Hindawi Publishing Corporation
Journal of Nanotechnology
Volume 2014, Article ID 259869, 14 pages
http://dx.doi.org/10.1155/2014/259869

Research Article

Nanosensor Data Processor in Quantum-Dot Cellular Automata

Fenghui Yao,[1] Mohamed Saleh Zein-Sabatto,[2] Guifeng Shao,[1] Mohammad Bodruzzaman,[2] and Mohan Malkani[2]

[1] *Department of Computer Science, College of Engineering, Tennessee State University, 3500 John A Merritt Blvd, Nashville, TN 37209, USA*
[2] *Department of Electrical and Computer Engineering, College of Engineering, Tennessee State University, 3500 John A Merritt Blvd, Nashville, TN 37209, USA*

Correspondence should be addressed to Fenghui Yao; fyao@tnstate.edu

Received 7 August 2013; Accepted 14 December 2013; Published 9 February 2014

Academic Editor: Valery Khabashesku

Quantum-dot cellular automata (QCA) is an attractive nanotechnology with the potential alterative to CMOS technology. QCA provides an interesting paradigm for faster speed, smaller size, and lower power consumption in comparison to transistor-based technology, in both communication and computation. This paper describes the design of a 4-bit multifunction nanosensor data processor (NSDP). The functions of NSDP contain (i) sending the preprocessed raw data to high-level processor, (ii) counting the number of the active majority gates, and (iii) generating the approximate sigmoid function. The whole system is designed and simulated with several different input data.

1. Introduction

Nanotechnology is a multidisciplinary field that brings together many science and engineering disciplines including physics, chemistry, biosciences, material science, computer science, and electrical and mechanical engineering. Nanotechnology will radically affect all these disciplines and their application areas. The economic impact is foreseen to be comparable to information technology and telecommunication industries [1]. Nanotechnology has direct applications in sensing, sensor miniaturization, and new materials development as well as in electronics and electromechanical devices. Substantial advances in nanotechnology development have been achieved in the fields of engineering and bioscience [2–4]. For example, integrating a large number (496) of programmable FET nodes in a small area of about 960 μm^2 and programing them into a full adder, subtractor, multiplexer, and demultiplexer has been reported. Work reported in [3] demonstrated that a lateral integration of 700 rows of Z_nO nanowires produces a peak voltage of 1.26 V at a low strain of 0.19%, which is potentially sufficient to recharge an AA battery. The nanosensor development described in [5], based on nanowires, is emerging as a powerful and general class of ultrasensitive, electrical sensors for the direct detection of biological and chemical species, from proteins and DNA to drug molecules and viruses down to the ultimate level of a single molecule. It also shows that a nanosensor array that contains 100 addressable elements provides unique opportunities for label-free multiplexed detection of biological and chemical species. The literature [6] provides an in-depth view of nanosensor technology and electromagnetic communication among nanosensors. With the integration of the technologies described in [3, 5, 6], that is, the integration of nanosensors, self-powered nanodevices, and wireless nanosensors, many nanosensor applications can be considered. These applications can be classified into four groups: biomedical, environmental, industrial, and military applications. In biomedical applications, health monitoring systems and drug delivery systems are some of the examples. In environmental applications, plant monitoring systems and plague defeating systems can be given as examples. In industrial and consumer goods applications, the examples are ultrahigh sensitivity touch surfaces and haptic interfaces. For military and defense applications, examples are nuclear, biological, and chemical defenses and damage detection systems.

In the above-mentioned applications, although the total number of nanosensors used in each case depends on the scale of the deployed devices, generally this number ranges from 10^3 to 10^7 or higher. These application devices have the following common characteristics.

(i) *Simple Data*. Each nanosensor delivers only simple data. The processing of the data provided by each individual nanosensor does not need complicated operation.

(ii) *Ambiguity*. Because some of the nanosensors may work at charging phase (self-powered) or the power may be low, nanosensors may provide ambiguous signals. The receiver (processing) side cannot obtain accurate data at all times. Also, like any other communication system, the signal may fade because of environment conditions and noise.

(iii) *Massive Data*. Nanosensing devices contain a large number of nanosensors (10^3–10^7 or higher) which generate data continuously. The data from each nanosensor may be primitive and ambiguous. However, massive amount of this type of data, if processed correctly, delivers useful and meaningful information from the device as a whole.

To analyze data from the nanosensing devices mentioned above, classical computing and processing models, such as loosely coupled distributed systems or tightly coupled parallel systems, can be employed. However, in the classical computing and processing models, the number of processors is usually in the range of 10^1–10^3. This number of processors can be increased, but it will bring tremendous increase of cost and system dimensions. Each processing unit usually has a local memory and can conduct complex operation independently. Also, they consume large amounts of energy (power) and heat dissipation becomes a critical issue. The above-mentioned computing and processing models are not suitable for processing data provided by nanosensors. The computing and processing models for nanosensing should meet the following three principles.

(i) *Simplicity*. The basic processing element (PE), that is, the cell, is simple. PE does not need complicated operations and does not need many instructions, as those in the existing general-purpose CPUs. It only needs a small number of operations because it will function as a bridge between nanosensors and high-level processors.

(ii) *Parallelism*. There will be a vast number of cells operating in parallel. Hence, the processing must be distributed.

(iii) *Locality*. All interactions take place on a purely local basis. A cell can interact with a few other cells.

ITRS report [7] summarizes several possible technology solutions for nanosensor data processing. Quantum-dot cellular automata (QCA) is an interesting possibility. Since QCAs were introduced in 1993 [8], several experimental devices have been developed [9–13]. Although they are certainly "not ready for prime time," recent papers show that QCAs may eventually achieve high density [14], fast switching speed [15], and room temperature operation [10, 16].

This paper describes the design of a PE for nanosensor data processing, named as nanosensor data processor (NSDP), based on QCA. NSDP works as a bridge between the nanosensors and high-level processor. Its functions are limited to three: (i) sending the preprocessed raw data to high-level processor, (ii) counting the number of the active majority gates (the active means that the output of a majority gate is 1), and (iii) generating the approximate sigmoid function for postprocessing based on artificial neural network. This paper is organized as follows. Section 2 presents the background of QCA focusing on its unique clocking scheme. Section 3 shows the details design of NSDP. Simulation results are shown in Section 4. Conclusions and future works are given in Section 5.

2. QCA Background

2.1. QCA Cells and Wires. A QCA cell is a square nanostructure with a quantum dot in each of the four corners [17], as shown schematically in Figure 1. The cell is populated with two electrons that can tunnel between two pairs of quantum dots connected via a tunnel junction. The two electrons occupy antipodal sites within the cell due to Coulombic repulsion. Tunneling action only occurs within the cell and no tunneling happens between cells. The combination of quantum confinement, Coulombic repulsion, and the discrete electronic charge produces bistable behavior. The two charge configurations can be used to represent binary "0" and "1" with polarization of –1 and +1, respectively. In contrast to a physical wire, a QCA "wire" is a chain of cells where the cells are adjacent to each other, as shown in Figure 1(b). Since no electrons tunnel between cells, QCA provides a mechanism for transferring information without current flow.

2.2. QCA Logic Gates. In QCA, three-input majority gates and inverters serve as the fundamental gates. A majority gate, as shown in Figure 2(a), consists of five QCA cells that realize the function of $M(a, b, c) = ab+bc+ac$. An inverter, as shown in Figure 2(b), is made by positioning cells diagonally from each other to achieve the inversion functionality. Figures 2(c) and 2(d) show the variation layouts of an inverter. Majority gates and inverters form a universal set; that is, any logic function can be implemented by using this set. For example, a two-input AND gate is realized by fixing one of the majority gate inputs to "0," that is, $\mathrm{AND}(a, b) = M(a, b, 0) = ab$. Similarly, an OR gate is realized by fixing one input to "1," that is, $\mathrm{OR}(a, b) = M(a, b, 1) = ab + b \cdot 1 + a \cdot 1 = a + b$.

2.3. QCA Clocking Scheme. Adiabatic switching is used for QCA clocking, which significantly reduces metastability issues and enables deep pipelines [18]. During each clock cycle, half of the wire is active for signal propagation, while the other half is unpolarized. During the next clock cycle, half of the previous active clock zone is deactivated and the remaining active zone cells trigger the newly activated cells to

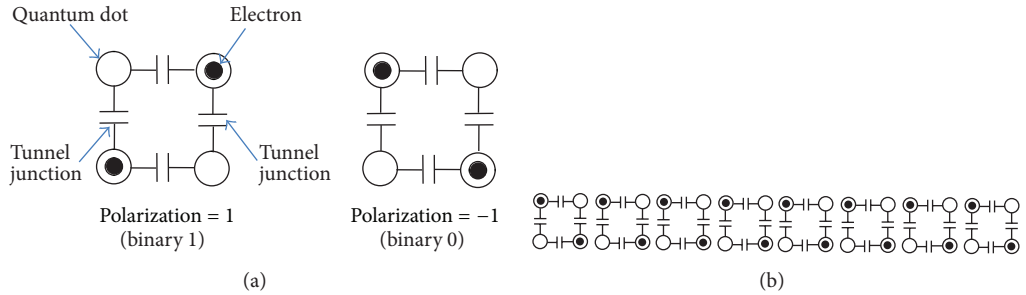


Figure 1: Basic QCA cell and wire.

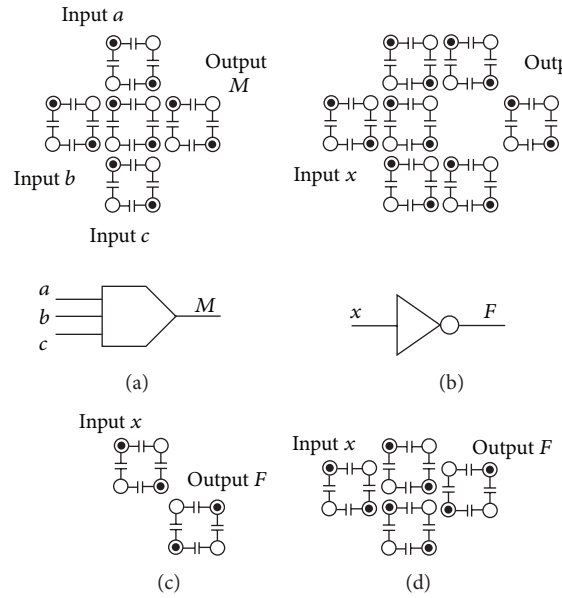


Figure 2: (a) Majority gate and its symbol; (b) inverter and its symbol; (c) and (d) variation layout of an inverter.

be polarized. Thus, signals propagate from one clock zone to the next. The circuit area is divided into four sections and they are driven by four phase clock signals, as shown in Figure 3. In each zone, the clock signal has four states: high-to-low, low, low-to-high, and high. The cell begins computing during the high-to-low state and holds the value during the low state. The cell is released when the clock is in the low-to-high state and inactive during the high state.

2.4. QCA Design Rules. A nominal cell size of 20 nm by 20 nm is assumed. The cell has a width and height of 18- and 5-nm-diameter quantum dots. The cells are placed on a grid with a cell center-to-center distance of 20 nm. QCA design rules are well studied in [19–21] and are summarized in the following.

(A) Layout Design Rules

(1) *Maximum Number of Cells in a Single Clocking Zone.* It can be as large as 47 cells; the maximum length of QCA wire is 25 cells. Any long QCA wire exceeding the length of 25 needs to be partitioned into different clocking zones.

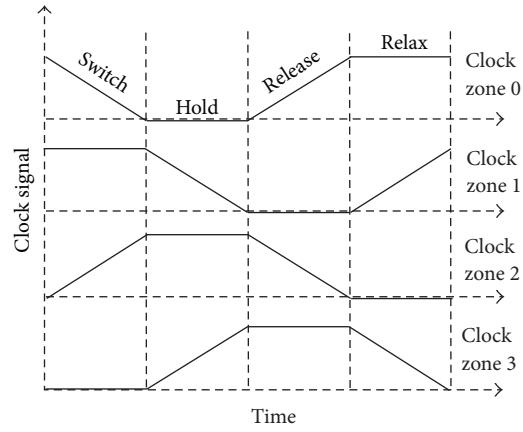


Figure 3: QCA clocking scheme.

(2) *Minimum Number of Cells in a Single Clocking Zone.* It can be one cell. However, the waveform of a one-cell clocking zone can become distorted and cascading of this kind of clocking zone could lead to incorrect

results [22]. To observe correct outputs from a circuit, it is recommended that clocking zones should consist of at least two cells.

(3) *Minimum Wire Spacing for Signal Separation*. A space of one QCA cell size is sufficient separation between two wires carrying different signals.

(4) *Wire Crossover*. A unique property of QCA layout is the possibility of implementing crossovers by using only one layer, known as coplanar crossing. Coplanar crossing uses both 45° and 90° cells. However, they can easily fail due to low robustness [23] and fabrication issues [24]. Another alternative is multilayer crossing, which uses more than one layer of cells similar to the routing of metal wires in CMOS technology. The extra layers of QCA are believed to be useful as active components of the circuits and consume less area compared to coplanar circuits [23].

(5) *QCA Equivalent λ-Rule*. "λ-rule" for QCA circuit design could be defined according to the size of a QCA cell, or perhaps the cell size itself could be used as the equivalent λ.

(B) Timing Design Rules

(1) *Logic Component Timing Rule*. The timing constraint on a QCA majority gate is that all three inputs are expected to reach the device cell (central cell) at the same time in order to have fair voting.

(2) *Clocking Zone Assignment Rule*. In QCA circuits, cells in each clocking zones should be synchronized.

(C) Special Rules for QCA

(1) *Majority Logic Reduction*. The logic primitive used in QCA is the majority gate. The majority logic-based reduction method [25] can significantly reduce the complexity of QCA circuits.

(2) *Systolic Design*. The features of systolic architecture in terms of synchrony, deep pipelines, and local interconnection are particularly suitable for accommodating the special timing requirement in QCA circuits. When applying systolic architecture to QCA, significant benefits can be achieved, even more than when applied to CMOS-based technology [26].

In the design of NSDP, the second rule (i.e., Minimum number of cells in a single clocking zone) in layout design is applied in the following way. For the fixed-value input, such as fixing one input of a majority gate to "–1.00" (binary 0) to make it an AND gate, it uses one-cell clocking zone. Or for the limited space, it uses one-cell clocking zone, but no cascading. For the 4th rule (i.e., wire crossover), it employs the multilayer crossing technique. The following section shows the details design of NSDP.

3. NSDP Architecture

NSDP is a processor that works as a bridge between nanosensors and the high-level processor. Its functions are (i) sending the preprocessed raw data to high-level processor, (ii) counting the number of the active majority gates, and (iii) generating the approximate sigmoid function. Among these functions, the last one is the focus of NSDP because it provides the sigmoid function output for the high-level processor to conduct the processing based on the artificial neural network (ANN). The block diagram of NSDP is shown in Figure 4. The detail of each block is explained below.

3.1. Preprocessing. Considering that a large number of nanosensors will generate a great amount of "0" or "1" data, and some nanosensors may generate ambiguous data, the first processing in NSDP computer is the majority operation that determines which one ("0" or "1") takes the majority. Generally, the number of the inputs of this unit can be any number. The larger the number of the inputs is, the better the majority operation is. However, considering the capability of the available QCA design tool, QCADesigner, the number of the inputs in NSDP is set at 12. This processing unit consists of four majority gates, and it has 12 inputs and 4 outputs. Figure 5(a) shows the schematic, Figure 5(b) shows the QCA layout, and Figure 5(c) shows the simulation result by QCADesigner, respectively. The output of this unit is given by

$$\begin{aligned} m_0 &= M(a_0, b_0, c_0), \\ m_1 &= M(a_1, b_1, c_1), \\ m_2 &= M(a_2, b_2, c_2), \\ m_3 &= M(a_3, b_3, c_3). \end{aligned} \tag{1}$$

As shown in Figure 5(c), for the input $a_0b_0c_0$ = {000, 001, 010, 001, 100, 101, 110, 111}, $a_1b_1c_1$ = {000, 100, 110, 101, 100, 011, 010, 011}, $a_2b_2c_2$ = {000, 111, 110, 101, 100, 011, 010, 001}, and $a_3b_3c_3$ = {000, 001, 010, 011, 100, 101, 110, 111}, the output m_0 = {0, 0, 0, 0, 0, 1, 1, 1}, m_1 = {0, 0, 1, 1, 0, 1, 0, 1}, m_2 = {0, 1, 1, 1, 0, 1, 0, 0}, m_3 = {0, 0, 0, 1, 0, 1, 1, 1}, respectively, and the delay is 3/4 clock.

3.2. Counter. The output of the preprocessing unit is separated into two, one goes to M-Latch and the other goes to the input of the counter. This unit is to count "1" in the output of the preprocessing unit. To count "1" from 4 parallel bits of the output of the preprocessing unit, it employs three full adders, as shown in Figure 6(a). QCA full adder has been well studied. Some representative QCA adders are [19, 27–31]. The QCA carry flow adder reported in [19] is a layout optimized multilayer full adder. In QCA, the path from carry-in to carry-out uses one majority gate that requires one clocking zone per bit in a ripple carry adder. This adder (referred to as Cho adder in this paper) consumes only one clocking zone delay per bit, which significantly reduces the delay for large adders. Three Cho adders are employed to implement the counter. The QCA layout is shown in Figure 6(b), and the simulation result shown in Figure 6(c). The output is given by

$$s_2s_1s_0 = m_3 + m_2 + m_1 + m_0. \tag{2}$$

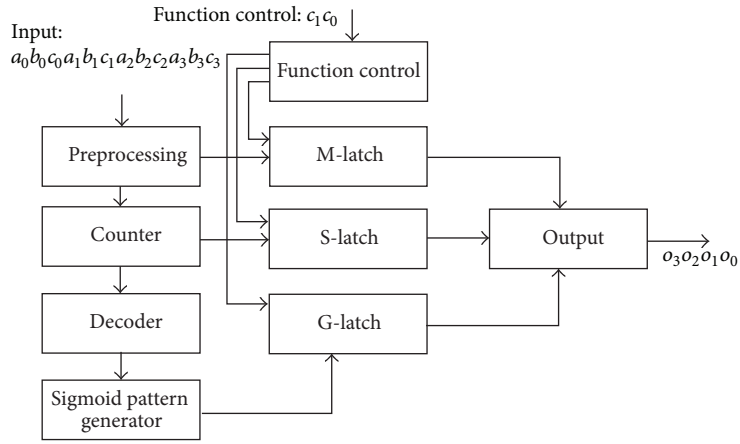


Figure 4: Block diagram of NSDP.

As shown in Figure 6(c), for the input $m_0 = \{0, 0, 0, 1, 0, 1, 1, 1\}$, $m_1 = \{0, 1, 1, 1, 0, 1, 0, 0\}$, $m_2 = \{0, 1, 1, 1, 0, 1, 0, 0\}$, $m_3 = \{0, 0, 0, 1, 0, 1, 1, 1\}$, the output $s_2s_1s_0 = \{0, 2, 2, 4, 0, 4, 2, 2\}$. The delay is 2 clocks. Note that the "+" operator here means arithmetic addition. In the following, "+" means logic OR operation if there is no specific notation.

3.3. Decoder. The decoder unit and the sigmoid pattern generator unit are used to approximate the sigmoid function $y = 1/(1 + e^{-x})$. As shown in Table 1, NSDP employs 4 points to approximate the sigmoid function. This is 3-to-4 decoder; the inputs are s_2, s_1, and s_0, and the outputs are f_4, f_3, f_2, and f_1. The output f_1 corresponds to that $s_2s_1s_0$ is equal to 0 or 1; that is, it corresponds to that $m_3m_2m_1m_0$ equals to 1000, 0100, 0010, 0001, or 0000. It is worth to note that the position of "1" does not matter. For the four inputs, m_3, m_2, m_1, and m_0, of the counter, m_3 does not mean the most significant bit and m_0 does not mean the least significant bit because the twelve input sensors can only be considered "fired" or "not fired" and the majority gate can be numbered in any order. This is the same here and after. Likewise, the output f_2 corresponds to that $s_2s_1s_0$ which is equal to 2, that is, it corresponds to that $m_3m_2m_1m_0$ which is equal to 1100, 0110, 0011, 1010, 0101, or 1001. The output f_3 corresponds to that $s_2s_1s_0$, is equal to 3, that is, it corresponds to that $m_3m_2m_1m_0$ which is equal to 1110, or 0111. The output f_4 corresponds to that $s_2s_1s_0$ is equal to 4; that is, it corresponds to that $m_3m_2m_1m_0$ equals to 1111. The outputs f_4, f_3, f_2, and f_1 are given by

$$\begin{aligned} f_1 &= \overline{s_2}\,\overline{s_1}, \\ f_2 &= \overline{s_2}s_1\overline{s_0}, \\ f_3 &= \overline{s_2}s_1s_0, \\ f_4 &= s_2\overline{s_1}\,\overline{s_0}. \end{aligned} \tag{3}$$

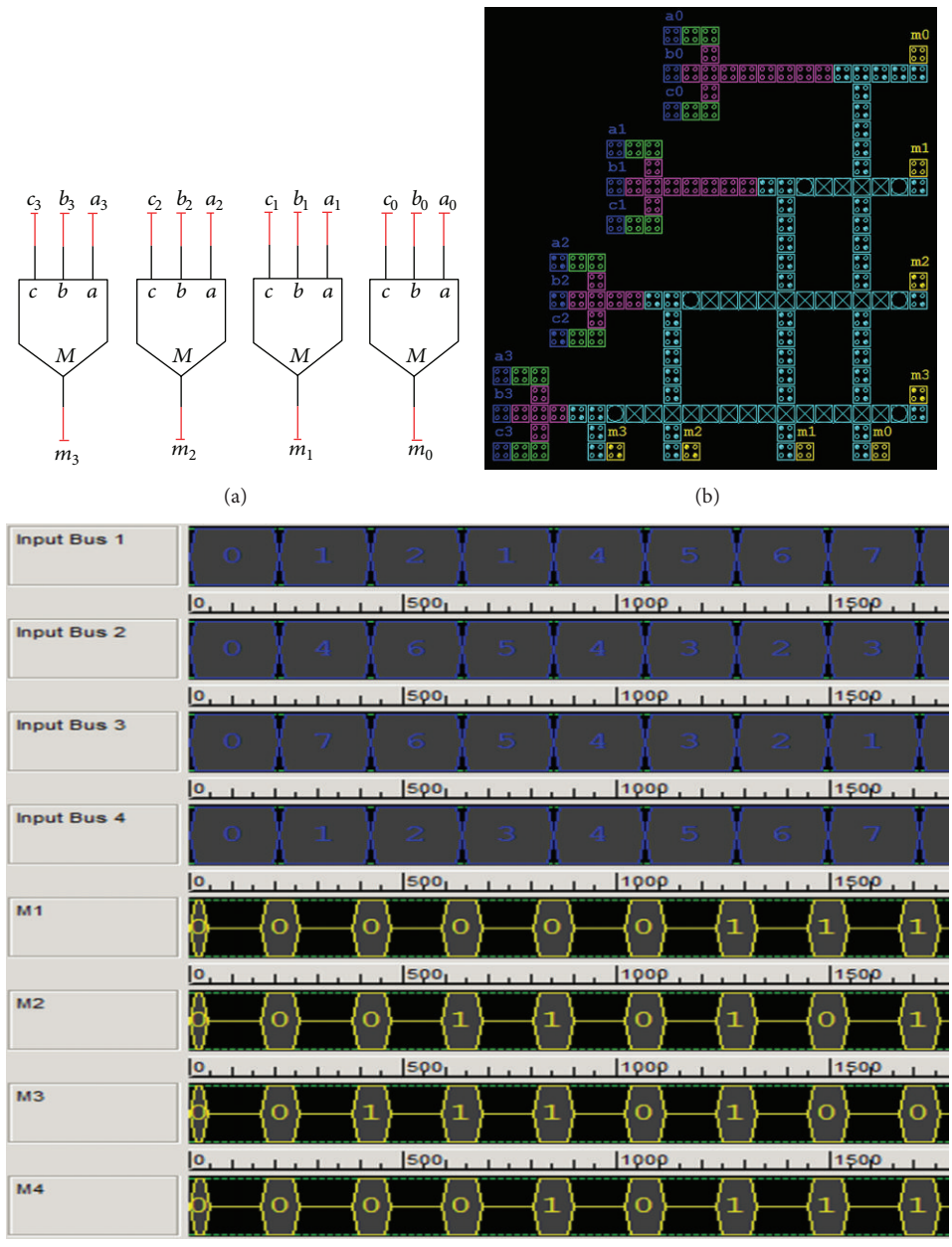
Figure 7(a) shows the schematic of this 3-to-4 decoder. Figure 7(b) shows the QCA layout, and Figure 7(c) shows the simulation result. As shown in Figure 7(c), when the input $s_2s_1s_0$ (sum) is equal 0, 1, 2, 4, 0, 3, 1, and 2, the output $f_4f_3f_2f_1$ will be equal to 0001, 0001, 0010, 1000, 0001, 0100, 0001, and 0010, respectively, and the delay is one clock.

Table 1: Decoding of the output of the counter.

$s_2s_1s_0$	Output: $f_4f_3f_2f_1$	Sigmoid function pattern
The number of "1" is less than or equal to 1 (the position does not matter).	0001	0000
The number of "1" is equal to 2 (the position does not matter).	0010	0011
The number of "1" is equal to 3 (the position does not matter).	0100	1101
The number of "1" is equal to 4.	1000	1111

3.4. Pattern Generator. The output of the decoder is used to generate four patterns, 1111, 1101, 0011, and 0000, as listed in the third column in Table 1, which are used to approximate the sigmoid function. These four patterns are generated in the following way:

$$\begin{aligned} f_{4a} &= f_4 \cdot 1, \\ f_{4b} &= f_4 \cdot 1, \\ f_{4c} &= f_4 \cdot 1, \\ f_{4d} &= f_4 \cdot 1, \\ f_{3a} &= f_3 \cdot 1, \\ f_{3b} &= f_3 \cdot 1, \\ f_{3c} &= f_3 \cdot 0, \\ f_{3d} &= f_3 \cdot 1, \\ f_{2a} &= f_2 \cdot 0, \\ f_{2b} &= f_2 \cdot 0, \end{aligned}$$



(a) (b) (c)

Figure 5: Preprocessing. (a) Schematic; (b) QCA layout; (c) simulation result.

$$
\begin{aligned}
f_{2c} &= f_2 \cdot 1,\\
f_{2d} &= f_2 \cdot 1,\\
f_{1a} &= f_1 \cdot 0,\\
f_{1b} &= f_1 \cdot 0,\\
f_{1c} &= f_1 \cdot 0,\\
f_{1d} &= f_1 \cdot 0.
\end{aligned} \tag{4}
$$

The schematics for $f_{4a}f_{4b}f_{4c}f_{4d}$, $f_{3a}f_{3b}f_{3c}f_{3d}$, $f_{2a}f_{2b}f_{2c}f_{2d}$ and $f_{1a}f_{1b}f_{1c}f_{1d}$ are shown in Figure 8(a). The QCA layout for $f_{4a}f_{4b}f_{4c}f_{4d}$ is shown in Figure 8(b). The layout for $f_{3a}f_{3b}f_{3c}f_{3d}$, $f_{2a}f_{2b}f_{2c}f_{2d}$ and $f_{1a}f_{1b}f_{1c}f_{1d}$ is similar to the one in Figure 8(b), except that the pattern to be generated is 1101, 0011, and 0000, respectively. The simulation result is shown in Figure 8(c). When the input $s_2s_1s_0$ (sum) is equal to 0, 1, 2, 4, 0, and 3, the generated pattern is 0000 (0) and 0000 (0) for $f_{1a}f_{1b}f_{1c}f_{1d}$, 0011 (3) for $f_{2a}f_{2b}f_{2c}f_{2d}$, 1111 (15) for $f_{4a}f_{4b}f_{4c}f_{4d}$, 0000 (0) for $f_{1a}f_{1b}f_{1c}f_{1d}$, and 1101 (13) for $f_{3a}f_{3b}f_{3c}f_{3d}$, respectively. The delay is 3/4 clock. Note that the delay for $f_{2a}f_{2b}f_{2c}f_{2d}$ and $f_{1a}f_{1b}f_{1c}f_{1d}$ is one clock longer than of $f_{4a}f_{4b}f_{4c}f_{4d}$ and $f_{3a}f_{3b}f_{3c}f_{3d}$. This is because that the input for $f_{2a}f_{2b}f_{2c}f_{2d}$ and $f_{1a}f_{1b}f_{1c}f_{1d}$ is delayed by one clock.

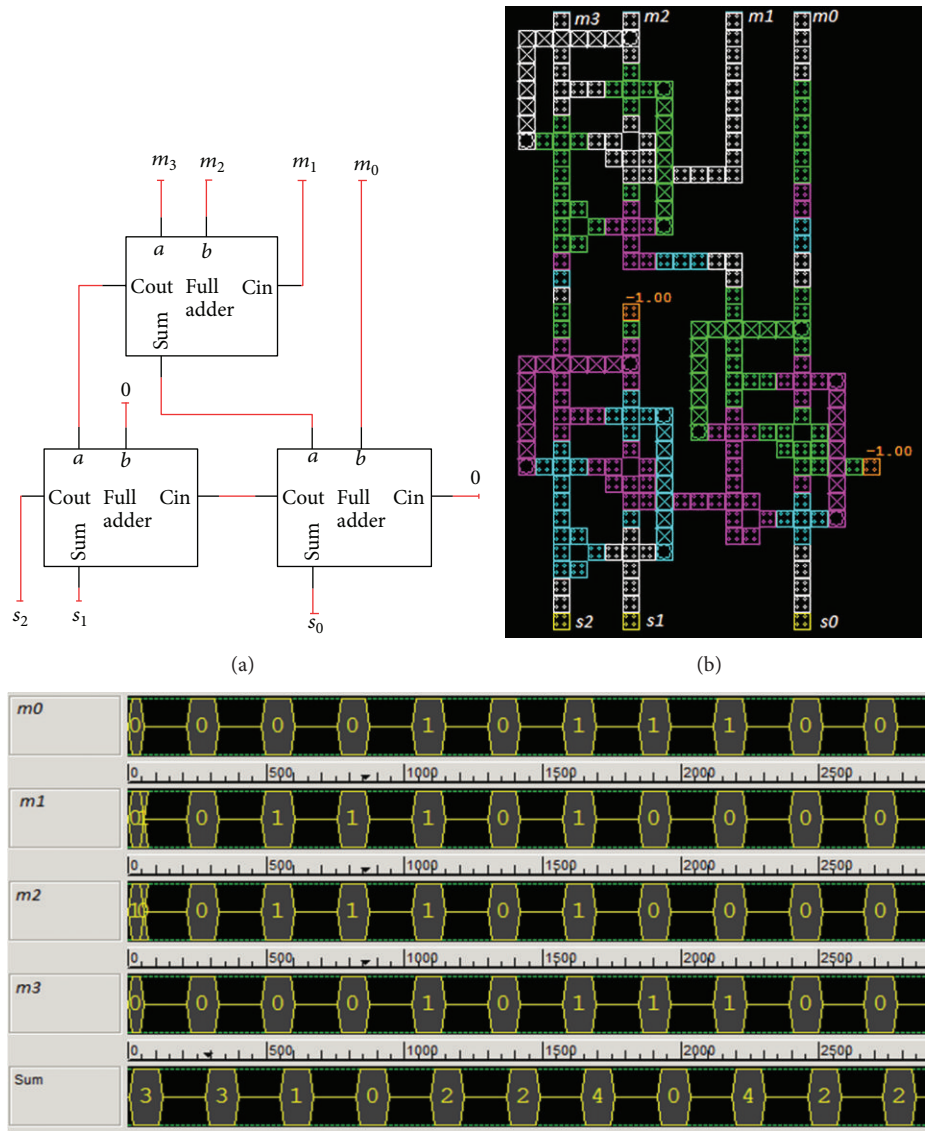


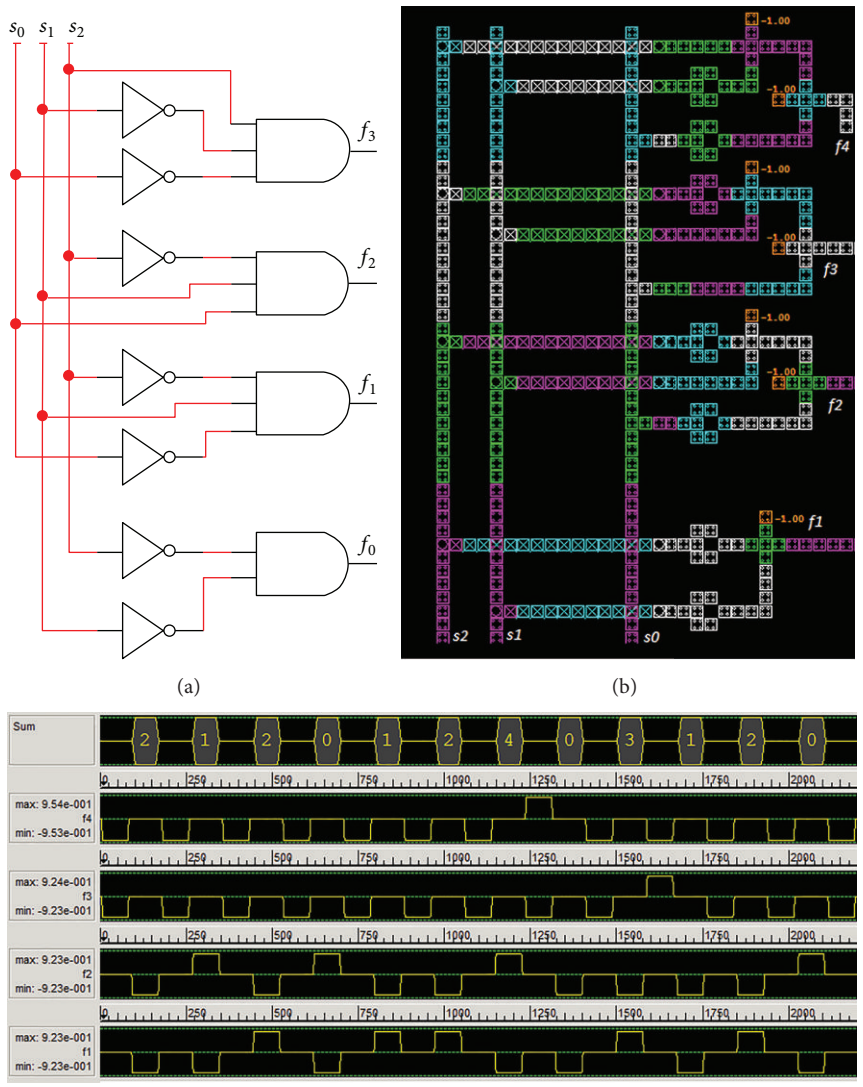
FIGURE 6: Counter. (a) Schematic; (b) QCA layout; (c) simulation result.

3.5. Sigmoid Function Output. The output of the sigmoid function is one of the four patterns, $f_{4a}f_{4b}f_{4c}f_{4d}$, $f_{3a}f_{3b}f_{3c}f_{3d}$, $f_{2a}f_{2b}f_{2c}f_{2d}$, and $f_{1a}f_{1b}f_{1c}f_{1d}$, depending on the number of 1's of the counter. Therefore, the output $g_3g_2g_1g_0$ of the sigmoid function can be written as

$$\begin{aligned} g_3 &= f_{4a} + f_{3a} + f_{2a} + f_{1a}, \\ g_2 &= f_{4b} + f_{3b} + f_{2b} + f_{1b}, \\ g_1 &= f_{4c} + f_{3c} + f_{2c} + f_{1c}, \\ g_0 &= f_{4d} + f_{3d} + f_{2d} + f_{1d}. \end{aligned} \tag{5}$$

The schematics for g_i ($i = 0, 1, 2,$ and 3) are shown in Figure 9(a). The QCA layout for g_0 are shown in Figure 9(b). This layout is also for other outputs g_i ($i = 1, 2,$ and 3), except that the inputs and output will be changed to $f_{4c}, f_{3c}, f_{2c}, f_{1c}, g_1, f_{4b}, f_{3b}, f_{2b}, f_{1b}, g_2$ and $f_{4a}, f_{3a}, f_{2a}, f_{1a}, g_3$, correspondingly. The simulation result is shown in Figure 9(c). When the input $s_2s_1s_0$ (sum) is equal to 0, 1, 2, 4, 0, and 3, the output $g_3g_2g_1g_0$ is 0000 (0), 0000 (0), 0011 (3), 1111 (15), 0000 (0), and 1101 (13), correspondingly. The delay from the output of the counter to the output of the sigmoid function is 2(3/4) clocks. Figure 10 shows the graph of the approximated sigmoid function, generated by the decoder and pattern generator unit described above.

3.6. Function Control. NSDP is a multifunction nanosensor data processor. The function selection of NSDP is determined by the function control unit, which is a 2-to-3 decoder. The truth table of the function control unit is shown in Table 2. When control bus c_1c_0 is equal to 00, the output of NSDP is the raw majority gate output, that is, $m_3m_2m_1m_0$ from 4



(a) (b)

(c)

Figure 7: 3-to-4 Decoder. (a) Schematic; (b) QCA layout; (c) simulation result.

independent majority gates. The bit position does not matter; that is, m_3 does not mean MSB and m_0 does not mean LSB. When control bus c_1c_0 is equal to 01, the output of NSDP is the number of 1s of the majority gate output, given by $0s_2s_1s_0$. Note that here s_0 is LSB and MSB is always 0. When control bus is c_1c_0 equal to 10, the output of NSDP is the sigmoid function output. The output of the function control unit is given by

$$\begin{aligned} s_{00} &= \overline{c_1}\,\overline{c_0}, \\ s_{01} &= \overline{c_1}c_0, \\ s_{10} &= c_1\overline{c_0}. \end{aligned} \tag{6}$$

The schematic of the function control unit is shown in Figure 11(a), the QCA layout is shown in Figure 11(b), and the simulation result is shown in Figure 11(c). When input c_1c_0 is equal to 00, 01, and 10, the output $s_{10}s_{01}s_{00}$ is equal to 001, 010, and 001, respectively. The delay from the input to output is 1 clock.

Table 2: Truth table for function control unit.

c_1c_0	Output of NSDP
00	Majority of raw sensor data, $m_3m_2m_1m_0$
01	Number of active outputs (i.e., 1s), $0s_2s_1s_0$
10	Sigmoid output, $g_3g_2g_1g_0$
11	Reserved

3.7. NSDP Output. The function control unit of NSDP determines which output of the three function units (raw majority data, the number of the active majority gates, and sigmoid

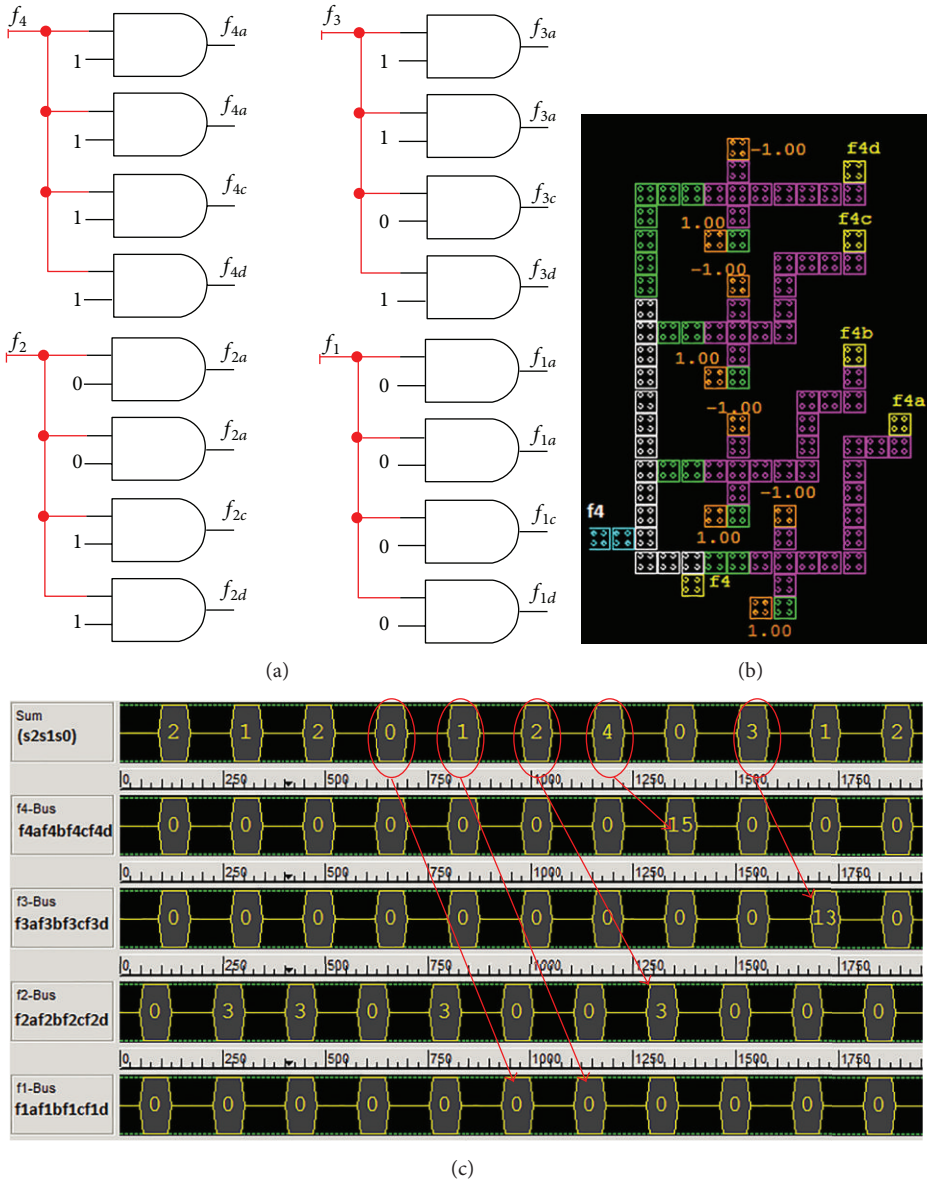


FIGURE 8: Pattern generator. (a) Schematic for 1111, 1101, 0011, and 0000 generators; (b) QCA layout for 1111 generator (this QCA layout can generate other patterns by changing the standard pattern to 1101, 0011, or 0000); (c) simulation result for 1111 (15), 1101 (13), 0011 (3), and 0000 (0) generators.

function) will be the output $o_3o_2o_1o_0$ of NSDP. $o_3o_2o_1o_0$ is given by

$$\begin{aligned} o_3 &= m_3 s_{00} + 0 \cdot s_{01} + g_3 s_{10}, \\ o_2 &= m_2 s_{00} + s_2 \cdot s_{01} + g_2 s_{10}, \\ o_1 &= m_1 s_{00} + s_1 \cdot s_{01} + g_1 s_{10}, \\ o_0 &= m_0 s_{00} + s_0 \cdot s_{01} + g_0 s_{10}. \end{aligned} \tag{7}$$

The schematic of NSDP output unit is shown in Figure 12(a), and the partial QCA layout of o_0 in is shown Figure 12(b). Note that m_{0a}, s_{0a}, and g_{0a} are the outputs of M-latch, S-latch, and G-latch for the inputs m_0, s_0, and g_0, respectively, which are not shown in Figure 12(b) (refer to Figure 13 for details).

The complete QCA layout of NSDP is shown in Figure 13. NSDP consists of the following units: preprocessing, counter, 3-to-4 decoder, pattern generator, sigmoid function output, function control, M-latch, S-latch, G-latch, and NSDP output. Each unit is surrounded by a red dotted line in Figure 13. The complete experiment results are given in the next section.

4. Results

NSDP is designed and simulated by using QCADesigner Ver. 2.0.3. NSDP contains 4436 cells. Its dimension is $3\mu m \times 2.3\,\mu m$ ($6.9\,\mu m^2$). The simulation employs the coherence

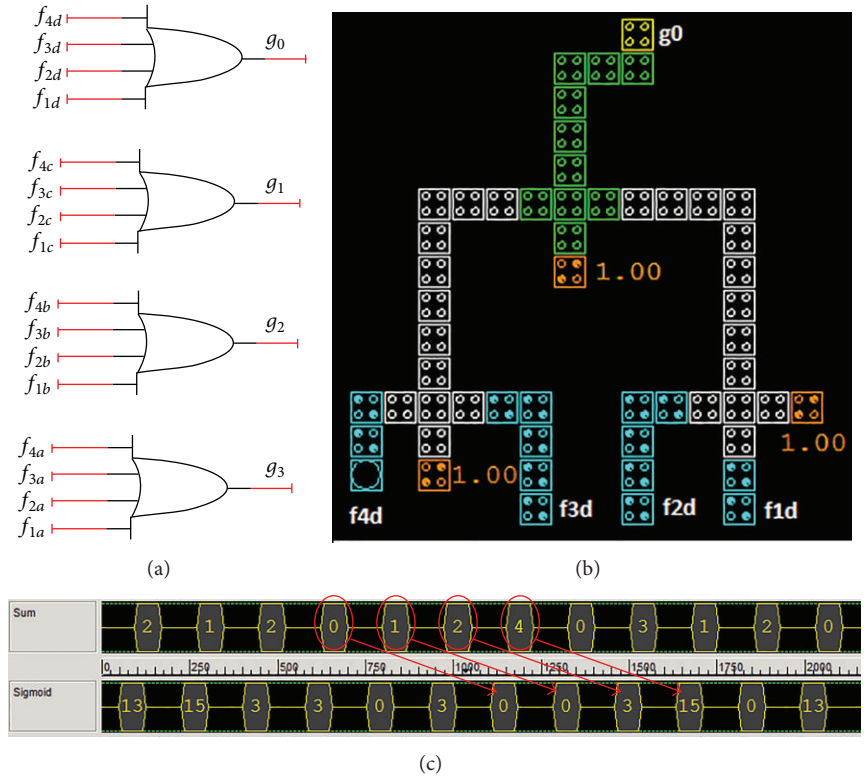


FIGURE 9: Output of sigmoid function. (a) Schematic for the output of the sigmoid function; (b) QCA layout for g_0 of the sigmoid function (this QCA layout is applicable to g_1, g_2, and g_3 by changing to the corresponding input); (c) simulation result of the sigmoid function output.

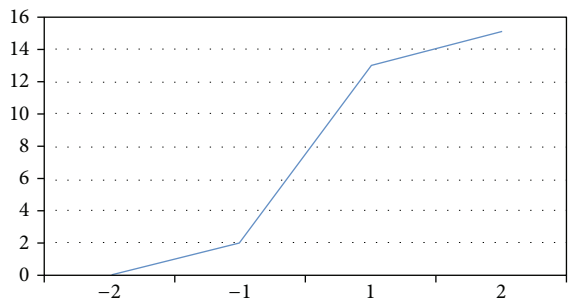


FIGURE 10: Approximate sigmoid function generated by NSDP.

vector engine. Figure 14 shows the parameter setting for coherence vector simulation engine.

4.1. Raw Majority Data Output (c_1c_0 = 00). In this mode, the data flow path is Preprocessing → M-Latch → NSDP Output (refer to Figure 13 for details).

As shown in Figure 15, 12 inputs are grouped into 4 groups: *Input* 3 ($a_3b_3c_3$), *Input* 2 ($a_2b_2c_2$), *Input* 1 ($a_1b_1c_1$), and *Input* 0 ($a_0b_0c_0$). The input data are as follows:

Input 3: 011 (3), 100 (4), 101 (5), 110 (6), 111 (7), ...

Input 2: 101 (5), 100 (4), 011 (3), 010 (2), 001 (1),...

Input 1: 101 (5), 100 (4), 011 (3), 010 (2), 001 (1), ...

Input 0: 011 (3), 100 (4), 001 (1), 010 (2), 110 (6), ...

The majority output on M bus ($m_3m_2m_1m_0$) is equal to 1111 (15), 0000 (0), 1110 (14), 1000 (8), 1001 (9), ... When c_1c_0 is equal to 00, s_{00} becomes 1, and s_{01} and s_{10} both become 0. NSDP picks the data on M bus as the final output. This is shown in 6th row, in Figure 15; that is, *Output* ($o_3o_2o_1o_0$) is equal to 1111 (15), 0000 (0), 1110 (14), 1000 (8), 1001 (9), ..., in accompany with the clocked s_{00}.

4.2. Counter Output (c_1c_0 = 01). In this mode, the data flow path is Preprocessing → Counter → S-Latch → NSDP Output (refer to Figure 13 for details).

As shown in Figure 16, the input data are as follows:

Input 3: 000 (0), 001 (1), 010 (2), 011 (3), 100 (4), 101 (5), 110 (6), 111 (7), ...

Input 2: 000 (0), 111 (7), 110 (6), 101 (5), 100 (4), 011 (3), 010 (2), 001 (1), ...

Input 1: 000 (0), 100 (4), 110 (6), 101 (5), 100 (4), 011 (3), 010 (2), 001 (1), ...

Input 0: 000 (0), 001 (1), 010 (2), 011 (3), 100 (4), 001 (1), 010 (2), 110 (6), ...

The counter output on sum bus ($s_2s_1s_0$) is equal to 000 (0), 001 (1), 010 (2), 100 (4), 000 (0), 011 (3), 001 (1), 010 (2), ... When c_1c_0 is equal to 01, s_{01} becomes 1, and s_{00} and s_{10} both become 0. NSDP picks the data on S bus and extends to 4 bits by padding leftmost bit with 0, as the final output. This is shown in 6th row, in Figure 16; that is, the *Output* ($o_3o_2o_1o_0$)

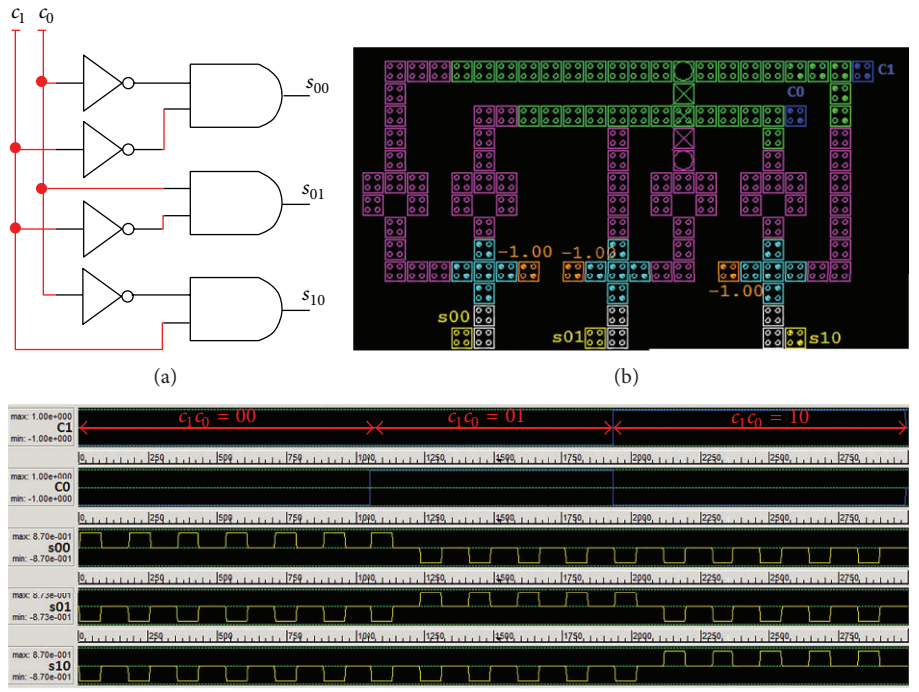


FIGURE 11: Function control unit. (a) Schematic; (b) QCA layout; (c) Simulation result for $c_1c_0 = 00$, $c_1c_0 = 01$, and $c_1c_0 = 10$.

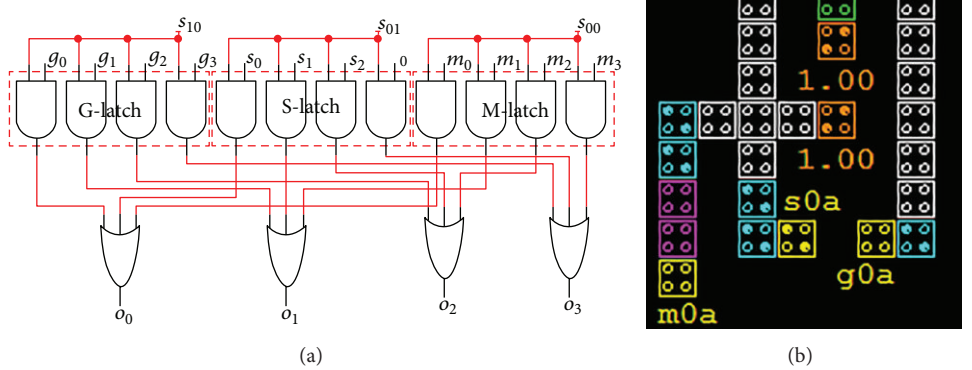


FIGURE 12: NSDP output. (a) Schematic; (b) Partial QCA layout for o_0.

is equal to 0000 (0), 0001 (1), 0010 (2), 0100 (4), 0000 (0), 0011 (3), 0001 (1), 0010 (2), . . ., in accompany with the clocked s_{01}.

4.3. Sigmoid Function Output (c_1c_0 = 10). In this mode, the data flow path is Preprocessing → Counter → 3-to-4 Decoder, → Pattern Generator → Sigmoid Function Output → G-Latch → NSDP Output (refer to Figure 13 for details).

As shown in Figure 17, the input data are as follows:

Input 3: 000 (0), 001 (1), 010 (2), 011 (3), 100 (4), 101 (5), 110 (6), 111 (7), . . .

Input 2: 000 (0), 111 (7), 110 (6), 101 (5), 100 (4), 011 (3), 010 (2), 001 (1), . . .

Input 1: 000 (0), 100 (4), 110 (6), 101 (5), 100 (4), 011 (3), 010 (2), 001 (1), . . .

Input 0: 000 (0), 001 (1), 010 (2), 011 (3), 100 (4), 001 (1), 010 (2), 110 (6), . . .

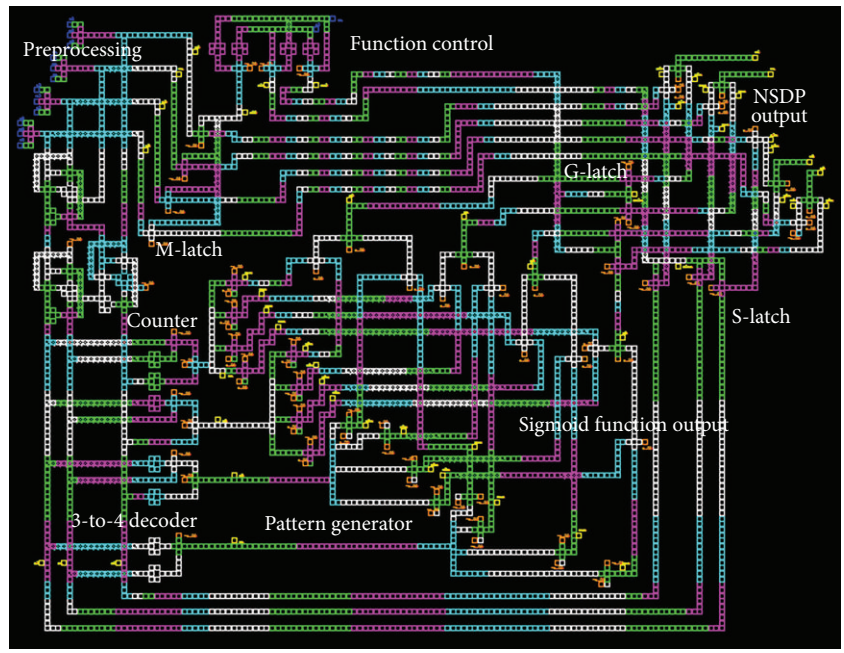


Figure 13: Complete QCA layout of NSDP.

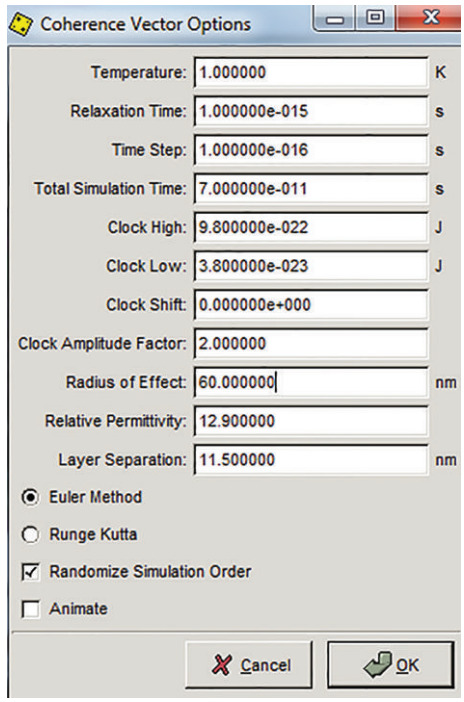


Figure 14: Parameters for coherence vector engine.

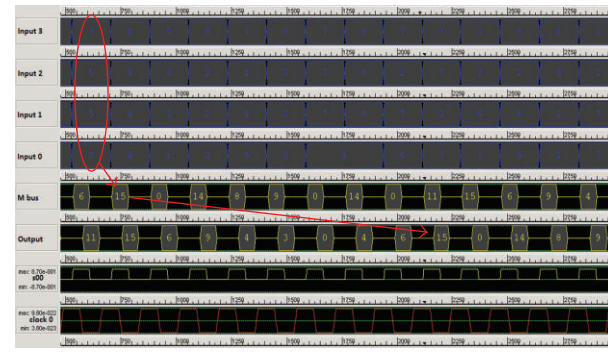


Figure 15: Simulation result for $c_1c_0 = 00$.

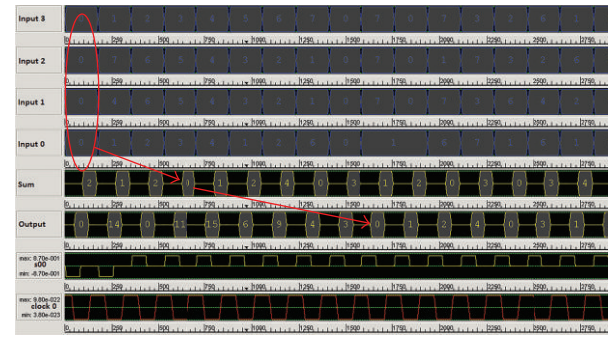


Figure 16: Simulation result for $c_1c_0 = 01$.

The counter output on sum bus ($s_2s_1s_0$) is equal to 000 (0), 001 (1), 010 (2), 100 (4), 000 (0), 011 (3), 001 (1), 010 (2), . . ., as shown in 5th row, in Figure 17. The sigmoid function converts these inputs to 0000 (0), 0000 (0), 0011 (3), 1111 (15), 0000 (0), 1101 (13), 0000 (0), 0000 (0), . . ., and outputs on the Sigmoid bus, as shown in 6th row. When c_1c_0 is equal to 10, s_{10} becomes 1, and s_{00} and s_{01} both become 0. NSDP picks the data on the sigmoid bus as the final output. This is shown in 7th row, in Figure 17; that is, the *Output* ($o_3o_2o_1o_0$) is equals to 0000 (0), 0000 (0), 0011 (3), 1111 (15), 0000 (0), 1101 (13), 0000 (0), 0000 (0), . . ., in accompany with the clocked s_{10}.

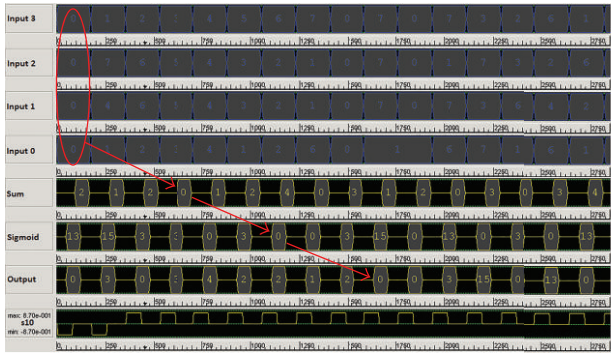


FIGURE 17: Simulation result for $c_1c_0 = 10$.

In above experiments, the processing in sigmoid function is the most complicated and the delay from input to the output is the longest. For the raw majority data output and the counter output, the delay is inserted to make them synchronize with the output of the sigmoid function output. Therefore, the delay of NSDP from the *Input* 3 ($a_3b_3c_3$), *Input* 2 ($a_2b_2c_2$), *Input* 1 ($a_1b_1c_1$), and *Input* 0 ($a_0b_0c_0$) to the *Output* ($o_3o_2o_1o_0$) is 8(3/4) clocks.

5. Conclusions and Future Works

This paper presents the design of a 4-bit nanosensor data processor (NSDP) which has 3 functions: (i) sending the result of 4 majority gates to high-level processor; (ii) counting the number of active majority gates (i.e., the output of the majority gate is 1); (iii) generating the approximate sigmoid function for the postprocessing based on the artificial neural network. QCA circuits have significant wire delays. For a fast design in QCA, it is generally necessary to minimize the complexity. The design uses systolic array structures to produce an output on every clock cycle with low latency to the first output. The layouts and functionality checks were done using QCADesigner. NSDP is a relatively complicated system. The simulation results show that its three functions work correctly. It is hoped that this paper will inspire further ideas on developing application systems based on QCA circuits.

As mentioned in Section 1, the nanosensor system generates a large number of nanosensor data. This type of system needs 10^3–10^7 PEs to process the real-time data. NSDP described in this paper works as a PE. A large number of PEs (10^3–10^7) need to be arranged in a parallel and distributed manner to handle the huge amount of nanosensor data. NSDP is a 4-bit processor. It will be more convenient to handle large amount of nanosensor data, by extending it to a 8-bit processor. All these are our future works.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by a Grant from AFRL under Minority Leaders Program, Contract no. TENN13-S567-019-02-C2. The authors would like to thank the anonymous reviewers for their careful review and valuable comments.

References

[1] V. Ermolov, M. Heino, A. Kärkkäinen et al., "Significance of nanotechnology for future wireless devices and communications," in *Proceedings of the 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '07)*, Athens, Greece, September 2007.

[2] J. N. Anker, W. P. Hall, O. Lyandres, N. C. Shah, J. Zhao, and R. P. Van Duyne, "Biosensing with plasmonic nanosensors," *Nature Materials*, vol. 7, no. 6, pp. 442–453, 2008.

[3] S. Xu, Y. Qin, C. Xu, Y. Wei, R. Yang, and Z. L. Wang, "Self-powered nanowire devices," *Nature Nanotechnology*, vol. 5, no. 5, pp. 366–373, 2010.

[4] H. Yan, H. S. Choe, S. Nam et al., "Programmable nanowire circuits for nanoprocessors," *Nature*, vol. 470, no. 7333, pp. 240–244, 2011.

[5] F. Patolsky and C. M. Lieber, "Nanowire nanosensors," *Materials Today*, vol. 8, no. 4, pp. 20–28, 2005.

[6] I. F. Akyildiz and J. M. Jornet, "Electromagnetic wireless nanosensor networks," *Nano Communication Networks*, vol. 1, no. 1, pp. 3–19, 2010.

[7] International Technology Roadmap for Semiconductors (ITRS), 2007, http://www.itrs.net/.

[8] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, "Quantum cellular automata," *Nanotechnology*, vol. 4, no. 1, pp. 49–57, 1993.

[9] I. Amlani, A. O. Orlov, R. K. Kummamuru, G. H. Bernstein, C. S. Lent, and G. L. Snider, "Experimental demonstration of a leadless quantum-dot cellular automata cell," *Applied Physics Letters*, vol. 77, no. 5, pp. 738–740, 2000.

[10] R. P. Cowburn and M. E. Welland, "Room temperature magnetic quantum cellular automata," *Science*, vol. 287, no. 5457, pp. 1466–1468, 2000.

[11] R. K. Kummamuru, A. O. Orlov, R. Ramasubramaniam, C. S. Lent, G. H. Bernstein, and G. L. Snider, "Operation of a quantum-dot cellular automata (QCA) shift register and analysis of errors," *IEEE Transactions on Electron Devices*, vol. 50, no. 9, pp. 1906–1913, 2003.

[12] A. O. Orlov, I. Amlani, G. Toth, C. S. Lent, G. H. Bernstein, and G. L. Snider, "Experimental demonstration of a binary wire for quantum-dot cellular automata," *Applied Physics Letters*, vol. 74, no. 19, pp. 2875–2877, 1999.

[13] H. Qi, S. Sharma, Z. Li et al., "Molecular quantum cellular automata cells. Electric field driven switching of a silicon surface bound array of vertically oriented two-dot molecular quantum cellular automata," *Journal of the American Chemical Society*, vol. 125, no. 49, pp. 15250–15259, 2003.

[14] A. DeHon and M. J. Wilson, "Nanowire-based sublithographic programmable logic arrays," in *Proceedings of the ACM/SIGDA 12th ACM International Symposium on Field-Programmable Gate Arrays (FPGA '04)*, pp. 123–132, February 2004.

[15] J. M. Seminario, P. A. Derosa, L. E. Cordova, and B. H. Bozard, "A molecular device operating at terahertz frequencies: theoretical simulations," *IEEE Transactions on Nanotechnology*, vol. 3, no. 1, pp. 215–218, 2004.

[16] Y. Wang and M. Lieberman, "Thermodynamic behavior of molecular-scale quantum-dot cellular automata (QCA) wires and logic devices," *IEEE Transactions on Nanotechnology*, vol. 3, no. 3, pp. 368–376, 2004.

[17] A. O. Orlov, I. Amlani, G. H. Bernstein, C. S. Lent, and G. L. Snider, "Realization of a functional cell for quantum-dot cellular automata," *Science*, vol. 277, no. 5328, pp. 928–930, 1997.

[18] C. S. Lent and P. D. Tougaw, "A device architecture for computing with quantum dots," *Proceedings of the IEEE*, vol. 85, no. 4, pp. 541–557, 1997.

[19] H. Cho and E. E. Swartzlander Jr., "Adder and multiplier design in quantum-dot cellular automata," *IEEE Transactions on Computers*, vol. 58, no. 6, pp. 721–727, 2009.

[20] W. Liu, L. Lu, M. O'Neill, and E. E. Swartzlander, "Design rules for quantum-dot cellular automata," in *Proceedings of the IEEE International Symposium of Circuits and Systems (ISCAS '11)*, pp. 2361–2364, May 2011.

[21] W. Liu, L. Lu, M. Orneill, E. E. Swartzlander Jr., and R. Woods, "Design of quantum-dot cellular automata circuits using cut-set retiming," *IEEE Transactions on Nanotechnology*, vol. 10, no. 5, pp. 1150–1160, 2011.

[22] K. Kim, K. Wu, and R. Karri, "Towards designing robust QCA architectures in the presence of sneak noise paths," in *Proceedings of the Design, Automation and Test in Europe (DATE '05)*, vol. 2, pp. 1214–1219, March 2005.

[23] K. Walus and G. A. Jullien, "Design tools for an emerging SoC technology: quantum-dot cellular automata," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1225–1243, 2006.

[24] M. Crocker, M. Niemier, X. S. Hu, and M. Lieberman, "Molecular QCA design with chemically reasonable constraints," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 4, no. 2, article 9, 2008.

[25] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "A method of majority logic reduction for quantum cellular automata," *IEEE Transactions on Nanotechnology*, vol. 3, no. 4, pp. 443–450, 2004.

[26] L. Lu, W. Liu, M. O'Neill, and E. E. Swartzlander Jr., "QCA systolic matrix multiplier," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI '10)*, pp. 149–154, July 2010.

[27] I. Hanninen and J. Takala, "Robust adders based on quantum-dot cellular automata," in *Proceedings of the IEEE International Conference on Application-Specific Systems, Architecture Processors*, pp. 391–396, 2007.

[28] V. Pudi and K. Sridharan, "Low complexity design of ripple carry and brent-kung adders in QCA," *IEEE Transactions on Nanotechnology*, vol. 11, no. 1, pp. 105–119, 2012.

[29] P. D. Tougaw and C. S. Lent, "Logical devices implemented using quantum cellular automata," *Journal of Applied Physics*, vol. 75, no. 3, pp. 1818–1825, 1994.

[30] W. Wang, K. Walus, and G. Jullien, "Quantum-dot cellular automata adders," in *Proceedings of the 3rd IEEE International Conference on Nanotechnology*, pp. 461–464, 2003.

[31] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "Performance comparison of quantum-dot cellular automata adders," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, pp. 2522–2526, May 2005.