Radix-4 Recoded Multiplier on Quantum-Dot Cellular Automata

Ismo Hänninen and Jarmo Takala

Tampere University of Technology,
Department of Computer Systems
PO BOX 553, FI-33101 Tampere, Finland
{ismo.hanninen,jarmo.takala}@tut.fi
http://www.tkt.cs.tut.fi/index-english.html

Abstract. This paper describes the implementation of an advanced multiplication algorithm on quantum-dot cellular automata (QCA) nanotechnology, promising molecular density circuits with extreme operating frequencies, using a single homogeneous layer of the basic cells. The multiplier layout is verified with time-dependent quantum mechanical simulation, to ensure stable ground state computation under the fine-grained pipelining constraints of the technology. The novel design utilizes radix-4 modified Booth recoding and ultra-fast carry-save addition, resulting in stall-free pipeline operation, with twice the throughput of the previous sequential structure and minimized active circuit area.

Keywords: QCA, nanotechnology, arithmetic, multiplication.

1 Introduction

Quantum-dot cellular automata (QCA) is a promising nanotechnology, which offers ways to reach molecular circuit densities and clock frequencies surpassing traditional digital technologies by several orders of magnitude, possibly reaching the teraherz regime. The concept was introduced in early 1990s [1,2] and has been demonstrated in laboratory environment with small systems [3, 4, 5]. The revolutionary operating principle promises outstanding energy-efficiency and computing performance, which has evoked considerable interest in the digital design community, although the implementation technologies are still in development.

The primitive device of QCA technology is a bistable cellular automaton, which is operated under clocked control. The physical cells and clocking can be created in several ways, and the approaches promising the fastest switching and highest performance are based on electrostatic coupling between the automata, manufactured with semiconductor, metal island, or molecular process [3, 4, 5].

Information is encoded into the local position of excess charged particles in the QCA cell, having two opposite polarization states to represent binary zero and one, as shown in Figs. 1(a) and 1(b). The state of a cell can be copied into the neighboring cell, enabling a simple wire construct, and a coplanar wire crossing shown in Fig. 1(c). A universal combinatorial logic set is usually constructed with

K. Bertels et al. (Eds.): SAMOS 2009, LNCS 5657, pp. 118–127, 2009.

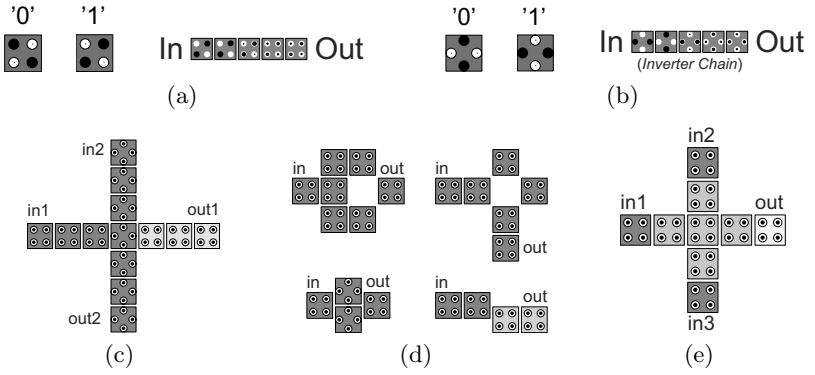


Fig. 1. QCA primitives: a) type 1 cell and wire, b) type 2 cell and wire, c) coplanar wire crossing, d) inverters, and e) three-input majority gate

an inverter gate and a three-input majority gate, shown in Figs. 1(d) and 1(e). The majority voter can be configured as a two-input AND gate by fixing the third input to zero value, or an OR gate, with the third input fixed to one. [1,2]

A clocking field determines when the cells are reset to an un-polarized state, latch their input values, and start driving neighboring cells. This naturally enables sequential logic, but on the nanotechnology, also ensures that the array of cells reaches a stable ground state and has true signal gain. Typically, the clock is applied by interleaving four clocking zones repeatedly across the design, and creating an ultra-fine-grained pipeline. [2,6,7,8]

This paper presents the implementation of radix-4 modified Booth multiplication on QCA, describing the resulting systolic layout. Noise coupling due to imperfect cellular interaction is avoided by the clock zone placement approach [9], which has a serious consequence: the additional latencies translate into a stalling pipeline, wrecking the performance completely. We avoid this problem by using ultra-fast carry-save addition, and develop components that have matched data rates. The stability of the ground state is verified by time-dependent simulation, using the coherence vector engine of QCADesigner software [10].

The novel multiplier reaches twice the throughput of the previous sequential unit, using only minimal circuit area for active logic, while passive wiring dominates with square-law dependence on the operand word length. The previous designs represent the simple extremities of parallel computation [11,12,13], while the more advanced algorithm leads to many ways to adjust the degree of parallelism, which enables exploring the design space for fault-tolerance.

The rest of this paper is organized as follows: Section 2 summarizes the previous work on QCA arithmetic and Section 3 the applied multiplication algorithm. Section 4 describes the proposed implementation, while Section 5 presents design analysis. The conclusions follow in Section 6.

2 Related Work

There has been a considerable amount of research into arithmetic circuits on QCA, aiming to solve the challenges of more general digital design. The first papers described a full adder unit [1, 2], followed by the optimal majority logic formulation [14]. The clock zone approach to avoid radius-of-effect induced noise coupling was considered in [9], and a dense layout with one-cycle carry latency, required in stall-free multi-bit pipelines, was presented in [15, 16].

Multi-bit addition was proposed using a standard bit-serial adder [11,17] and a ripple carry adder (RCA) [14, 18, 15, 16]. Advanced structures with reduced latency, the carry lookahead adder (CLA) and the conditional sum adder (CSA), were adapted to QCA and analyzed in [19].

The first multiplier proposal for QCA was the serial-parallel structure [11,12], processing one of the operands as a parallel word and the other bit-serially. This was followed by a fully parallel cellular array multiplier, which was shown to be quite competitive structure on QCA, due to the wiring overhead evening out the area difference between the designs [13, 20]. The novel approach presented here is situated between the previous proposals, and offers several directions for further improvements, not available for the simpler structures.

3 Radix-4 Multiplication

The popular radix-4 modified Booth algorithm handles two's complement numbers directly, and enables a systolic structure with no control overhead or feedback signals, which is very important simplification on QCA. The approach, as shown in Fig. 2, scans the multiplier word in groups of three bits, including an overlapping lookbehind reference, and transforms it into a high-radix signed-digit (SD) number, where the digits represent the required *operations*. The number of clock cycles is halved, in comparison to the direct sequential algorithm.

The recoding function shown in Table 1 enables skipping over continuous sequences of zero or one bits in the multiplier word, replacing a sequence of operations with just two additions and shifting, while handling also isolated ones efficiently, as opposed to the standard radix-2 Booth algorithm, which can double the number of operations in the worst case. Since the transformation has no dependence between adjacent digits, all the groups can be recoded in parallel, opening a way to compute also the partial products in parallel. This would not be possible with canonical SD recoding, offering maximum number of zero digits.

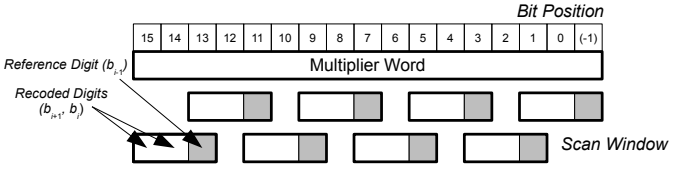


Fig. 2. Radix-4 overlapped scanning, processing two bits and a lookbehind reference

Table 1. Recoding function, converting multiplier bits into the radix-4 operation

Multiplier bits			Comment	Operation
b_{i+1}	b_i	b_{i-1}		
0	0	0	string of zeros	0
0	1	0	single one	$+A$
1	0	0	start of ones	$-2A$
1	1	0	start of ones	$-A$
0	0	1	end of ones	$+A$
0	1	1	end of ones	$+2A$
1	0	1	single zero	$-A$
1	1	1	string of ones	0

4 Implementation

The main design objective was to achieve continuous operation without stalling the pipeline, since this would deteriorate the performance and require complex control, which seems currently very unpractical on QCA due to the timing restrictions. The proposed multiplier is a pipelined systolic structure with no feedback signals, achieving the halved latency and doubled throughput promised by the radix-4 algorithm. However, the design does not benefit from skipping of continuous bit sequences in the multiplier word, which on traditional technologies gives a variable performance boost, depending on the particular multiplier word. The reason for this is the ultra-low latency carry-save adder (CSA), which we are using to accumulate the partial products, making the cost of just shifting relatively high, since the physical distance translates directly into latency and additional pipeline stages, on the nanotechnology. If the adder had lower throughput than the other components, the zero/shift operations could be utilized to skip the adder completely, but with considerable control cost.

The top-level structure is shown in Fig. 3, having eight modules: two blocks for multiplier word recoding (shift register and recoder logic), three blocks for creating and selecting the multiples (complementer, distribution network, and mux), and three blocks for accumulating the partial product and converting it into the final result (carry-save adder, vector merge adder, and shift register).

Table 2. Latency and area of the components

Component	Latency (clock cycles)	Area (cell area units)
Multiplier shift reg.	2	$100n_B + 100$
Multiplier recoder	4	1650
Multiplicand compl.	n_A	$200n_A$
Multiple distribution	$(n_A + 1)/2 + 6$	$140n_A^2 + 500n_A$
Multiple select mux	$(n_A + 1)/2 + 3$	$900n_A + 900$
Carry-save adder	$\lceil (n_A + 1)/2 \rceil + 1$	$360n_A + 700$
Vector merge adder	4	700
Result shift reg.	$\lceil (n_A + n_B)/2 \rceil - 1$	$200\lceil (n_A + n_B)/2 \rceil$

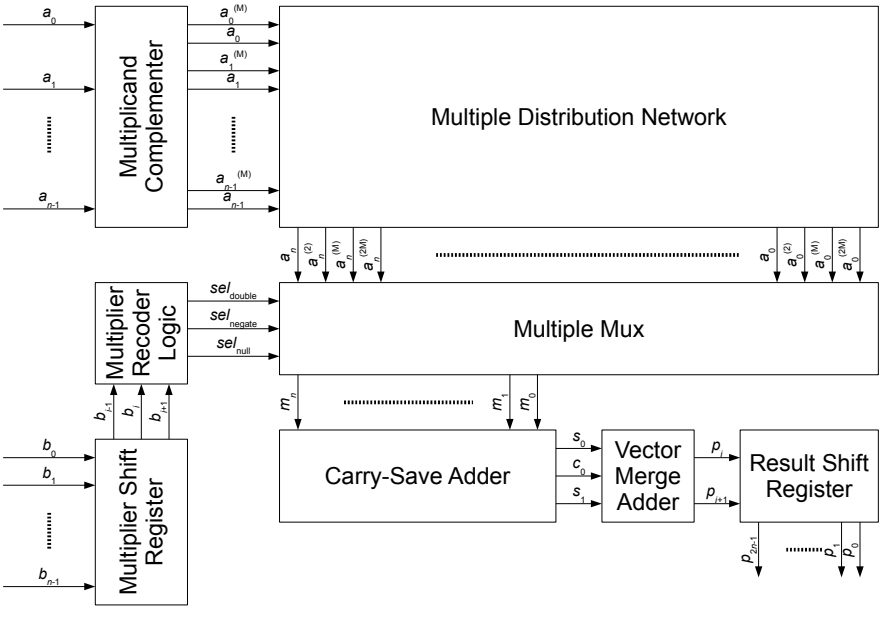


Fig. 3. Radix-4 recoded multiplier on QCA, block diagram

Table 2 summarizes the operand word length dependence of the latency and area of the separate components, but the pipelined bit slices run in parallel, effectively hiding the internal delays. The total latency is linear: $L_{total} = 2n_A + n_B/2 + 16$, where n_A is the multiplicand word length and n_B the multiplier word length.

The handcrafted layouts were verified with the QCADesigner tool (v. 2.0.3), using the coherence vector engine [10]. The fixed size modules and bit slices were simulated exhaustively, covering all the pipeline states, and the combined blocks were tested with a number of cases and word lengths, as exhaustive runs were prevented by the exponential number of states. Correct functionality was obtained with simulated clock frequencies up to one terahertz, using default parameters and cell width 18 nm, corresponding to a semiconductor technology.

4.1 Recoding the Multiplier and Creating the Multiples

The n_B-bit multiplier word B is initially fed into the shift register for conversion from parallel format into serial stream of bit groups, effectively providing a 3-bit sliding window into the multiplier word, shifting two bits per clock cycle (one overlapping reference bit). The recoder block transforms the groups into the radix-4 SD format sequentially ($\lceil n_B/2 \rceil$ digits), the digits internally expressed as the multiple selection control signals sel_{double}, sel_{negate}, sel_{null}.

The n_A-bit multiplicand word A (parallel format) is initially fed into the block forming the negated multiple $(-A)$ (two's complement, with linear latency), while the following network produces the doubled multiples $(2A, -2A)$ by wiring

that implements the 1-bit shift and extension into $n_A + 1$ bit length. The four words are interleaved by the bit position and fed into the mux, which selects one of them or produces the zero-multiple (choosing from $+A$, $+2A$, $-A$, $-2A$, 0), based on the three control signals. This wiring as a whole has a linear latency, and a dominating square-law area, in respect to the operand word length.

4.2 Accumulating the Multiples

The word-sequentially operating carry-save adder, shown in Fig. 4, accumulates the selected partial product and shifts the carry-save formatted sum two digits to the right. This effectively combines three full width binary operands (new multiple, sum vector, and carry vector) and computes the whole $n_A + 1$ bit accumulation in one clock cycle, matching the input data rate and enabling the continuous operation of the pipeline. A ripple-carry adder on QCA would require a linear number of stall cycles, and the best lookahead and conditional sum adders logarithmic, on the operand word length. This would stop the pipeline between each addition, waiting for the availability of the feedback value.

The carry-save adder consists of two-digit wide slices that compute in parallel, to obtain the continuous data flow using the QCA full adders as (3,2)-counters. The full adder has a carry latency of one clock cycle and sum latency of two cycles (the fastest noise rejecting design [15, 16]), which matches the two-bit shifting very well: there is just enough time for the critical path to produce a digit for the next slice, to be used on the next clock cycle. Because of the shifting, a carry bit is moved one digit position to LSB direction, and the sum bit, requiring longer

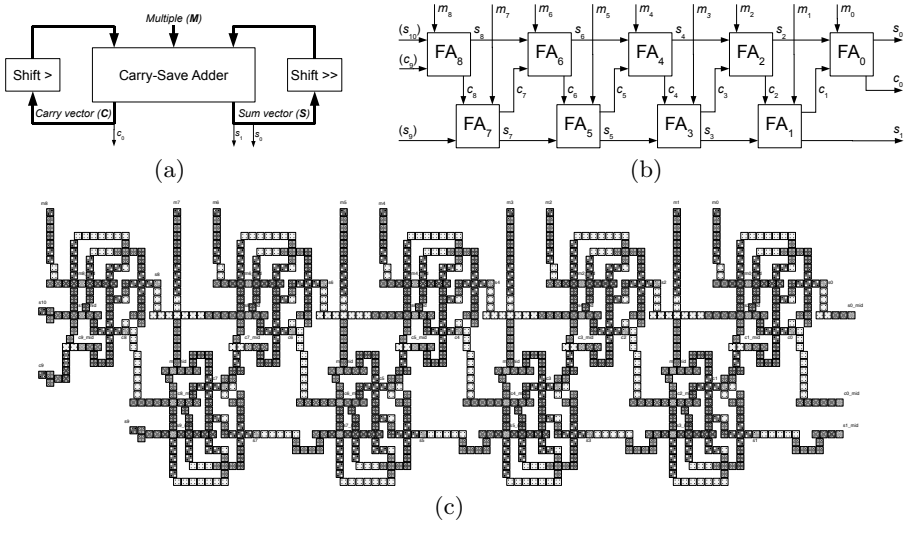


Fig. 4. Word-sequential carry-save adder with shifting: a) conceptual structure, b) logic with full adder (FA) components, and c) QCA layout with 9-bit operand

computation time, two digit positions to LSB direction. An accumulation step is spread in time, each digit slice computing with different operand.

4.3 Merging the Vectors

The accumulated partial product has to be processed into standard non-redundant binary number (radix-2). The vector merge adder, shown in Fig. 5, adds the sequential stream of sum and carry vector bits with proper weights, propagating the carries from the LSB side towards MSB side. On each clock cycle, two bits from the sum vector (s_0, s_1) and one bit from the carry vector (only c_0, since c_{-1} does not exist) are transformed into two bits of the final result (p_i, p_{i+1}), which are interleaved by the shift register to produce the parallel result word P.

The vector merge adder consists logically of a half adder (HA) and a full adder (FA) component, but the existing QCA designs for these blocks could not be utilized, since their combination would have a total latency of two clock cycles, stalling the pipeline. A layout-level optimization of the combination was designed, to match the data rate and solve the carry dependence between two bit positions, since we are summing two radix-4 digits, obtaining bi-directional carry exchange during one clock cycle.

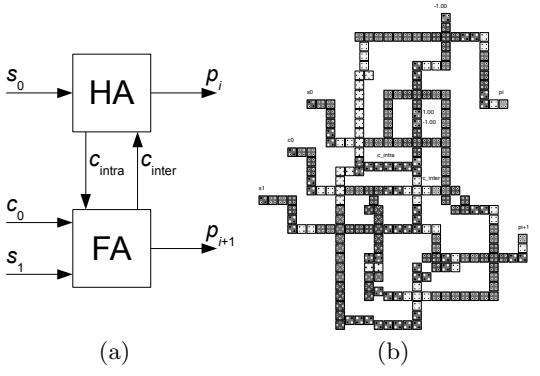
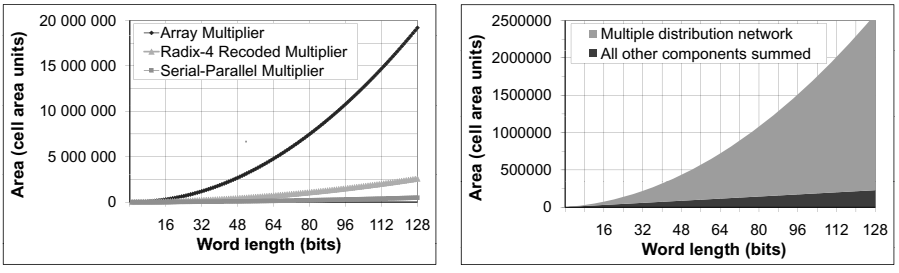


Fig. 5. Vector merge adder: a) logical structure and b) latency-optimized QCA layout

5 Design Analysis

The parallelism of the novel radix-4 recoded multiplier is between the two previous proposals, i.e. the area of the implementation is traded against time. The serial-parallel multiplier [11,13,12] uses time-multiplexed hardware, fully parallel in respect to the multiplicand operand, bit-serial in respect to the multiplier operand, while the array multiplier [13,20] has dedicated adder rows, fully parallel in respect to both operands. The performance and cost metrics are compared in Table 3, as functions of the common operand word length n bits.



(a) (b)

Fig. 6. a) Circuit area of the multipliers and b) Radix-4 multiplier component areas

The latency of the nanotechnology multipliers is in the linear regime, but there is tremendous difference in throughput, which is usually considered the most important performance metric. After the pipelines are filled, the clear winner is the array multiplier, producing a new result on each clock cycle. Our novel radix-4 unit produces a new result once in every n cycles, which is the theoretical maximum when the partial products are accumulated sequentially. The least amount of results is obtained from the previous bit-serial-parallel multiplier.

The circuit area of the units grows with a square-law dependence on the operand length, as shown in Fig. 6(a), due to the dominating wiring overhead [13]. The smallest design is the serial-parallel [11, 13, 12], while the novel radix-4 multiplier is about five times as large. The massive array multiplier [13, 20] reaches 40 times the size of the smallest design. The ratios settle asymptotically.

The contribution of active and passive circuitry has a common trend: The previous designs suffer from the need to synchronize the operands to the pipelined bit slices with extensive delay wiring (square-law), while the active circuitry of the serial-parallel design grows only linearly and the core of the array multiplier with quadruple dependence on the operand word length. The active circuitry of the novel radix-4 unit is very small and also limited to linear growth, while the huge multiple distribution network consumes most of the area, as shown in Fig. 6(b). The compared designs are all area-optimized and will not yield to further layout-level improvements, leaving architectural and algorithmic approaches as the only way to reduce the high wiring overhead.

The previous multiplier proposals are structurally simple, but in the novel radix-4 design, a compromise in the degree of parallelism has a complexity cost.

Table 3. Multiplier comparison (common word length n)

Design	**Latency** (cycles)	**Throughput** (results per c.)	**Area** (cells)
Proposed radix-4	$2.5n + 16$	$1/n$	$140n^2$
Serial-parallel [11, 13], optimized in [12]	$3n + 2$ $2n$	$1/(2n)$	$26n^2$
Array [13, 20]	$4n - 1$	1	$1100n^2$

The overhead (nearly all in the multiple distribution wiring) increases the area over the previous sequential design, but we are achieving the doubled throughput and utilizing the underlying full adders with 100% efficiency, while the previous design can feed the bit slices with new operands only in about 75% of the cycles.

The radix-4 recoded multiplier can be customized to obtain variable degrees of parallel computation and tolerance against malfunctioning low-level hardware, physical defects and faults. Instead of recoding only one digit of the multiplier per cycle, we can obtain several or all of them at once, and use a tree of carry-save adders to sum several multiples in parallel. Sequential operation with several adders offers the option of module level redundancy to increase the reliability, but requires a runtime control mechanism.

6 Conclusions

This paper has described the design of a novel recoded radix-4 multiplier, under the fine-grained pipelining constraints needed to obtain stable ground-state computation on quantum-dot cellular automata. Correct operation of the layout-level implementation has been verified with time-dependent quantum mechanical simulation, up to one terahertz frequency. The unit has twice the throughput (results/cycle) of the previous sequential design, at the cost of complexity and larger area, but provides also a versatile starting point for development.

Our work is one of the first attempts to design advanced computer arithmetic for the very promising QCA technology, which has several challenges hindering large scale manufacturing. The fundamental issue still to be addressed is the presence of defects and faults inherent to the molecular implementations, resulting in the need to find a hierarchical redundancy scheme to enable practical large systems. Another as important an issue is the power dissipation: QCA is predicted to re-use signal energy so efficiently, that the most important heat source will be the irreversible bit erasures, limiting the operating frequency and giving incentive to develop reversible computing approaches.

The challenges need attention on the architectural level. Our aim is to incorporate fault-tolerance into the multiplier design by using redundant hardware in the adder portion, and evaluate the gains and costs of this both on the data path and the required control subsystem. In the long run, practical reversible computing might be obtained, since there is preliminary evidence, that multiplication hardware might be able to maintain information about the system state trajectory, with relatively low cost. [13]

References

1. Lent, C., Tougaw, P., Porod, W.: Quantum cellular automata: the physics of computing with arrays of quantum dot molecules. In: Proc. Workshop Phys. Comp., Dallas, TX, November 17-20, pp. 5–13 (1994)
2. Lent, C., Tougaw, P.: A device architecture for computing with quantum dots. Proc. IEEE 85(4), 541–557 (1997)

3. Snider, G., Orlov, A., Amlani, I., Bernstein, G., Lent, C., Merz, J., Porod, W.: Quantum-dot cellular automata. In: Dig. Papers of Microprocesses and Nanotechnology Conf., Yokohama, Japan, July 6-8, pp. 90–91 (1999)
4. Orlov, A., Kummamuru, R., Ramasubramaniam, R., Lent, C., Bernstein, G., Snider, G.: Clocked quantum-dot cellular automata devices: experimental studies. In: Proc. IEEE Conf. Nanotechnology, Maui, HI, October 28-30, pp. 425–430 (2001)
5. Kummamuru, R., Orlov, A., Ramasubramaniam, R., Lent, C., Bernstein, G., Snider, G.: Operation of a quantum-dot cellular automata (QCA) shift register and analysis of errors. IEEE Trans. Electron Devices 50(9), 1906–1913 (2003)
6. Blair, E., Lent, C.: Quantum-dot cellular automata: an architecture for molecular computing. In: Proc. Int. Conf. Simulation of Semiconductor Processes and Devices, Boston, MA, September 3–5, pp. 14–18 (2003)
7. Lent, C., Liu, M., Lu, Y.: Bennett clocking of quantum-dot cellular automata and the limits to binary logic scaling. Nanotechnology 17(16), 4240–4251 (2006)
8. Frost-Murphy, S., DeBenedictis, E., Kogge, P.: General floorplan for reversible quantum-dot cellular automata. In: Proc. ACM Int. Conf. Computing Frontiers, Ischia, Italy, May 7-9, pp. 77–81 (2007)
9. Kim, K., Wu, K., Karri, R.: The robust QCA adder designs using composable QCA building blocks. IEEE Trans. Computer-Aided Design 26(1), 176–183 (2007)
10. Walus, K., Jullien, G.: Design tools for an emerging SoC technology: quantum-dot cellular automata. Proc. IEEE 94(6), 1225–1244 (2006)
11. Walus, K., Jullien, G., Dimitrow, V.: Computer arithmetic structures for quantum cellular automata. In: Conf. Rec. 37th Asilomar Conf. Signals, Systems and Computers, Pacific Grove, CA, November 9-12, pp. 1435–1439 (2003)
12. Cho, H., Swartzlander, E.: Serial parallel multiplier design in quantum-dot cellular automata. In: Proc. IEEE Symp. Computer Arithmetic, Montepellier, France, June 25-27, pp. 7–15 (2007)
13. Hänninen, I., Takala, J.: Binary multipliers on quantum-dot cellular automata. Facta Universitatis 20(3), 541–560 (2007)
14. Wang, W., Walus, K., Jullien, G.: Quantum-dot cellular automata adders. In: Proc. IEEE Conf. Nanotechnology, San Francisco, CA, August 11-14, pp. 461–464 (2003)
15. Hänninen, I., Takala, J.: Binary adders on quantum-dot cellular automata. J. Sign. Process. Syst. (to appear), http://dx.doi.org/10.1007/s11265-008-0284-5
16. Hänninen, I., Takala, J.: Robust adders based on quantum-dot cellular automata. In: Proc. IEEE Int. Conf. Application-Specific Systems, Architectures and Processors, Montréal, QC, Canada, July 8-11, pp. 391–396 (2007)
17. Fijany, A., Toomarian, N., Modarress, K., Spotnitz, M.: Bit-serial adder based on quantum dots. Tech. Rep. NPO-20869, NASA's Jet Propulsion Laboratory, Pasadena, CA (2003)
18. Zhang, R., Walus, K., Wang, W., Jullien, G.: Performance comparison of quantum-dot cellular automata adders. In: IEEE Int. Symp. Circ. and Syst., Kobe, Japan, May 23-26, pp. 2522–2526 (2005)
19. Cho, H., Swartzlander, E.: Adder designs and analyses for quantum-dot cellular automata. IEEE Trans. Nanotechnol. 6(3), 374–383 (2007)
20. Hänninen, I., Takala, J.: Pipelined array multiplier based on quantum-dot cellular automata. In: Proc. European Conf. Circuit Theory and Design, Seville, Spain, August 26-30, pp. 938–941 (2007)