 5 shows an implementation of an XNOR using a simple Boolean translation.

The counter mechanism uses an alternating signal generator and a series of frequency dividers to produce a sequential binary count form 0 to 15. The implementation of the signal source uses an inverter loop with an AND gate to reset. The frequency divider proved less trivial, as the normal D-type latch implementation will not work with pipelined pulses. In our circuit the left-hand logic block first selects a clock edge. This then sends a pulse to the right-hand logic block, which inverts the output signal using an XOR gate with the previous

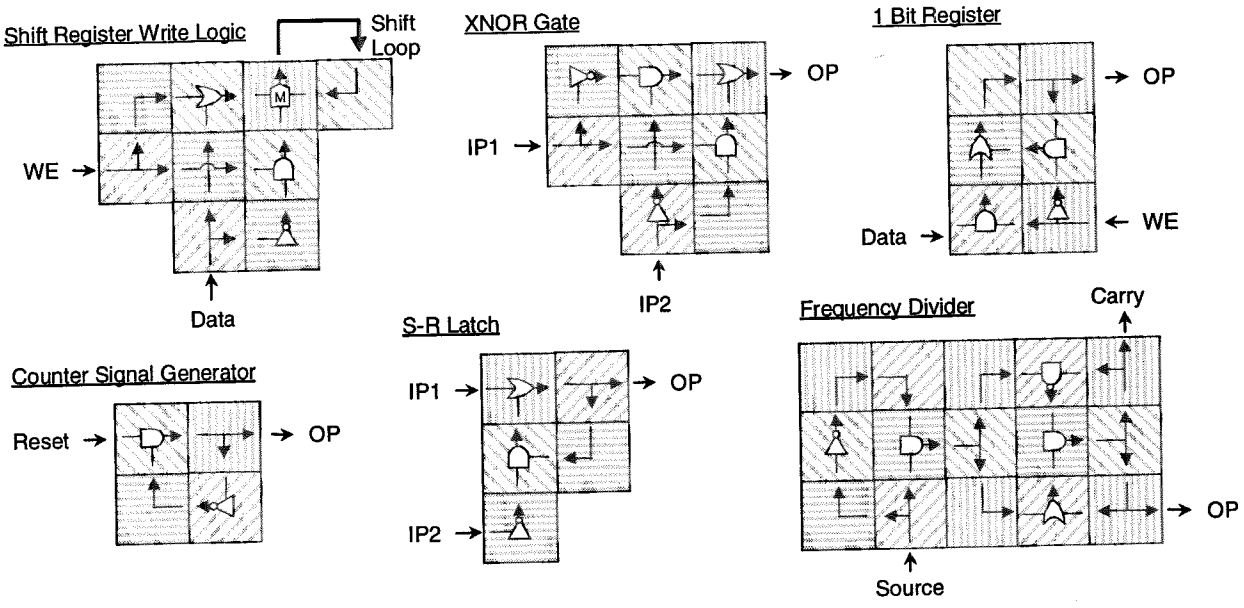

Figure 5. SQUARES implementation of the elements of the addressable shift register.

output as an input. The overall effect is to produce an alternating output with half the frequency of the input.

These components have been combined to produce a working shift register circuit. The full layout, too complex to be included here but available on the World Wide Web [7], is contained within 24×28 SQUARES for 16 bits, and represents a ×2 increase in logic density over the S-RAM design. The majority of this area is required for the counter/address comparison and input register circuitry. The extension of the architecture to parallel shift loops offers further logic density gains, as the addressing circuitry will remain constant in size.

The propagation of information in the individual components has been simulated using a spreadsheet and at the time of going to press we are simulating the full circuit. Detailed information concerning access times and synchrony considerations remain to be collected.

4. Evaluation

The SQUARES formalism has major advantages from an engineering perspective in that it both standardizes the interconnection rules between circuit elements and allows them to be treated as black boxes. The major disadvantages over low-level designs come in the extra cost both in terms of access times and spatial redundancy.

Exact values for clock transition lengths – and hence access times – are determined by the physical parameters of QCAs. The sub gate level pipelining required by our designs is unprecedented in its complexity and GHz clock frequencies will be needed to make the systems worthwhile.

The SQUARES are spatially redundant at two levels. There is unused space inside the SQUARES themselves, which could be avoided if the modularity was at a higher level. However this would require a more flexible connection system and may not be possible if input delay dependence is an issue. At a higher level space is wasted due to the micro pipelining of the interconnections and the necessity of ensuring phase regions connect sequentially. Again this situation could be alleviated if the input delay issue is resolved.

5. References

[1] C.S. Lent, P.D. Tougaw, W Porod and G.H. Bernstein "Quantum Cellular Automata", Nanotechnology, 4–49, 1993.

[2] C.S. Lent and P.D. Tougaw, "A device architecture for computing with quantum dots", Proceedings of the IEEE, 85–541, 1997.

[3] P.D. Tougaw and C.S. Lent, "Logical Devices Implemented Using Quantum Cellular Automata", Journal of Applied Physics, 75–1818, 1994.

[4] A QUantum Interconnected Network Array Simulator (AQUINAS) © 1996 – 1997, www.nd.edu/~qcahome.

[5] D.J. Griffiths, "Introduction to Quantum Mechanics", Prentice Hall, 1994, Chap 10.

[6] D. Berzon and T.J Fountain, "Computer Memory Structures using QCAs", IPG Rept No 98/1, http://ipga.phys.ucl.ac.uk/

[7] http://ipga.phys.ucl.ac.uk/research/arrays/qca.html