Microelectronics Journal 55 (2016) 152–161

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Quantum-dot cellular automata serial decimal processing-in-wire: Run-time reconfigurable wiring approach

Michael Gladshtein

ORT Braude College, 51 Snunit st., Karmiel 2161002, Israel

ARTICLE INFO

Article history:
Received 7 April 2015
Received in revised form
12 April 2016
Accepted 20 July 2016

Keywords:
Quantum-dot cellular automata
Reconfigurable computing
Decimal adder/subtractor
Johnson–Mobius code

ABSTRACT

The quantum-dot cellular automata (QCA) technology is promising to overcome the limits of CMOS technology for perspective computers. A chain of QCA is used as a wire. Logic gates are simply implemented by a cross pattern of QCA. Because the leading role of QCA wires, the serial data transfer/processing is preferable. The growing market of financial, Internet-based, and automatic control computer applications requires a binary-coded decimal data encoding for direct processing of decimal information without representation and conversion errors. The 5-bit decimal Johnson–Mobius encoding and radical departure from Boolean logic allow using a delay element, implemented by short length of QCA wire, as a function element. The previous published by author serial decimal QCA arithmetic designs demonstrate hardware simplification in comparison with traditional designs. The paper presents novel serial decimal adder and adder/subtractor designs used the run-time reconfigurable wiring approach, which results in further significant QCA hardware simplification.

1. Introduction

The CMOS is now the main technology for implementation of microelectronic computers. All microelectronic computers use binary notation and bit-parallel data transfer/processing. These principles are optimal for microelectronic computers: the binary number system, as it requires a smaller number of expensive elements—transistors—and allows implementation of arithmetic operations by logic gates; parallel data transfer/processing increases the binary processor performance (but increases number of wires and poses the binary carry propagation problem). Today the CMOS technology is approaching to its physical limit according to the scientific prediction [1].

The next stage in computer production is a nanoelectronic technology. The possible nanoelectronic devices are summarized in [2]. One of perspective nanoelectronic technologies is the quantum-dot cellular automata (QCA) based technology. The notion of QCA was introduced in 1993 [3]. The possible physical implementations (metal tunnel junction QCA, molecular QCA, and magnetic QCA) and experiments are reviewed in [4]. Despite the several limitations (low working temperature for the metal tunnel junction QCA, low switching speed for the magnetic QCA, etc.) the QCA technology is considered as a future computer technology [5]. The clocked molecular QCA provide high density, high switching speed, and ultralow power dissipation [6,7]. The design and fabrication aspects of clocked molecular QCA are discussed in [8].

A paradigm for computing with QCA is described in [9]. QCA operates by the Coulombic interaction between neighbor cells. Synchronization is provided by clocking in four phases. The signal propagation through a QCA wire, built by a chain of cells, is similar to the signal propagation through a conventional *shift register*. Logic primitives are simply implemented with a cross pattern of cells. The complete set of logic primitives includes an inverter and three-input majority voter. Two-input logic AND and OR gates are implemented from the majority voter by setting the third input permanently to a “0” or “1” value.

The computational power of QCA memory circuits is based on the *memory-in-motion* technique [10], and arithmetic circuits, on *processing-in-wire* technique [11]. The report [12] summarizes state-of-art of QCA arithmetic units design. The projects of arithmetic-logic unit [13] and simple 4-bit processor [14] have been also published. However, all these designs use the fundamental information principles inherited from microelectronic computers: binary notation and parallel data transfer/processing. Because these arithmetic designs are based on binary arithmetic, only *logic-in-wire* [15] technique is used as an instance of processing-in-wire. The price ratio between switching elements and wires reverses in QCA nanotechnology. Wires become the main elements: they contain more cells and have the longer propagation delay [16,17].

The QCA combinational and sequential circuits design problems (partitioning, placement, routing, floor-planning, etc.) are discussed in technical literature [18,19]. Several CAD tools are available for design and simulation of QCA circuits: QCADesigner [20], QCAPro [21], QCA-LG [22], etc. The system-level simulation-based researches drive the QCA nanotechnology forward [8], despite the

http://dx.doi.org/10.1016/j.mejo.2016.07.009
0026-2692/

fabrication of complex QCA circuits is impossible today.

Besides, the application area of computers extends. Computers process large volumes of decimal information in financial, commercial, Internet-based, and automatic control applications, which cannot tolerate errors from converting between decimal and binary formats [23]. Because of the growing importance of decimal floating-point (DFP) arithmetic, specifications for it have been added to the IEEE 754-2008 Standard for Floating-Point Arithmetic [24]. Moreover, IBM added DFP instructions to the z9 and z10 microprocessors [25,26].

Presently, decimal arithmetic operations for fixed-point Binary Coded Decimal (BCD) operands are supported by any binary general purpose processor using decimal adjust instructions. This approach is ineffective: a binary instruction and adjust instruction must be executed for any decimal operation. Several QCA parallel BCD adder designs have been published [27–29]; but they are less effective in comparison with binary adders. More effective BCD parallel CFA (carry flow adder) and CLA (carry lookahead adder) are proposed in [30] but they also based on binary addition and adjust principle. The biquinary coded decimal adder is published in [31] but it also uses additional adjust operation. It follows that further improvements are required in decimal coding, algorithm concepts, and hardware design [32].

2. Author's previous research and further direction

2.1. First approach: direct serial decimal processing

As was shown by the author in [33], alternative principles should be chosen for a nanocomputer implementation: *serial data transfer and processing* (to decrease total length of wires) and *direct decimal processing* (for elimination of decimal-binary and binary-decimal conversions with their inevitable errors). Because the QCA are inherently two-state, the last choice requires a binary-coded decimal encoding, which meets the QCA properties. The main criterion of an optimal encoding is arithmetic processing support by *shift* operation. The additional criterions are arithmetic hardware simplification and error detection.

A biquinary redundancy encoding for decimal digits allows arithmetic hardware simplification by separate processing of a binary and quinary components of the code as well as error detection. The quinary component represented by the unitary or unary code supports arithmetic processing by shift operation.

As was shown by the author [34], the 5-bit Johnson–Mobius decimal code (JMC) presented in Table 1 is also a biquinary code. The most significant bit (MSB) of the JMC represents its binary component, and four lower bits represent its quinary component. The last component is encoded in the direct (if MSB=0) or inverted (if MSB=1) 4-bit fixed-length unary code. Among other possible decimal biquinary codes the JMC has the significant advantages: it includes only one additional bit in comparison with the minimal BCD code and supports the simplest arithmetic processing (counting) by *twisted-ring rotation* (the shift operation in a Johnson counter). Moreover, inverting all of the bits in the decimal JMC is equivalent to addition/subtraction 5. It follows that the decimal JMC allows separate processing of its binary component (by inverting all of the bits) and quinary component (by twisted-ring rotating the code). Despite minor redundancy it presupposes the error detection capability.

The QCA serial decimal JMC adder design [35] proved a possibility of the decimal JMC addition implementation by using INVERT (+5) and LEFT TWISTED-RING ROTATE (+1) operations. Two variants of the serial decimal JMC subtractor, based on the described above serial decimal adder, were designed on QCA [36]. The paper [37] reports on two variants of the serial decimal JMC adder/subtractor designs and serial JMC change-of-level error detector.

Table 1
Decimal Johnson–Mobius encoding.

Digit	Johnson–Mobius code
0	0 0000
1	0 0001
2	0 0011
3	0 0111
4	0 1111
5	1 1111
6	1 1110
7	1 1100
8	1 1000
9	1 0000

2.2. Second approach: delay-based processing-in-wire

In the *delay-based* nanocomputing approach suggested in [38,39], computations are performed on values that are represented by time measured as the delay between two events. Such approach incorporates both switching delay and interconnection delay into the computation, no longer requiring the specialized techniques to minimize the delays. The main operation —addition—is performed by propagating the signal through programmable delay element implemented by a clocked QCA wire. The delay can be programmed by choice the position where the output is taken. The possible wire length limits the range of representable values. As it is concluded in the thesis [39], the delay-based QCA computing devices are currently too large to compete with their conventional counterparts and need to be optimized for complexity and area.

As shown by the author [40], several serial binary-coded decimal codes meet the delay-based computing requirements (10-bit one-hot code, 9-bit unary code, biquinary codes used one-hot or unary encoding for a quinary component). The 5-bit-serial decimal JMC, by virtue of its unique properties, is the best one for implementation of the QCA *decimal delay-based processing-in-wire*. The main idea of delay-based JMC processing-in-wire implies that if the serial JMC of decimal digit and its inverted copy are propagating through a QCA wire, a 1 CLK delay (propagating through four clock zones) performs a 1-bit left twisted-ring rotation of JMC (increment operation).

The main building block for delay-based processing-in-wire—a serial left barrel twisted-ring rotator (SLBTR)—can be built by a Johnson counter and 5-to-1 multiplexer controlled by the parallel shift amount one-hot code *D*. The Johnson counter can be simply implemented by a loop of five 1 CLK delay elements (short patterns of QCA wire), closed by an inverter. The needed delay is implemented by the 5-to-1 multiplexer, which selects the appropriative point in the loop of delay elements in accordance with the shift amount one-hot code *D*. The limited set of processing-in-wire serial building blocks (bit flipper, left twisting-ring rotator, left barrel twisting-ring rotator, 9's complementer, and multiplier by 5) makes it possible performing all of four arithmetic operations [40].

2.3. Next direction: run-time reconfigurable wiring approach

A disadvantage of the described above main delay-based processing-in-wire building block—SLBTR—is encoding the shift amount code *D* by the parallel 5-bit one-hot code. This fact requires complex processing of the second operand *B* serial JMC: serial to parallel code conversion and conversion of the direct/