Journal of Computer and Communications, 2025, 13(10), 187-226
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

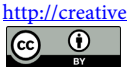
Convolutional Homeomorphic Error-Correcting Codes Utilizing Nano Controlled-Selector Architectural Effectuation

Anas N. Al-Rabadi[1,2]

[1]Department of Computer Engineering, School of Engineering, The University of Jordan, Amman, Jordan
[2]Department of Data Science and Artificial Intelligence, Faculty of Information Technology, Zaytoonah University of Jordan, Amman, Jordan
Email: alrabadi@yahoo.com, anasnaseressa123@gmail.com

How to cite this paper: Al-Rabadi, A.N. (2025) Convolutional Homeomorphic Error-Correcting Codes Utilizing Nano Controlled-Selector Architectural Effectuation. *Journal of Computer and Communications*, **13**, 187-226.
https://doi.org/10.4236/jcc.2025.1310010

Received: September 8, 2025
Accepted: October 25, 2025
Published: October 28, 2025



Abstract

A new nano-based architectural design of multiple-stream convolutional homeomorphic error-control coding will be conducted, and a corresponding hierarchical implementation of important class of the homeomorphic Viterbi algorithm within convolutional homeomorphic error-control coding using lattice networks via nano carbon-based field emission controlled-switching will be achieved. Error-correction coding is highly important in modern wireless networking and digital data transaction since channel coding is required to counteract data errors that are encountered in wireless data networking due to the corresponding existence of unavoidable channel noise. Further, the new lattice nano-based implementation will be useful for enhancing the error-control system performance such as the corresponding enhancements within error-correcting capability, regular synthesis, speed improvement and the minimization of power consumption. Logic homeomorphism describes properties-preserving logic mapping that is intrinsically bijective. Homeomorphic property in error-control coding is important since it is shown that the homeomorphism relationship between multiple-streams of data can be used for further correction of errors that are uncorrectable using the implemented decoding algorithm such as in the case of triple-errors that are uncorrectable using the classical Viterbi algorithm. Applications of the new regular nano-based homeomorphic architecture include low-power design of circuits and systems for enhanced and more reliable wireless data networking and transaction.

Keywords

Error-Control Coding, Homeomorphic Logic, Lattice Networks, Low-Power Computing, Regular Circuits and Systems

1. Introduction

Nano computing will play an increasingly crucial role in building more compact and less power consuming computers in current and future technologies [1]-[19]. Due to this fact, and because several nano-scale computer gates should be homeomorphic such as in nano-scale quantum computing systems [2] [3] [20]-[24], homeomorphic computing will have an increasingly more existence in the future design of regular, compact, and universal circuits and systems. (k, k) homeomorphic circuits are circuits that have the same number of inputs (k) and outputs (k) and are one-to-one mappings between vectors of inputs and outputs, thus the vector of input states can always be uniquely reconstructed from the vector of output states [2] [3] [5] [20]-[28].

Other motivations for pursuing the possibility of implementing circuits and systems using homeomorphic logic would include items such as: 1) *power*: the fact that, theoretically, the internal computations in homeomorphic systems consume no power. It is shown that the amount of energy (heat) dissipated for every non-homeomorphic bit operation is given by $K \cdot T \cdot \ln(2)$, where K is the Boltzmann constant and T is the operating temperature, and that a necessary (but not sufficient) condition for not dissipating power in any physical circuit is that all system circuits must be built using fully homeomorphic logical components. Thus, homeomorphic logic circuits are information-lossless. For this reason, different technologies have been studied to implement homeomorphic logic in hardware such as in bioinformatics, nanotechnology-based circuits and systems, adiabatic CMOS VLSI circuit design, optical systems, and quantum circuits [2] [3] [5] [20]-[26]. Fully homeomorphic digital systems will greatly reduce the power consumption (theoretically eliminate) through three conditions: i) *logical homeomorphism*: the vector of input states can always be uniquely reconstructed from the vector of output states, ii) *physical homeomorphism*: the physical switch operates backwards as well as forwards, and iii) the use of "*ideal-like*" *switches* that have no parasitic resistances; 2) *size*: the current trends related to more dense hardware implementations are heading towards 1 Angstrom (atomic size), at which nano mechanical effects have to be accounted for; and 3) *speed* (*performance*): significant nano-scale computational speed improvements can be expected.

In general, in data communications between two communicating systems (nodes), noise exists and corrupts the sent data messages, and thus noisy corrupted messages will be received. The corrupting noise is usually sourced from the communication channel. Therefore, error correction of communicated data and homeomorphic error correction of communicated batch of data (*i.e.*, parallel data streams) are highly important tasks in situations where noise occurs [20]-[22] [25] [26] [29]-[57]. Many solutions have been classically implemented to solve for the classical error detection and correction problems: 1) one solution to solve for error-control is *parity checking* [43], which is one of the most widely used methods for error detection in digital logic circuits and systems, in which re-sending data is

performed in case error is detected in the transmitted data. This error is detected by the parity checker in the receiver side. Various parity-preserving circuits have been implemented in which the parity of the outputs matches that of the inputs, and such circuits can be fault-tolerant since a circuit output can detect a single error; 2) another solution to solve this highly important problem, that is to extract the correct data message from the noisy erroneous counterpart, is by using various coding schemes that work optimally for specific types of statistical distributions of noise [29] [35] [38] [45] [47]-[51] [53] [56] [57].

The main contribution of this paper is the introduction of new nano-based implementation of convolution-based error-control coding that applies the homeomorphism property in both the convolution-based encoder for multiple-stream error-control encoding and in the new homeomorphic Viterbi decoding algorithm for multiple-stream error-control decoding. Figure 1 shows the introduced new system hierarchy implementation in this article, where the first bottom level represents the applied mathematics of Galois algebra which will be used in the upper levels, the second middle level represents the used applied physics which is comprised of three sub-levels of field-emission physics and carbon-based field-emission devices and field-emission circuits, and the third upper level represents the implemented computation which is comprised of two sub-levels of logic function implementation using regular two-dimensional lattice networks and system implementation of the homeomorphic Viterbi algorithm.

Homeomorphic Viterbi Implementation
Lattice Network Realizations
Field Emission-Based Circuits
Carbon-Based Field Emission Devices
Field-Emission Physics
Galois Algebra

Figure 1. The introduced and implemented system design hierarchy.

This article is organized as follows: Basic background in error-control coding, homeomorphic logic and homeomorphic error-control coding is presented in Section 2. Fundamentals of lattice networks are presented in Section 3. Basics of computing using carbon nano devices are presented in Section 4. The new nano circuit design of the homeomorphic Viterbi algorithm is introduced in Section 5. Conclusions and future work are presented in Section 6.

2. Homeomorphic Error Correction

This Section presents basic background in the topics of error-correction coding, homeomorphic logic and homeomorphic error-control coding. The fundamentals presented in this section will be utilized in the development of the new results

which will be introduced in Section 5.

2.1. Error-Control Coding

In data communication, noise usually exists and is generated from the channel in which transmitted data is communicated. Such noise corrupts sent messages from one end and thus noisy corrupted messages are received on the other end. To solve the problem of extracting a correct message from its corrupted counterpart, noise must be modeled and accordingly an appropriate encoding/decoding communication schemes must be implemented. For this reason, various coding schemes and techniques have been proposed and one very important utilized family is the method of convolutional codes [20]-[22] [25] [26] [29]-[57].

Each of the two nodes' sides in a networked system consists of three major parts: 1) encoding (e.g., generating a convolutional code using a convolutional encoder) to generate an encoded transmitted message, 2) channel noise, and 3) decoding (e.g., *generating the correct convolution code using the corresponding decoding algorithm such as the Viterbi algorithm for the case of Additive White Guassian Noise* ($AWGN$)) to generate the decoded correct received data message. In general, in block coding, the encoder receives a k-bit message block and generates an n-bit code word, and therefore code words are generated on a block-by-block basis, and the whole message block must be buffered before the generation of the associated code word. On the other hand, message bits are received serially rather than in blocks where it is undesirable to use a buffer. In such case, one uses convloutional coding, in which a convolutional coder generates redundant bits by using modulo-2 convolutions. The binary convolutional encoder can be seen as a finite state machine consisting of an M-stage shift register with interconnections to n modulo-2 adders and a multiplexer to serialize the outputs of the adders, in which an L-bit message sequence generates a coded output sequence of length $n(L + M)$ bits.

Definition 1. For an L-bit message sequence, M-stage shift register, n modulo-2 adders, and a generated coded output sequence of length $n(L + M)$ bits, the code rate r is calculated as $r = \frac{L}{n(L+M)}$ bits/symbol, where for the typical case of $L \gg M$ the code rate reduces to $r \approx \frac{1}{n}$ bits/symbol.

Definition 2. The constraint length of a convolutional code is the number of shifts over which a single message bit can influence the encoder output. Thus, for an encoder with an M-stage shift register, the number of shifts required for a message bit to enter the shift register and then come out of it is equal to $K = M + 1$. Thus, the encoder constraint length is equal to K.

A binary convolutional code can be generated with code rate $r = \frac{k}{n}$ by using k shift registers, an n modulo-2 adders, an input multiplexer, and an output multiplexer. An example of a convolutional encoder with constraint length = 3 and rate = ½ is the one shown in **Figure 2**.

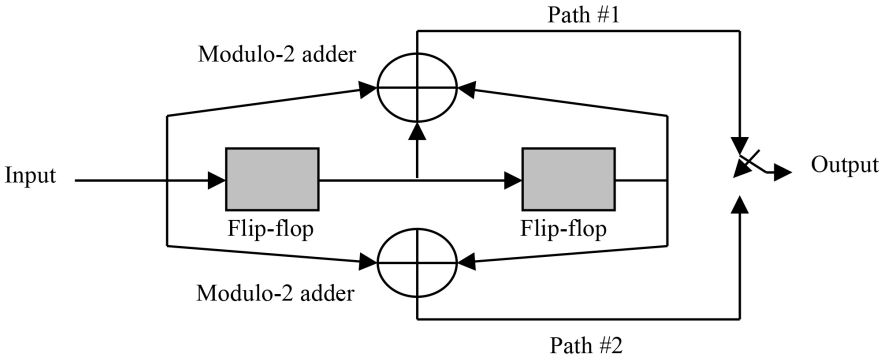


Figure 2. Convolutional encoder with constraint length = 3 and rate = ½. The flip-flop is a unit-delay element, and the modulo-2 adder is the logic XOR operation.

The convolutional codes generated by the encoder in **Figure 2** are part of what is generally called *nonsystematic codes*. Each path connecting the output to the input of a convolutional encoder can be characterized in terms of the impulse response which is defined as the response of that path to "1" applied to its input, with each flip-flop of the encoder set initially to "0". Equivalently, we can characterize each path in terms of a generator polynomial defined as the unit-delay transform of the impulse response. More specifically, the generator polynomial is defined as:

$$g(D)=\sum_{i=0}^{M} g_i D^i \tag{1}$$

where g_i is the generator coefficients $\in \{0, 1\}$, and the generator sequence $\{g_0, g_1, \ldots, g_M\}$ composed of generator coefficients is the impulse response of the corresponding path in the convolutional encoder, and D is the unit-delay variable.

Example 1. For the convolutional encoder in **Figure 2**, path #1 impulse response is (1, 1, 1), and path #2 impulse response is (1, 0, 1). Thus, according to Equation (1), the following are the corresponding generating polynomials, respectively, where addition is performed in modulo-2 addition arithmetic:

$$g_1(D)=1\cdot D^0+1\cdot D^1+1\cdot D^2=1+D+D^2$$

$$g_2(D)=1\cdot D^0+0\cdot D^1+1\cdot D^2=1+D^2$$

For a message sequence (10011), the following is the D-domain polynomial representation:

$$m(D)=1\cdot D^0+0\cdot D^1+0\cdot D^2+1\cdot D^3+1\cdot D^4=1+D^3+D^4$$

As convolution in time domain is transformed into multiplication in the D-domain, path #1 output polynomial and path #2 output polynomial are as follows, respectively:

$$c_1(D)=g_1(D)m(D)=\left(1+D+D^2\right)\left(1+D^3+D^4\right)$$
$$=1+D+D^2+D^3+D^6$$

$$c_2(D) = g_2(D)m(D) = (1+D^2)(1+D^3+D^4)$$
$$= 1+D^2+D^3+D^4+D^5+D^6$$

Therefore, the output sequences of paths #1 and #2 are as follows, respectively:

Output sequence of path #1: (1111001)

Output sequence of path #2: (1011111)

The resulting encoded sequence from the convolutional encoder in Figure 2 is obtained by multiplexing the two output sequences of paths #1 and #2 as follows:

$$c = (11,10,11,11,01,01,11)$$

Example 2. For the convolutional encoder in Figure 2, the following are examples of encoded data messages:

$$m_1 = (11011) \to c_1 = (11010100010111),$$
$$m_2 = (00011) \to c_2 = (00000011010111),$$
$$m_3 = (01001) \to c_3 = (00111011111011)$$

In general, a data message sequence of length L bits results in an encoded sequence of length equals to $n(L+K-1)$ bits. Usually a terminating sequence of $(K-1)$ zeros called the tail of the message is appended to the last input bit of the message sequence in order for the shift register to be restored to its zero initial state. The structural properties of the convolutional encoder (cf. Figure 2) can be represented graphically in several equivalent representations (cf. Figure 3) using 1) code tree, 2) trellis and 3) state diagram. The trellis contains $(L+K)$ levels where L is the length of the incoming message sequence and K is the constraint length of the code. Therefore, the trellis form is preferred over the code tree form because the number of nodes at any level of the trellis does not continue to grow as the number of incoming message bits increases, but rather it remains constant at 2^{K-1}, where K is the constraint length of the code. Figure 3 shows the various graphical representations for the convolutional encoder in Figure 2.

Therefore, any encoded output sequence can be generated from the corresponding input message sequence using the following equivalent methods: 1) circuit of the convolutional encoder (cf. Figure 2), 2) polynomial generator (cf. Examples 1 and 2), 3) code tree (cf. Figure 3(a)), 4) trellis (cf. Figure 3(b)), and 5) state diagram (cf. Figure 3(c)). An important decoder that uses the trellis representation to correct received erroneous messages is the Viterbi decoding algorithm [39] [56] [57]. The Viterbi algorithm is a dynamic programming algorithm which is used to find the maximum-likelihood sequence of hidden states, which results in a sequence of observed events particularly in the context of hidden Markov models. The Viterbi algorithm forms a subset of information theory, and has been extensively used in a wide range of applications including speech recognition, keyword spotting, computational linguistics, bioinformatics, and in communications including digital cellular, dial-up modems, satellite, deep-space and wireless local area networks.

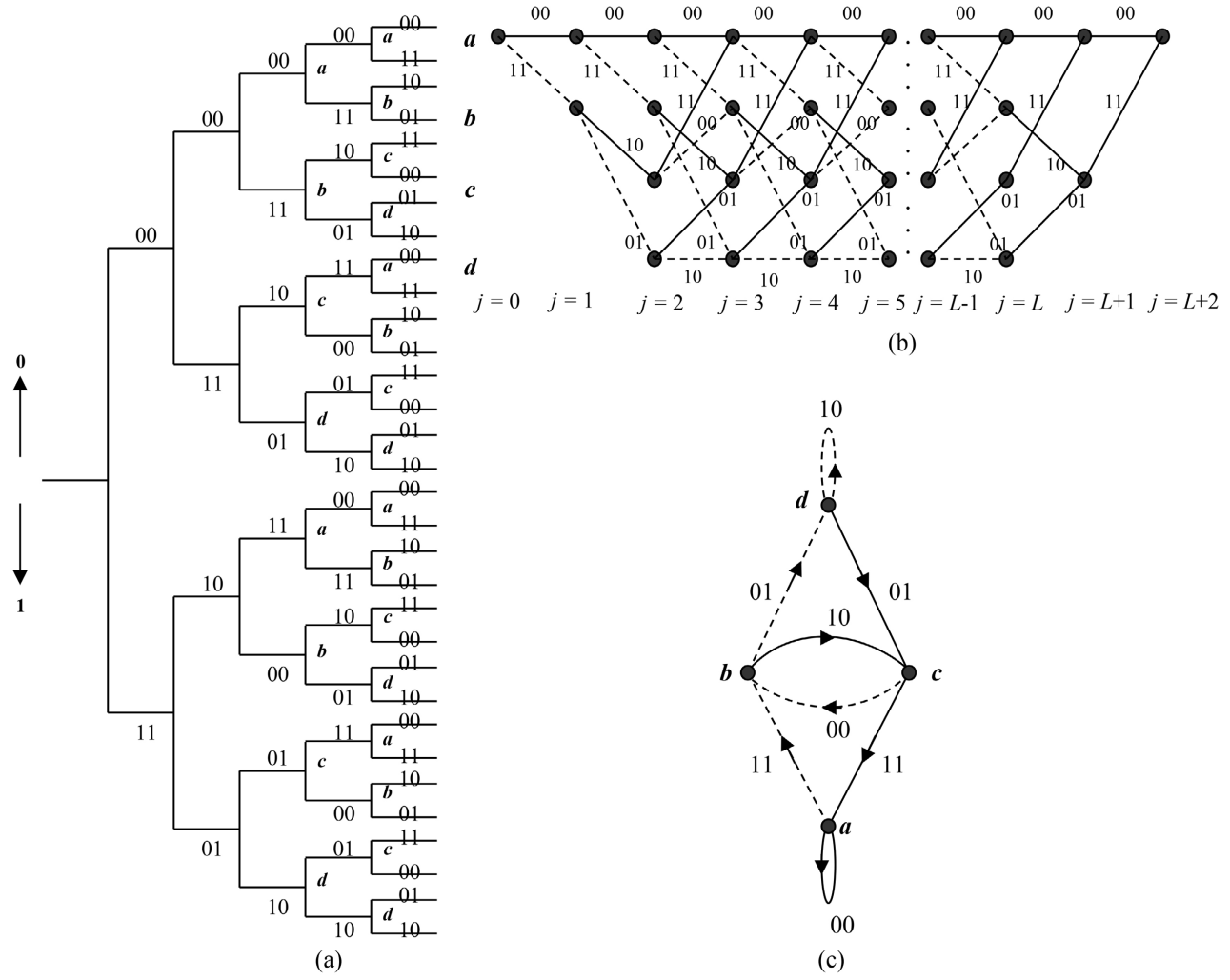


Figure 3. Representations for the circuit of the convolutional encoder in **Figure 2**: (a) code tree, (b) trellis and (c) state diagram. Solid line is input of value "0" and the dashed line is input of value "1". The binary label on each branch is the encoder's output as it moves from one state to another. The state encoding of the states can be as {$a = 00$, $b = 10$, $c = 01$, $d = 11$}.

The Viterbi algorithm is a maximum-likelihood decoder which is optimum for a noise type which is statistically characterized as an Additive White Gaussian Noise (*AWGN*). This algorithm operates by computing a metric for every possible path in the trellis representation. The metric for a specific path is computed as the Hamming distance between the coded sequence represented by that path and the received sequence. For a pair of code vectors c_1 and c_2 that have the same number of elements, the Hamming distance $d(c_1, c_2)$ between such a pair of code vectors is defined as the number of locations in which their respective elements differ. In the Viterbi algorithm context, the Hamming distance is computed by counting how many bits are different between the received channel symbol pair and the possible channel symbol pairs, in which the results can only be "0", "1" or "2". Therefore, for each node (*i.e.*, state) in the trellis the Viterbi algorithm compares the two paths entering the node. The path with the lower metric is retained and the other path is discarded. This computation is repeated for every level j of the

trellis in the range $M \le j \le L$, where $M = K - 1$ is the encoder's memory and L is the length of the incoming message sequence. The paths that are retained are called survivor or active paths. In some cases, applying the Viterbi algorithm leads to the following difficulty: when the paths entering a node (state) are compared and their metrics are found to be identical then a choice is made by making a guess (flipping a fair coin). The Viterbi algorithm is a maximum likelihood sequence estimator, where the following procedure and Examples 3 - 4 illustrate the detailed steps for applying Algorithm 1.

Algorithm 1. Viterbi.

1. *Initialization step*: Label the left-most state of the trellis (i.e., all zero state at level 0) as 0
2. *Computation step*: Let $j = 0, 1, 2, \ldots$, and assume at the previous j the following is performed:
   a. All survivor paths are identified
   b. The survivor paths and its metric for each state of the trellis are stored
   Then, at level (clock time) $(j+1)$ and **for** all the paths entering each state of the trellis, compute the metric by adding the metric of the incoming branches to the metric of the connecting survivor path from level j. Thus, for each state, identify the path with the lowest metric as the survivor of step $(j + 1)$, therefore updating the computation
3. *Final step*: **Continue** the computation until the algorithm completes the forward search through the trellis and thus reaches the terminating node (i.e., all zero state), at which time it makes a decision on the maximum-likelihood path. **Then**, the sequence of symbols associated with that path is released to the destination as the decoded version of the received sequence

Example 3. Suppose that the resulting encoded sequence from the convolutional encoder in Figure 2 is as follows:

$$c = (0000000000)$$

Now suppose a noise corrupts this sequence, and the noisy received sequence is as follows:

$$c' = (0100010000)$$

Using the Viterbi algorithm, Figure 4 shows the resulting step-by-step illustration to produce the survivor path which generates the correct sent message c = (0000000000).

Example 4. For the convolutional encoder in Figure 2, path #1 impulse response is (1, 1, 1), and path #2 impulse response is (1, 0, 1). Thus, the following are the corresponding generating polynomials, respectively:

$$g_1(D) = 1 \cdot D^0 + 1 \cdot D^1 + 1 \cdot D^2 = 1 + D + D^2$$

$$g_2(D) = 1 \cdot D^0 + 0 \cdot D^1 + 1 \cdot D^2 = 1 + D^2$$

For a message sequence (101), the following is the D-domain polynomial representation:

$$m(D) = 1 \cdot D^0 + 0 \cdot D^1 + 1 \cdot D^2 = 1 + D^2$$

As convolution in time domain is transformed into multiplication in the D-domain, the path #1 output polynomial and path #2 output polynomial are as follows, respectively, where addition is performed in modulo-2 arithmetic:

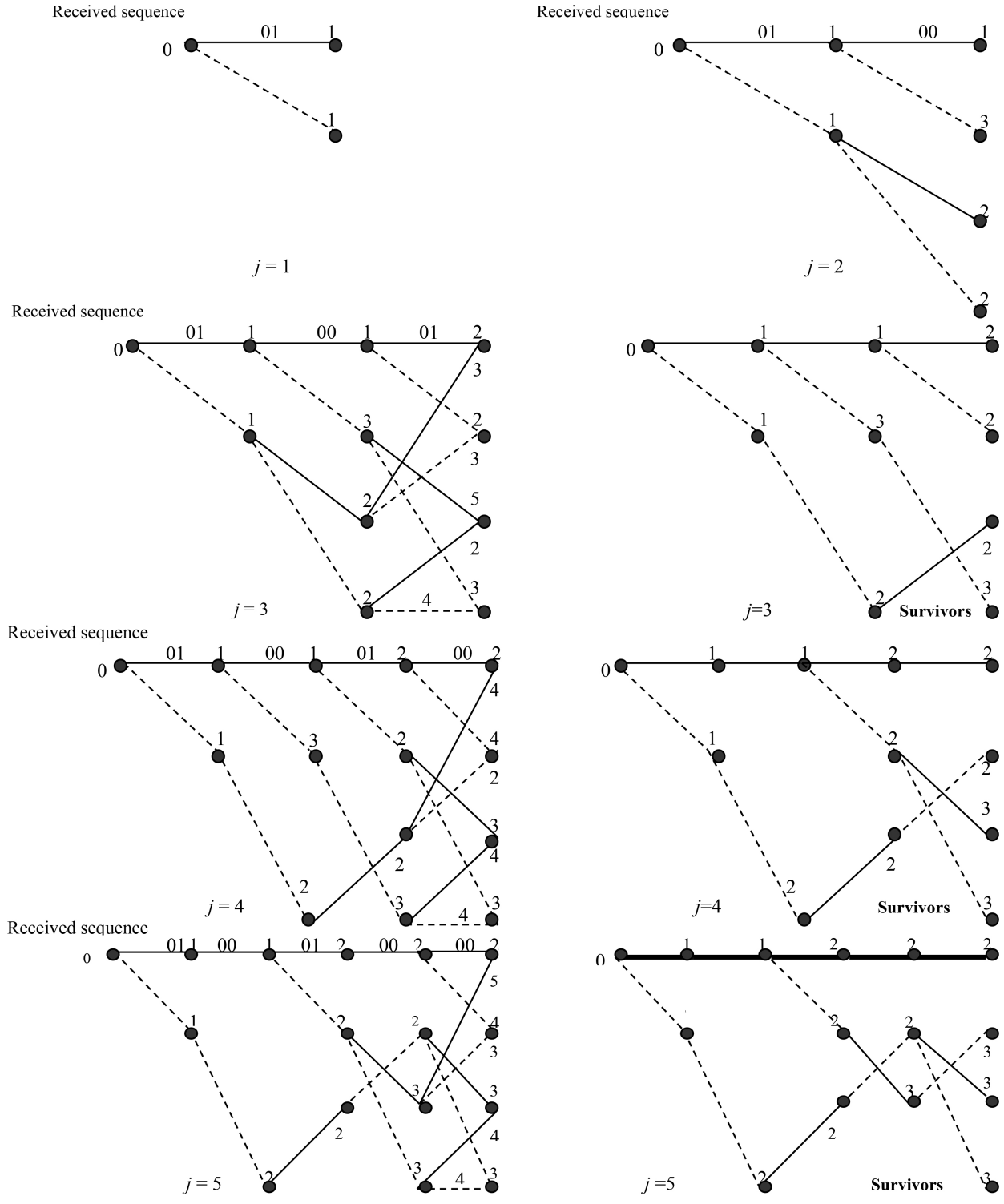


Figure 4. The illustration of the steps of the Viterbi algorithm when applied for Example 3, where the bold path (in level $j = 5$) is the survivor path.

$$c_1(D) = g_1(D)m(D) = \left(1 + D + D^2\right)\left(1 + D^2\right) = 1 + D + D^3 + D^4$$

$$c_2(D) = g_2(D)m(D) = \left(1 + D^2\right)\left(1 + D^2\right) = 1 + D^4$$

Therefore, the output sequences of paths #1 and #2 are as follows, respectively:

Output sequence of path #1: (11011)

Output sequence of path #2: (10001)

The resulting encoded sequence from the convolutional encoder in Figure 2 is obtained by multiplexing the two output sequences of paths #1 and #2 as follows:

$$c = (11,10,00,10,11)$$

Now suppose a noise corrupts this sequence, and the noisy received sequence is as follows:

$$c' = (01,10,10,10,11)$$

Using the Viterbi algorithm, Figure 5 shows the resulting survivor path which generates the correct sent message:

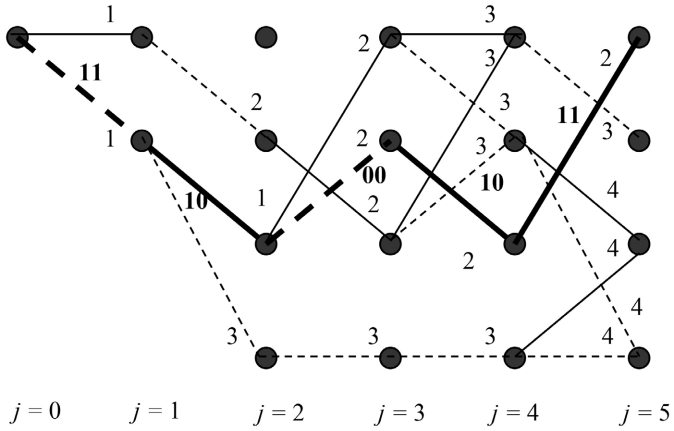


Figure 5. The resulting survivors of the Viterbi algorithm for Example 4 where the bold path is the survivor path.

$$c = (11,10,00,10,11).$$

A difficulty with the application of the Viterbi algorithm occurs when the received sequence is very long. In this case the Viterbi algorithm is applied to a truncated path memory using a decoding window of length greater or equal five times the convolutional code constraint length K, in which the algorithm operates on a frame-by-frame of the received sequence each of length $l \geq 5K$. The decoding decisions made in this way are not a truly maximum likelihood, but they can be made almost as good provided that the decoding window is long enough. Another difficulty is the number of errors; for example, in case of three errors, the Viterbi algorithm when applied to a convolutional code of $r = ½$ and $K = 3$ cannot produce a correctable decoded message from the incoming erroneous message. Exceptions are triple-error patterns that spread over a time span > K.

2.2. Homeomorphic Digital Logic

In general, an (n, k) homeomorphic circuit is a circuit that has n number of inputs and k number of outputs and is one-to-one mapping between vectors of inputs

and outputs, thus the vector of input states can be always uniquely reconstructed from the vector of output states [2] [23] [24] [28]. Thus, a (k, k) homeomorphic map is a bijective function which is both injective (one-to-one) and surjective (onto). (Such bijective systems are also known as equipollent, equipotent, and one-to-one correspondence.) The auxiliary outputs that are needed only for the purpose of homeomorphism are called "garbage" outputs. These are auxiliary outputs from which a homeomorphic map is constructed (cf. Example 5). Therefore, ***homeomorphic systems are information-lossless***.

Geometrically, achieving homeomorphism leads to value space-partitioning that leads to spatial partitions of unique values. Algebraically and in terms of systems representation, homeomorphism leads to multi-input multi-output (MIMO) bijective maps (*i.e.*, bijective functions). An algorithm called homeomorphic Boolean function (HomBF) that produces a homeomorphic form from a non-homeomorphic Boolean function is as follows [2] (**Algorithm 2**).

Algorithm 2. HomBF.

1. To achieve (k, k) homeomorphism, add sufficient number of auxiliary output variables such that the number of outputs equals the number of inputs. Allocate a new column in the mapping table for each auxiliary variable
2. For construction of the first auxiliary output, assign a constant C_1 to half of the cells in the corresponding table column (e.g., zeros), and the second half as another constant C_2 (e.g., ones). For convenience, one may assign C_1 to the first half of the column, and C_2 to the second half of the column (cf. **Table 1(a)**, column W_1)
3. For the next auxiliary output, **If** non-homeomorphism still exists, **Then** assign for *identical* output tuples (non-homeomorphic map entries) values which are half zeros and half ones, and then assign a constant for the remainder that are already homeomorphic
4. **Do** step 3 until all map entries are homeomorphic

Example 5. The standard two-variable Boolean equivalence (XNOR): $W = c \otimes d$ is non-homeomorphic. The following table lists the mapping components:

c	d	W
0	0	1
0	1	0
1	0	0
1	1	1

Applying the above HomBF algorithm, **Table 1** presents four possible homeomorphic two-variable Boolean maps for the XNOR function:

For example, using the HomBF algorithm, the construction of the homeomorphic map in (a) of **Table 1** is obtained as follows: since W is non-homeomorphic, assign auxiliary output W_1 and assign the first half of its values the constant "0" and the second half another constant "1". The new XNOR map is now homeomorphic. This gate is also called the inverted Controlled-NOT (inverted C-NOT) gate) in which: $W_1 = c$ and $W = c \otimes d = (c \oplus d)'$.

Table 1. Four possible (2, 2) homeomorphic maps for the Boolean XNOR (Boolean equivalence).

(a)			
c	***d***	**W**	**W₁**
0	0	1	**0**
0	1	0	**0**
1	0	0	**1**
1	1	1	**1**
(b)			
c	***d***	**W**	**W₁**
0	0	1	**1**
0	1	0	**1**
1	0	0	**0**
1	1	1	**0**
(c)			
C	***d***	**W**	**W₁**
0	0	1	**0**
0	1	0	**1**
1	0	0	**0**
1	1	1	**1**
(d)			
c	***d***	**W**	**W₁**
0	0	1	**1**
0	1	0	**0**
1	0	0	**1**
1	1	1	**0**

2.3. Error Correction via the Homeomorphic Viterbi Algorithm

While in Subsection 2.1 the error correction of communicated data was done for the case of single-input single-output (SISO) systems, this section presents homeomorphic error correction of communicated batch (parallel) of data in multiple-input multiple-output (MIMO) systems. Homeomorphism in parallel-based data communication is directly observed since:

$$\vec{O}_1 = \vec{I}_2 \tag{2}$$

where $\vec{O}_1$ is the *unique* output transmitted data from node #1 and $\vec{I}_2$ is the *unique* input received data to node #2.

In MIMO systems, the existence of noise will cause an error that may lead to

non-homeomorphism in data communication (*i.e.*, non-homeomorphism in data mapping) since $\vec{O}_1 \neq \vec{I}_2$. The implementation of homeomorphic error correction can be performed: 1) in software using the homeomorphic error-correction algorithm and 2) in hardware using nano error correction hardware. The following algorithm, which is called Homeomorphic Viterbi (HV) Algorithm, presents the implementation of homeomorphic error correction [20]-[22] [25] [26] (**Algorithm 3**).

Algorithm 3. HV.

1. Use the HomBF Algorithm to reversibly encode the communicated batch of data
2. Given a specific convolutional encoder circuit, determine the generator polynomials for all paths
3. **For** each communicated message within the batch, determine the encoded message sequence
4. **For** each received message, use the Viterbi Algorithm to decode the received erroneous message
5. Generate the total maximum-likelihood trellis resulting from the iterative application of the Viterbi algorithm
6. Generate the corrected communicated batch of data messages
7. **End**

The convolutional encoding for the HV algorithm can be performed *serially* using a single convolutional encoder from **Figure 2**, or in *parallel* using the general parallel convolutional encoder circuit shown in **Figure 6** in which several s convolutional encoders operate in parallel for encoding s number of simultaneously submitted messages (*i.e.*, data message set of cardinality (size) equal to s) generated from s nodes.

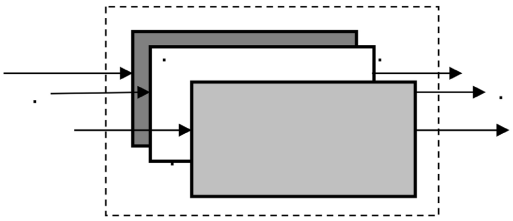


Figure 6. General MIMO encoder circuit for the parallel generation of convolutional codes where each box represents a single SISO convolutional encoder such as the one shown in **Figure 2**.

Example 6. The homeomorphism implementation (e.g., HomBF Algorithm) upon the following input bit stream $\{m_1 = 1, m_2 = 1, m_3 = 1\}$ produces the following homeomorphic set of message sequences:

$$m_1 = (101),\ m_2 = (001),\ m_3 = (011)$$

For the convolutional encoder in **Figure 6**, the following is the D-domain polynomial representations, respectively:

$$m_1(D) = 1 \cdot D^0 + 0 \cdot D^1 + 1 \cdot D^2 = 1 + D^2$$

$$m_2(D) = 0 \cdot D^0 + 0 \cdot D^1 + 1 \cdot D^2 = D^2$$

$$m_3(D) = 0 \cdot D^0 + 1 \cdot D^1 + 1 \cdot D^2 = D + D^2$$

The resulting encoded sequences are generated in parallel as follows, respec-

tively:

$$c_1 = (1110001011)$$
$$c_2 = (0000111011)$$
$$c_3 = (0011010111)$$

Now suppose noise sources corrupt these sequences, and the noisy received sequences are as follows:

$$c_1' = (1111001001),\ c_2' = (0100101011),\ c_3' = (0010011111)$$

Using the HV algorithm, Figure 7 shows the resulting survivor paths which generate the correct sent messages:

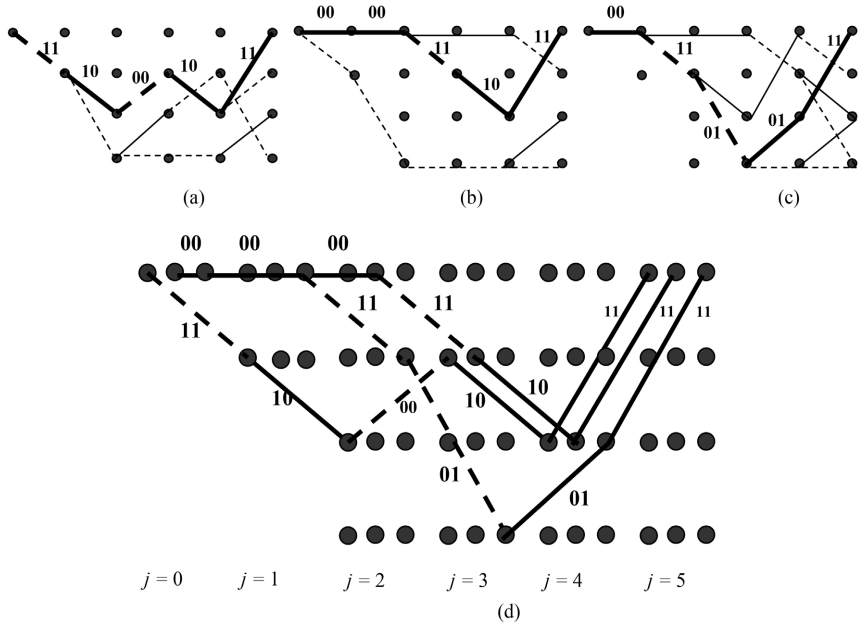


Figure 7. The resulting survivor paths of the HV algorithm when applied to Example 6.

$$\{c_1 = (1110001011), c_2 = (0000111011), c_3 = (0011010111)\}.$$

As in the non-homeomorphic Viterbi Algorithm, in some cases, applying the homeomorphic Viterbi (HV) algorithm leads to the following difficulties: 1) when the paths entering a node (state) are compared and their metrics are found to be identical then a choice is made by making a guess (*i.e.*, flipping a fair coin); 2) when the received sequence is very long and in this case the homeomorphic Viterbi algorithm is applied to a truncated path memory using a decoding window of length greater or equal five times the convolutional code constraint length K, in which the algorithm operates on a frame-by-frame of the received sequence each of length $l \geq 5K$, and the decoding decisions made in this way are not a truly maximum likelihood, but they can be made almost as good provided that the decoding window is long enough; 3) the number of errors: for example, in case of three errors, the Viterbi algorithm when applied to a convolutional code of $r = ½$ and $K = 3$ cannot produce a correctable decoded message from the incoming er-

roneous noisy (corrupted) message. (Exceptions are triple-error patterns that spread over a time span > K.)

Yet, parallelism in multi-stream data submission (transmission) allows for the possible existence of extra relationship(s) between the submitted data-streams that can be used for: 1) detection of error existence and 2) further correction after HV algorithm in case the HV algorithm fails to correct for the occurring errors. Examples of such inter-stream relationships are: 1) parity (even and odd) relationship between the corresponding bits within the inter-stream submitted data, 2) homeomorphism relationship between the parallel submitted data streams and this relationship exists from applying a known homeomorphic mapping such as the HomBF algorithm, or 3) combination of parity and homeomorphism properties. The homeomorphism property in the HV algorithm produces a homeomorphism relationship between the sent parallel streams of data, and this known homeomorphism mapping can be used to correct the uncorrectable errors (e.g., triple errors) which the HV algorithm fails to correct.

Example 7. The following is a version of the HomBF algorithm that produces homeomorphism as follows (Algorithm 4).

Algorithm 4. HomBF (Version 1).

1. To achieve (k, k) homeomorphism, add sufficient number of auxiliary output variables (starting from right to left) such that the number of outputs equals the number of inputs. Allocate a new column in the mapping's table for each auxiliary variable
2. For construction of the first auxiliary output, assign a constant C_1 = "0" to half of the cells in the corresponding table column, and the second half as another constant C_2 = "1". Assign C_1 to the first half of the column, and C_2 to the second half of the column
3. For the next auxiliary output, **If** non-homeomorphism still exists, **Then** assign for *identical* output tuples (non-homeomorphic map entries) values which are half ones and half zeros, and then assign a constant for the remainder that are already homeomorphic which is the one's complement (NOT; inversion) of the previously assigned constant to that remainder
4. **Do** step 3 until all map entries are homeomorphic

For the parallel sent bit stream {1, 1, 1} in Example 6 in which the homeomorphism implementation (using Version 1 of the HomBF Algorithm) produces the following homeomorphic sent set of data sequences: {m_1 = (101), m_2 = (001), m_3 = (011)}. Suppose that m_1 and m_2 are decoded correctly and m_3 is still erroneous due to submission. Figure 8 shows possible tables in which erroneous m_3 exist.

m_1	1	0	1
m_2	0	0	1
m_3	0	1	1

(a)

m_1	1	0	1
m_2	0	0	1
m_3	0	**0**	1

(b)

m_1	1	0	1
m_2	0	0	1
m_3	**1**	1	1

(c)

m_1	1	0	1
m_2	0	0	1
m_3	**1**	**0**	1

(d)

m_1	1	0	1
m_2	0	0	1
m_3	0	**1**	**0**

(e)

m_1	1	0	1
m_2	0	0	1
m_3	**1**	**0**	**0**

(f)

m_1	1	0	1
m_2	0	0	1
m_3	**1**	1	**0**

(g)

m_1	1	0	1
m_2	0	0	1
m_3	0	**0**	**0**

(h)

Figure 8. Tables for possible errors in data stream m_3 that is generated by the HomBF Algorithm V1: (a) original sent correct m_3 that resulted from the application of the HomBF Algorithm V1, and (b)-(h) possibilities of the erroneous received m_3.

Note that the erroneous m_3 is Figures 8(b)-(e) and Figure 8(g) and Figure 8(h) are correctable using the HV algorithm since less than triple-errors exits, but the triple error as in Figure 8(f) is (usually) uncorrectable using the HV algorithm. ***Yet, the existence of the homeomorphism property using the HomBF algorithm adds information that can be used to correct m_3 as follows***: By applying the HomBF Algorithm (Version 1) from right-to-left in Figure 8(f) one notes that in the second column (from right) two "0" cells are added in the top in the correctly received m_1 and m_2 messages, which means that in the most right column the last cell must be "1" since otherwise the top two cells in the correctly received m_1 and m_2 messages should have been "0" and "1" respectively to achieve value space-partitioning. Now, since the 3rd cell of the most right column must be "1" then the last cell of the 2nd column from the right must be "1" also because of the uniqueness requirement according to the HomBF algorithm (Version 1) for value space-partitioning between the first two messages $\{m_1, m_2\}$ and the 3rd message m_3. Then, and according to the HomBF algorithm (Version 1) the 3rd cell of the last column from right must have the value "0" which is the one's complement (NOT) of the previously assigned constant "1" to the 3rd cell of the 2nd column from the right. *Consequently, the correct message $m_3 = (011)$ is obtained.*

3. Lattice Networks

It is well-known in logic synthesis that certain classes of logic functions exhibit specific types of symmetries [2] [58] [59]. One method to characterize a symmetry that might exist in a logic function is performed by using symmetry indices.

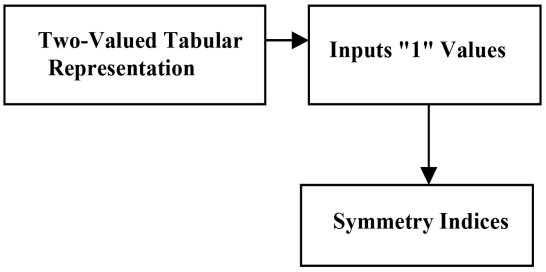
Definition 3. A single index symmetric function, denoted as $S^k(x_1, x_2, \ldots, x_n)$ has value 1 when exactly k of its n inputs are equal to 1, and exactly $(n - k)$ of its remaining inputs are 0.

Example 8. The following Karnaugh map represents symmetric function $F = ab \oplus bc \oplus ac$.

In Figure 9, a symmetry index S^i specifies a Karnaugh map cell that counts value "1" in the specified minterm in number of times equal to i.

AB / C	0	1
00	0 S0	0 S1
01	0 S1	1 S2
11	1 S2	1 S3
10	0 S1	1 S2

(a)



(b)

Figure 9. A three-variable symmetric Boolean function: (a) Karnaugh map, and (b) relation with symmetry indices.

Definition 4. The elementary symmetric functions of n variables are:

$$S^0 = \overline{x}_1\overline{x}_2\cdots\overline{x}_n,$$

$$S^1 = x_1\overline{x}_2\cdots\overline{x}_n + \overline{x}_1x_2\overline{x}_3\cdots\overline{x}_n + \cdots + \overline{x}_1\overline{x}_2\cdots\overline{x}_{n-1}x_n,\ \cdots,\ S^n = x_1x_2\cdots x_n.$$

Thus, for Boolean function of three variables one obtains the following sets of symmetry indices: $S^0 = \{\overline{a}\overline{b}\overline{c}\}$, $S^1 = \{\overline{a}\overline{b}c, \overline{a}b\overline{c}, a\overline{b}\overline{c}\}$, $S^2 = \{\overline{a}bc, ab\overline{c}, a\overline{b}c\}$, and $S^3 = \{abc\}$. It has been shown that an arbitrary n-variable symmetric function f is uniquely represented by elementary symmetric functions $\{S^0, S^1, \cdots, S^n\}$ as follows: $f = \sum_{i\in A} S^i = S^A$, where $A \in \{0,1,\cdots,n\}$. Also it can be shown that for $f = S^A$ and $g = S^B$, the following operations are obtained as in Equations (3)-(6):

$$f.g = S^{A\cap B} \tag{3}$$

$$f + g = S^{A\cup B} \tag{4}$$

$$f \oplus g = S^{A\oplus B} \tag{5}$$

$$\overline{f} = S^{\overline{A}} \tag{6}$$

It has been shown [2] that a function which is not symmetric can be symmetrized by repeating its variables. This method of variable repetition transforms the values of Karnaugh map cells which make the function non-symmetric into don't cares which make the function symmetric.

Example 9. The following Karnaugh map demonstrates the symmetrization by repeating the variables of non-symmetric Boolean function $F = \overline{a} + b$.

One notes that while in **Figure 10(a)** conflicting values occur for symmetry index S^1 in minterms $\overline{a}b$ and $a\overline{b}$ and thus producing non-symmetric function, non-conflicting values are produced for the same non-symmetric function in **Figure 10(b)** by repeating variable $\{a\}$ two times. As stated previously, various applications of symmetry indices for the synthesis of logic functions have been previously shown. This includes symmetric networks, Akers arrays [58] and lattice networks [2] [6] [7] [60]-[62] among other several implementations. The concept of lattice networks for switching functions involves three components: 1) expansion of a function that corresponds to the root in the lattice which creates several successor nodes of the expanded node, 2) joining of several nodes to a single node, and 3) regular geometry to which the nodes are mapped. **Figure 11(a)** shows an example of a four-variable (*i.e.*, four-level) lattice network and **Figure 11(b)** and **Figure 11(c)** show the relationship between **Figure 11(a)** and symmetry indices. Therefore, **Figure 11** shows an example of the fact that regular lattice networks exhibit a close relationship with symmetry indices, where symmetry indices represent the sets of all possible paths from leafs to the root of the corresponding lattice network. One also notes that each internal node in **Figure 11(a)** is a Shannon node which is a two-to-one multiplexer whose output goes in two directions. Other types of expansion nodes can be implemented as well that will produce corresponding types of various kinds of lattice networks [2].

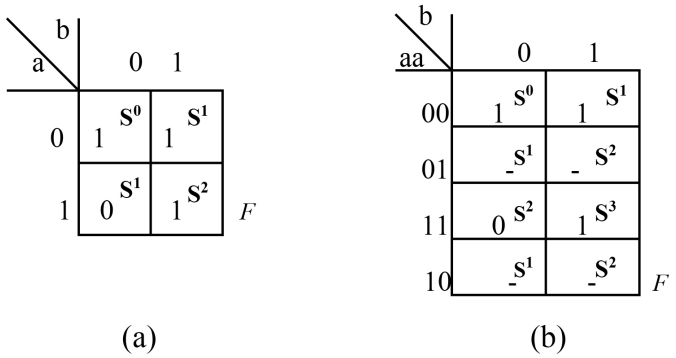


Figure 10. Symmetrization by repeating variables: (a) non-symmetric Boolean function *F*, and (b) symmetric Boolean function *F* which is obtained by the repetition of variable {*a*}.

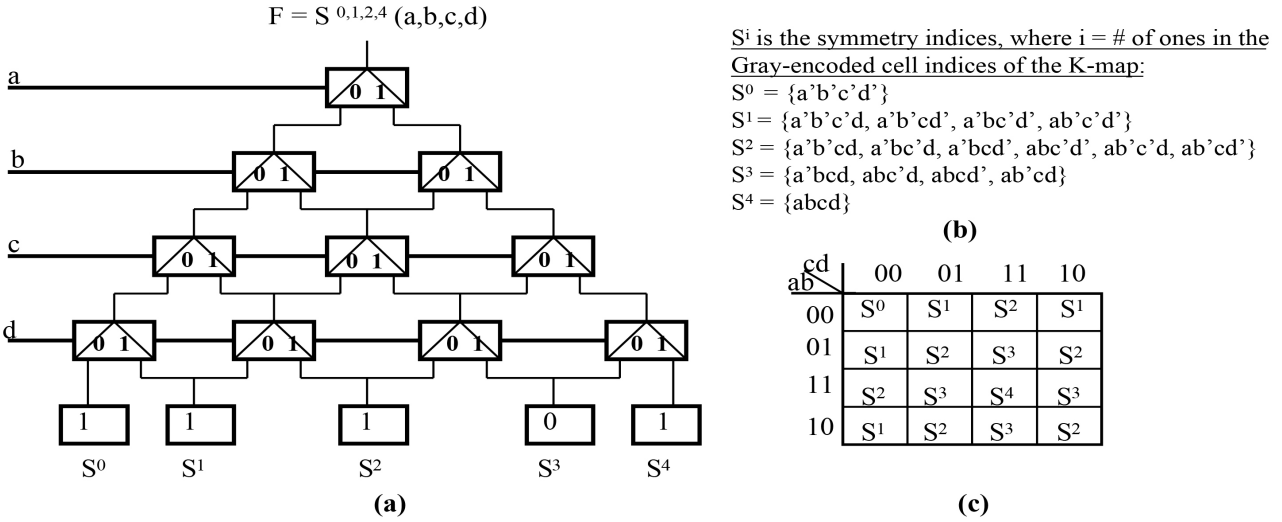


Figure 11. The relation between a lattice and symmetry indices: (a) lattice network for a four-variable function, (b) sets of binary symmetry indices, and (c) the corresponding Karnaugh map interpretation of the binary symmetry indices.

It has been shown that every non-symmetric function can be symmetrized by repeating its variables [2]. Therefore, since a single variable corresponds to a single level in the lattice network, repeating variables produces a repetition of levels in a lattice network. In general, three main factors control the size of a lattice network that realizes non-symmetric functions: 1) expansion types that are used in the internal nodes, 2) order of variables upon which functions are expanded in each level of the lattice, and 3) the choice of repeated variables. Consequently, various optimization methods have been addressed for an optimal choice of the three factors that are mentioned above in order to minimize the size of the corresponding lattice network.

The realization of non-symmetric functions using lattice networks demands the repetition of variables [2]. In many cases, one has to repeat variables so many times that will result in a big size two-dimensional lattice network which doesn't fit the specified area. On the other hand, one can re-route the corresponding interconnects between the internal nodes of the lattice network using optimization

methods in a way such that the network will fit into the specified layout space. Yet, this process will make the used interconnects between lattice cells of many different lengths, and consequently strips the lattice network from one of its most important features for which all of the interconnects are of the same length.

The binary hierarchy of families of canonical forms [2] [63] and the corresponding decision diagrams [2] [64] are based on three basic functional expansions: Shannon, positive Davio and negative Davio expansions [2] [59] [65], which are given below respectively:

$$f(x_1, x_2, \cdots, x_n) = x_1' \cdot f_0(x_1, x_2, \cdots, x_n) \oplus x_1 \cdot f_1(x_1, x_2, \cdots, x_n) \tag{7}$$

$$f(x_1, x_2, \cdots, x_n) = 1 \cdot f_0(x_1, x_2, \cdots, x_n) \oplus x_1 \cdot f_2(x_1, x_2, \cdots, x_n) \tag{8}$$

$$f(x_1, x_2, \cdots, x_n) = 1 \cdot f_1(x_1, x_2, \cdots, x_n) \oplus x_1' \cdot f_2(x_1, x_2, \cdots, x_n) \tag{9}$$

where $f_0(x_1, x_2, \cdots, x_n) = f(0, x_2, \cdots, x_n) = f_0$ is the negative cofactor of variable x_1, $f_1(x_1, x_2, \cdots, x_n) = f(1, x_2, \cdots, x_n) = f_1$ is the positive cofactor of variable x_1, and $f_2(x_1, x_2, \cdots, x_n) = f(0, x_2, \cdots, x_n) \oplus f(1, x_2, \cdots, x_n) = f_0 \oplus f_1$. All operations in Equations (7)-(9) are performed using Boolean algebra, *i.e.*, $\oplus$ is Boolean XOR and $\cdot$ is Boolean multiplication.

As stated previously, the concept of lattice networks for switching functions involves three components [2]: 1) *expansion* of a function that corresponds to the root (initial node) in the lattice which creates several successor nodes of the expanded node, 2) *joining* of several nodes of a decision tree level to a single node which is the reverse operation of the expansion process, and 3) *regular geometry* to which the nodes are mapped that guides which nodes of the level are to be joined. While the realization of Boolean non-symmetric functions in Akers arrays requires an exponential growth of the repetition of variables in the worst case, the realization of Boolean non-symmetric functions in lattice networks requires a linear growth of repetition of variables, and consequently one does not need to repeat the variables of non-symmetric functions too many times to realize such functions in lattice networks for most practical benchmarks [2].

Figure 12 illustrates, as an example, the geometry of a four-neighbor lattice network and joining operations on the nodes where each cell has two inputs and two outputs (*i.e.*, four neighbors). The construction of the lattice network in **Figure 12** implements the following one possible convention of top-to-bottom expansion and left-to-right joining (*i.e.*, left-to-right propagation of the corresponding correction functions in **Figure 12(c)** and **Figure 12(d)**, respectively). The function that is generated by joining two nodes (sub-functions) in a lattice network is called the joined function. The function that is generated in nodes other than the joining nodes, to preserve the functionality within the lattice network, is called the correction function. Note that the lattices shown in **Figure 12** preserve the functionality of the corresponding sub-functions *f* and *g*. This can be observed, for instance, in **Figure 12(b)** as the negated variable {*a*'} will cancel the un-complemented variable {*a*}, when propagating the cofactors from the lower levels to the upper levels or vice versa, without the need for any correction functions to

preserve the output functionality of the corresponding lattice network. This simple observation cannot be seen directly in Figure 12(c) and Figure 12(d), as the correction functions are involved to cancel the effect of the new joining nodes for the preservation of the functionality of the new lattice networks (these correction functions are shown in the extreme right leaves of the second level in Figure 12(c) and Figure 12(d), respectively). It is shown that every function that is not symmetric can be symmetrized by repeating variables, and that a totally symmetric function can be obtained from an arbitrary non-symmetric function by the repetition of variables.

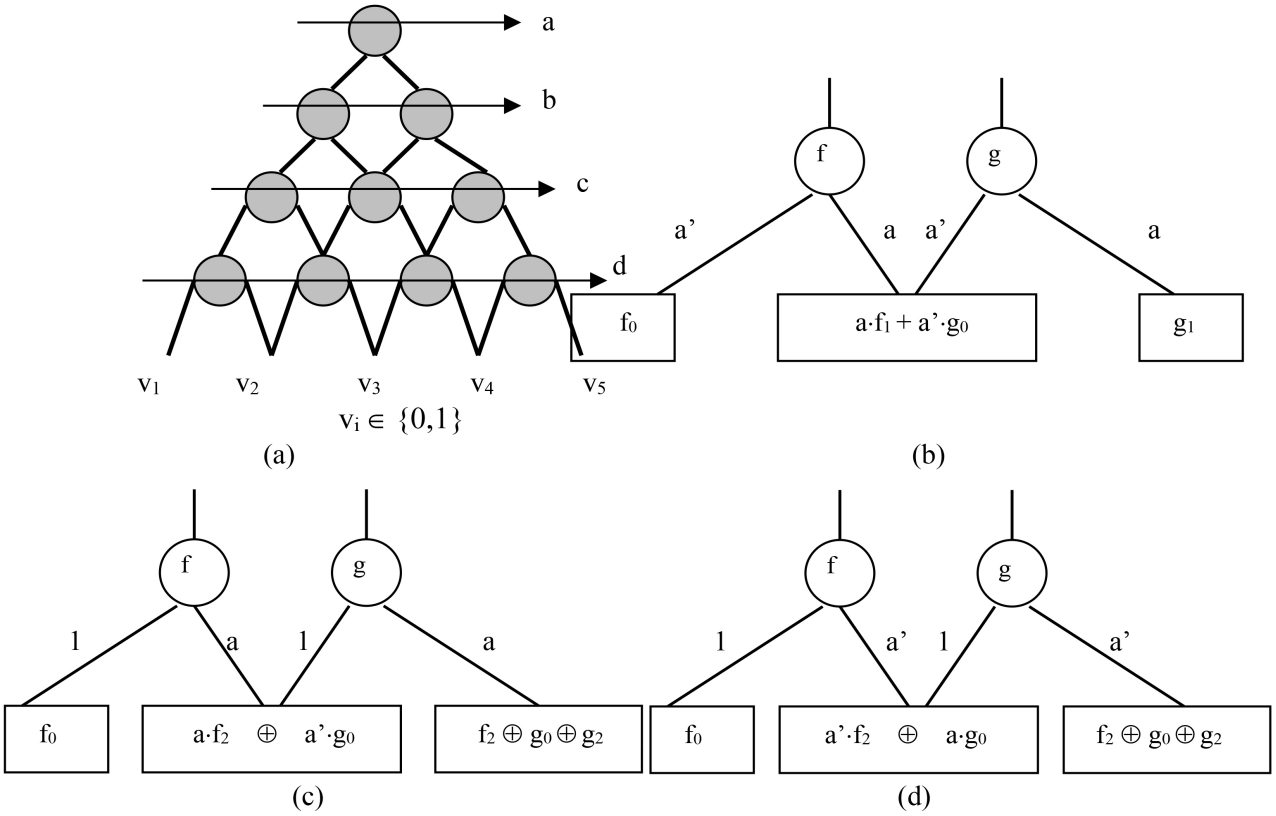


Figure 12. Two-dimensional binary lattice networks: (a) two-dimensional 4-neighbor lattice network, (b) Shannon lattice network, (c) positive Davio lattice network, and (d) negative Davio lattice network.

Example 10. For the following non-symmetric three-variable Boolean function $F = a \cdot b + a' \cdot c$, by utilizing the joining rule that was presented in Figure 12(b) for two-dimensional lattice network with binary Shannon nodes, one obtains the lattice network shown in Figure 13. One can note that without the repetition of variable(s) (e.g., variable b in Figure 13) F cannot be produced by any lattice network. It is also to be noted that all internal nodes in Figure 13 are two-to-one multiplexers (*i.e.*, selectors). In Figure 13, if one multiplies each leaf value, from left to right, with *all* possible bottom-up paths (from the leaves to the root F) and add them over Boolean algebra then one obtains the function F (*i.e.*, the root) as follows:

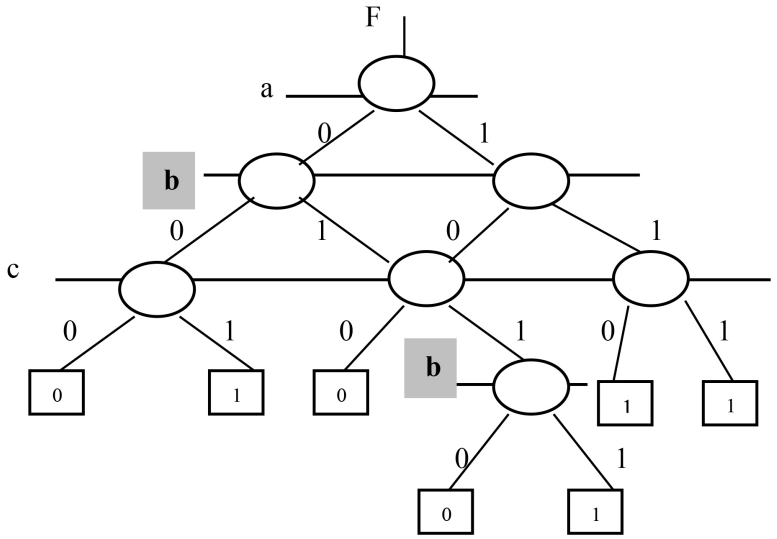


Figure 13. Shannon lattice network for the corresponding non-symmetric function $F = a \cdot b + a' \cdot c$.

$$\begin{aligned} F &= (0 \cdot c' \cdot b' \cdot a') + (1 \cdot c \cdot b' \cdot a') + (0 \cdot c' \cdot b \cdot a') + (0 \cdot b' \cdot c \cdot b \cdot a') \\ &\quad + (1 \cdot b \cdot c \cdot b \cdot a') + (1 \cdot c' \cdot b \cdot a) + (1 \cdot c \cdot b \cdot a) \\ &= (1 \cdot c \cdot b' \cdot a') + (1 \cdot b \cdot c \cdot b \cdot a') + (1 \cdot c' \cdot b \cdot a) + (1 \cdot c \cdot b \cdot a) \\ &= a' \cdot c + a \cdot b \end{aligned}$$

One can observe that in order to represent the non-symmetric function in Example 10 in the lattice network, variable b is repeated, where the nodes in Figure 13 are Shannon nodes, which are merely two-input one-output multiplexers, whose output goes in two directions, with the set of variables $\{a, b, c\}$ operate as control signals.

One method to define the symmetry of a Boolean function is by using symmetry indices S^i [2].

Definition 5. A symmetry index (S^i) has superscript i equals to the count of the number of "1" values in the states of variables in the corresponding cell in a Karnaugh map.

Example 11. For the binary non-symmetric implication function $F = a' + b$, Figure 14 illustrates the relationship between the karnaugh map with non-conflicting symmetry indices S^i and the lattice network with non-conflicting leaves, which is achieved by repeating variable $\{a\}$ twice in the two-dimensional lattice network.

It is to be noted that all internal nodes in Figure 14 are two-to-one multiplexers, where if one multiplies each leaf value (from left to right) with all possible bottom-up paths (from the leaves to the root F) and add them over Boolean algebra then one obtains the required function F in the root. Regular lattice networks that are presented in this section will be used to synthesize the corresponding functions within the homeomorphic Viterbi algorithm which will be introduced in Section 5.

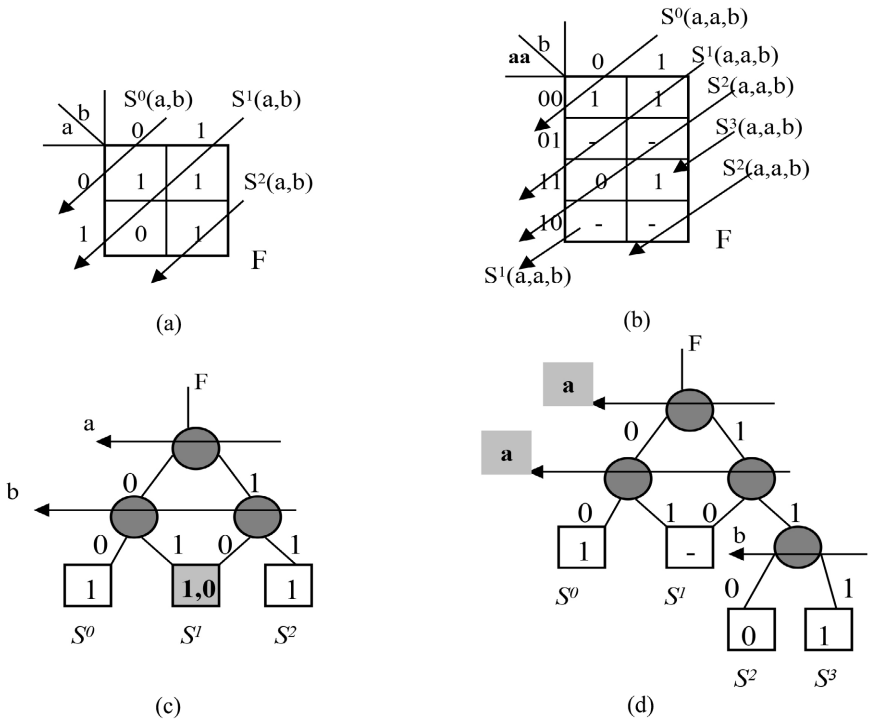


Figure 14. Using symmetry indices S^i: (a) non-symmetric implication Boolean function, (b) symmetrization by the repetition of variable $\{a\}$, (c) two-dimensional lattice network that corresponds to (a) with conflicting leaf (in dark box), and (d) two-dimensional lattice network that corresponds to (b) with non-conflicting leaves.

4. Carbon Field Emission-Based Controlled Switching and Computing

This section presents important and needed background on carbon-based nanotubes and nanotips where the presented carbon-based nanotubes and nanotips will be utilized in building controlled-switching devices and circuits that will be used in Section 5 for the construction of nano-based lattice homeomorphic Viterbi circuits.

4.1. Carbon Nanotubes Characteristics and Properties

The CNT, which is a cylindrical sheet of graphite, is formed geometrically in two distinct forms which affect CNT properties: 1) straight CNT in which a CNT is formed as a straight cut from graphite sheet and rolled into a carbon nanotube and 2) twisted CNT in which a CNT is formed as a cut at an angle from graphite sheet and rolled into a carbon nanotube. The newly emerging CNT technology has been implemented in many new promising applications [4]-[8] [10] [12] [14]-[16]. This includes TVs that are based on field emission of CNTs that consume much less power, thinner and have much higher resolution than the best plasma-based TVs available, nano circuits based on CNTs such as CNT-based FETs that consume less power and are much faster than the available silicon-based FETs, carbon nanocoils that can be used as inductors in nanofilters and as nanosprings

in nano dynamic systems, and CNT rings. In addition, the CNT has also been demonstrated for potential exciting applications such as in CNT probes, new composite materials, CNT data storage devices capable of storing 10^{15} bytes/cm^2, drug delivery systems, nano lithography, and CNT gears in which larger gears drive smaller gears or vice versa The CNT growth, as observed using Transmission Electron Microscopy, Atomic Force Microscopy and Scanning Electron Microscopy, requires processes with correct conditions and materials.

The field emission property is used in the recently developed highly efficient CNT-based TVs and is used in newly developed prototype vacuum tube lamps in six colors that are twice as bright as conventional light bulbs, longer-lived and at least ten times more energy-efficient. The properties of current carrying capacity, thermal stability, power consumption and electron scattering are qualifying the CNT for very promising future use in highly efficient power transmission. The property of preserving the electron spin will be utilized in using the CNT for reliable nano computing. The size property is very useful for using CNTs as nanowires that would result in decreasing the total size of areas and volumes that are occupied by wires and interconnects in the corresponding integrated circuits. The property of resilience is useful in building circuits and structures that has to maintain stress without structural damages. The CNT property of energy band gaps qualifies CNTs to be used in wide applications that require wide range of energy band gaps from the conductor state to the semiconductor state. In addition, CNTs are excellent field emitters and facilitate the miniaturization of electronic devices.

4.2. Carbon Nanotube-Based Multiplexing Devices

Figure 15 shows the synthesis of a logic inverter, that operates the same way as a CMOS inverter, using two nanotube FETs [4] [16]. In Figure 15, silicon (Si) is used as back-gate in the device, silicon dioxide (SiO_2) is used as an insulator, gold is used as electrodes (conductors), and PMMA is a cover that protects anything beneath it from being exposed to oxygen (O_2) where oxygen is used to convert n-CNTFET to p-CNTFET.

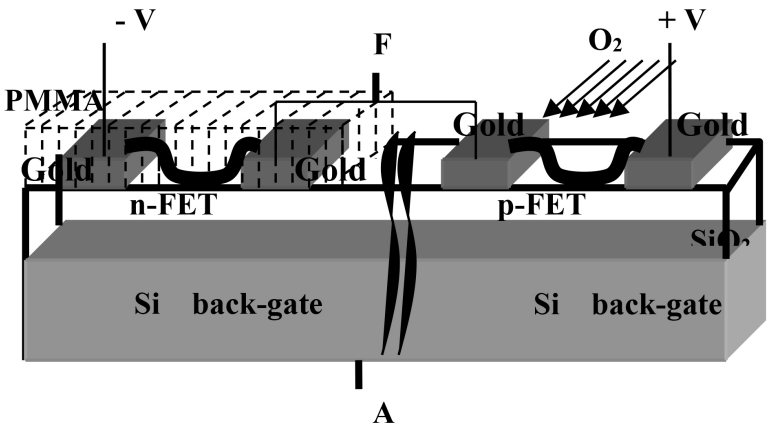


Figure 15. The NOT (inverter) logic gate using two nanotube-based FETs.

Figure 16 shows solid-state CNT-based multiplexer [4] where CNT is used as a channel in a Field Effect Transistor (FET). As was shown in Figure 15, the use of CNT as a channel in FET has been shown and a simple inverter that works exactly the same way as an ordinary CMOS inverter was built. Here, we use the same principle of using CNT as a channel but in a different device that uses two CNT n-FETs and two CNT p-FETs with different topologies of interconnects that are used for inputs A and B, controls C and C', and the output F. In Figure 16, silicon (Si) is used as back gate in the device, silicon dioxide (SiO_2) is used as an insulator, and gold is used as electrodes (conductors). Four metallic catalyst islands with each pair located at the facing ends of each pair of gold electrodes can be also used to grow CNTs between each pair of gold electrodes, rather than just placing CNTs in contact with the gold electrodes. The PMMA is a cover that protects anything beneath it from being exposed to oxygen (O_2) where oxygen is used to convert an n-CNTFET to a p-CNTFET. The function of the device in Figure 16 can be analyzed as follows [16]: If C = "1" then the upper transmission gate (t-gate) is activated and A is passed to F while the lower t-gate is deactivated and B is not passed to F, and if C = "0" then the lower t-gate is activated and B is passed to F while the upper t-gate is deactivated and A is not passed to F. Thus, the solid-state CNT multiplexer functions exactly as a 2-to-1 multiplexer.

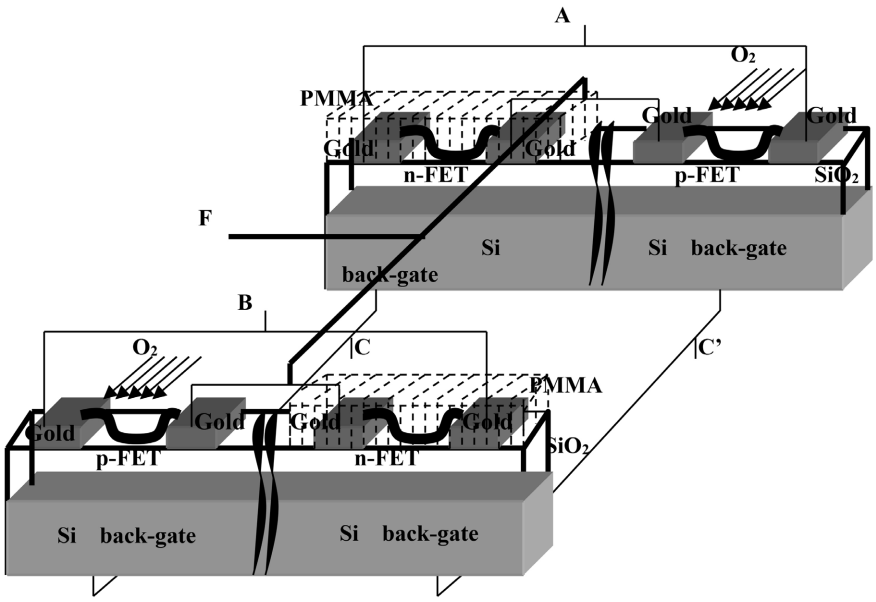


Figure 16. The device structure of solid-state inter-molecular CNT multiplexer.

Figure 17 shows the magnetic CNT-based multiplexer [4]. The numbers on the parts of the device in Figure 17 indicate the following: (#1, #2, #3, #4, #5, #6) are CNTs, (#7, #8, #9, #10, #11) are electrical current directions, (#12) is the body of the device which is an electrical insulator such as glass, SiO_2, plastics or any other type of electrical insulator, and (#13) is a hollow space with structural walls in two opposite sides (sides of CNTs #1 and #2) made of an electrical insulator.

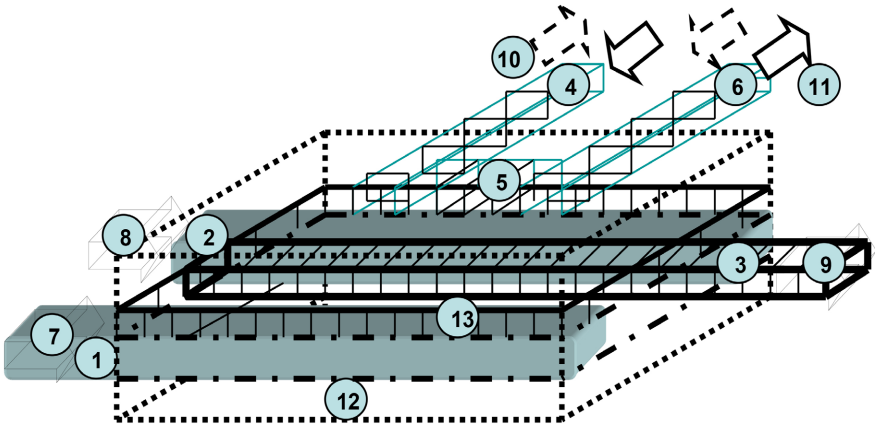


Figure 17. The device structure of the magnetic CNT-based multiplexer.

The device in Figure 17 operates as follows: electrical current (#7) is input B and can have two levels x and y that indicate logics "0" and "1", respectively. Electrical current (#8) is input A and can have two levels x and y that indicate logics "0" and "1", respectively. The two electrical currents (#7, #8) are flowing in the directions indicated in Figure 17 in CNTs (#1) and (#2), respectively. The flow of these currents will produce magnetic fields around CNTs (#1, #2) according to Ampere's (circuital) law in the Maxwell's equations that relates the magnetic field around a closed loop to the electric current passing through the loop for which the differential form representation in terms of the total charge and current takes the formulation as shown in Equation (10).

$$\vec{\nabla} \times \vec{B} = \mu_0 \vec{J} + \varepsilon_0 \mu_0 \frac{\partial \vec{E}}{\partial t} \tag{10}$$

where $\vec{B} = \mu \vec{H}$ is the magnetic field (magnetic flux density; magnetic field density), $\vec{H}$ is the magnetizing field (or magnetic field intensity), $\vec{\nabla} \times \vec{B}$ is the curl of the magnetic field, μ_0 is permeability, ε_0 is permittivity, $\vec{J}$ is current density, $\vec{E} = \frac{\vec{D}}{\varepsilon}$ is the electric field, $\vec{D}$ is the electric displacement field (or electric flux density), and $\frac{\partial \vec{E}}{\partial t}$ is the time derivative of the electric field. The direction of such magnetic fields follows the conventional right-hand thumb rule.

The CNT (#3) initially is in contact with either CNT (#1) or CNT (#2), and thus a current would be running in CNT (#3) in the same direction as currents (#7, #8). An electrical current is pumped through CNTs (#4, #5, #6), and this current can be pumped either in clock wise (CW) direction (#10) or counter clock wise (CCW) direction (#11). The electrical current (#10) flowing in CNTs (#4, #5, #6) will produce a magnetic field according to Maxwell's equations in a direction according to the right-hand thumb rule, and if current (#11) flows in CNTs (#4, #5, #6) a magnetic field will be produced in an opposite direction of that which is produced by current (#10). The direction of the current flowing in CNTs (#4, #5, #6) and thus the direction of the magnetic fields plays the role of the control signal

in a regular multiplexer as follows: if current (#11) is flowing in CNTs (#4, #5, #6) then an attractive Lorentz force occurs as shown in Equation (11):

$$\vec{F} = \vec{I} \times \vec{B} \tag{11}$$

where $\vec{I} \times \vec{B}$ is the cross product between current $\vec{I}$ in a conductor that lays in magnetic field $\vec{B}$, and $\vec{F}$ is the Lorentz force between two current-carrying conductors that occurs between CNT (#5) and CNT (#3) that causes CNT (#3) to move in the space (#13) towards CNT (#5) and thus makes a contact with CNT (#2) which means that current (#8) or input A is selected to flow to the output. On the other hand, if current (#10) is flowing in CNTs (#4, #5, #6) then a repulsive Lorentz force occurs between CNT (#5) and CNT (#3) that causes CNT (#3) to move in the space (#13) away from CNT (#5) and thus makes a contact with CNT (#1) which means that current (#7) or input B is selected to flow to the output. Thus, the nano mechanical device in **Figure 17** implements a two-to-one logic multiplexer.

4.3. Carbon Nanotubes-Based Field Emitters

Field electron emission is the emission of electrons from the surface of a cathode under the influence of the applied electric field (of 3×10^9 V/m) which is strongly dependent upon the work function of the emitting material. The general form of Fowler-Nordheim-type (FN-type) equation can be produced as shown in Equation (12) [66] [67]:

$$J = \lambda_L a \varphi^{-1} F^2 \exp\left[-V_F b \varphi^{3/2} / F\right] \tag{12}$$

Equation (12) is used in all cases of field emission processes, where J is the local emission current density, a and b are the first and the second Fowler-Nordheim constants, respectively, ν_F is the barrier form correction factor and it accounts for the particular shape of the potential barrier model, and λ_L is the local pre-exponential correction factor where it takes into account all of the other factors that influence the emission. The factors ν_F and λ_L depend on the applied field F.

This section presents important functional modeling of field emitters and the corresponding experimental setups and measurements that characterize their time-dependent response. In order to perform the static modeling of CNT field emitters, four field emission carbon nanotubes, which are shown in **Figures 18(b)-(e)**, were manufactured by Xintek, Inc. The copper anode is at the right and the CNT emitter is mounted on a tungsten wire attached to the copper cylinder at the left as shown in **Figure 18(a)**. **Figures 18(b)-(e)** show the images of CNT emitters for each carbon nanotube, taken with a JEOL Ltd. model JEM 6300 SEM. Carbon nanotubes M-1 and M-4 have a single MWCNT as the emitter, and carbon nanotubes C-3 and C-6 have a single SWCNT as the emitter. The used CNTs were formed in bundles that have diameters of 10 - 30 nm, but in each carbon nanotube the field emission is from the one CNT at the end of the bundle where the electric field is most intense.

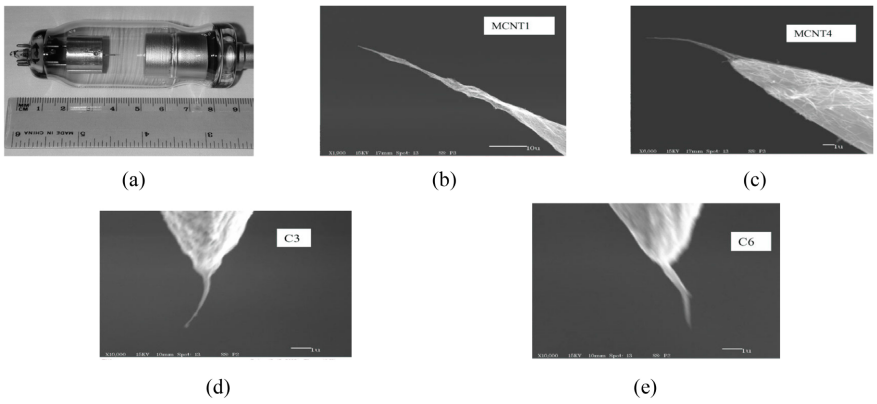


Figure 18. Carbon nanotube-based field emission: (a) structure of the field emission carbon nanotubes manufactured by Xintek Inc. and (b)-(e) Scanning Electron Microscopy images of the CNT emitters in the four utilized carbon nanotubes.

The DC current-voltage characteristics were measured for these four carbon nanotubes, as well as a field emitter tube from Leybold Didactic GmbH, which has an etched single crystal of tungsten as the emitter. All of the measurements that were made with the five tubes were performed at room temperature. The tungsten tip is mounted on a filament so that this tip is heated for cleaning shortly before each session of measurements. However, it is not possible to clean the CNT, which probably causes the "switch-on" effect in which the supply voltage must be momentarily increased well beyond the operating point to initiate field emission with the CNT. The data from the DC measurements were reduced by the Fowler-Nordheim analysis which is based on the simplified form of the Fowler-Nordheim in Equation (13) that provides the magnitude of the current density as a function of the applied static field for the field emission from a specific material:

$$J = AE^2 \mathrm{e}^{-B/E} \tag{13}$$

where J and E are the magnitudes of the current density and the electric field intensity, $A = 1.541 \times 10^{-6}/\Phi$, and $B = 6.831 \times 10^9 \Phi^{3/2}$. The work function values are $\Phi = 4.5$ eV for tungsten and (for the CNT) was set to $\Phi = 4.9$ eV for graphene. In order to apply the Fowler-Nordheim equation to the DC current-voltage data, Equation (14) was also used which is valid for a given carbon nanotube, where I is the field emission current and V is the potential applied between the anode and the cathode:

$$I = CV^2 \mathrm{e}^{-D/V} \tag{14}$$

Equations (13)-(14) can be combined to obtain Equations (15)-(16) for the parameters S and R, which are used to characterize the field emitters:

$$S = CD^2 / AB^2 \tag{15}$$

$$R \equiv V/E = D/B \tag{16}$$

where parameter S refers to the effective emitting area which would be the physical area of the emitter if the current density were uniform over a fixed area and zero elsewhere, and parameter R refers to the effective radius of curvature of the

emitter but also includes the effects of local intensification of the electric field caused by elongation of the emitter or the reduction of the field which may be caused by shielding due to adjacent structures.

The Fowler-Nordheim plots of the DC current-voltage data were conducted using $\ln(I/V^2)$ as the ordinate and $(1/V)$ as the abscissa. Equation (14) requires that these plots should be straight lines and typically with correlation $c \approx -0.998$. Figure 19 shows the anomalous data which were obtained using the 2.575 GΩ ballast resistor. The values of parameter R, the effective radius of curvature of the emitter, were found to vary within 77 - 110 nm for the four carbon nanotubes with CNT emitters. This suggests that values of the local electric field at the emitting sites were as high as 14 V/nm in some of these measurements. The Fowler-Nordheim analysis gave a value of 91 nm for the effective radius of curvature of the emitter in the Leybold tube, suggesting that the local electric field was as high as 5 V/nm in some of the performed measurements. The Fowler-Nordheim analysis also showed that the parameter S, the effective emitting area, varied within 81 - 230 nm^2 for the four carbon nanotubes with CNT emitters. If the current density was uniform, this would correspond to circular emitting spots having radii of approximately 5 - 9 nm. The Fowler-Nordheim analysis also showed that the effective emitting area for the tungsten tip in the Leybold tube would correspond to a hemisphere with a radius of 290 nm.

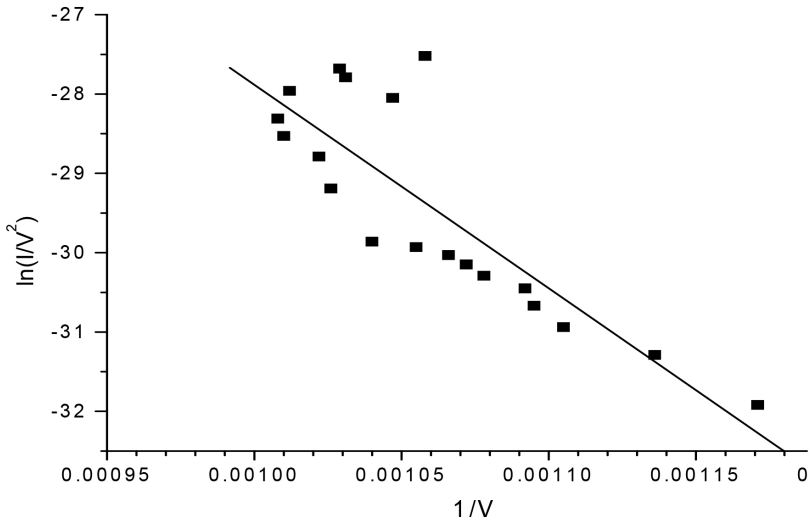


Figure 19. The obtained Fowler-Nordheim plot for CNT C-6 with 2.575 GΩ ballast resistor.

4.4. Carbon Field Emission-Based Two-to-One Controlled Switching

Multiplexing also known as controlled switching is considered as a highly fundamental operation in digital systems [68]. This sub-section presents carbon field emission-based controlled switching. By the utilization of the previously experimented and observed characterizations and operations of carbon field-emission from the corresponding nano-apex carbon fibers and CNTs that were previously

presented, Figure 20 presents the carbon field emission-based primitive that realizes the two-to-one controlled switching. In Figure 20, the input control signal that is used to control the electric conduct of the device is implemented using the imposed electric field intensity (E) or equivalently the work function (Φ) or voltage (V). The description of the operation of the carbon field emission-based device shown in Figure 20(b) is as follows: by imposing the control signal of high voltage (HV), the voltage difference between the carbon cathodes and the facing anode is varied. This change will make the carbon cathode with control signal (HV) to be field emitting while the other carbon cathode with the complementary control signal ($\overline{\text{HV}}$) to be without field emission. When the voltage difference is reversed, the carbon cathode with the complementary control signal ($\overline{\text{HV}}$) will be field emitting while the other carbon cathode with the control signal (HV) will be without field emission. Thus, this device implements the functionality of the 2-to-1 controlled switching ($G = ac + bc'$) which is shown in Figure 20(a).

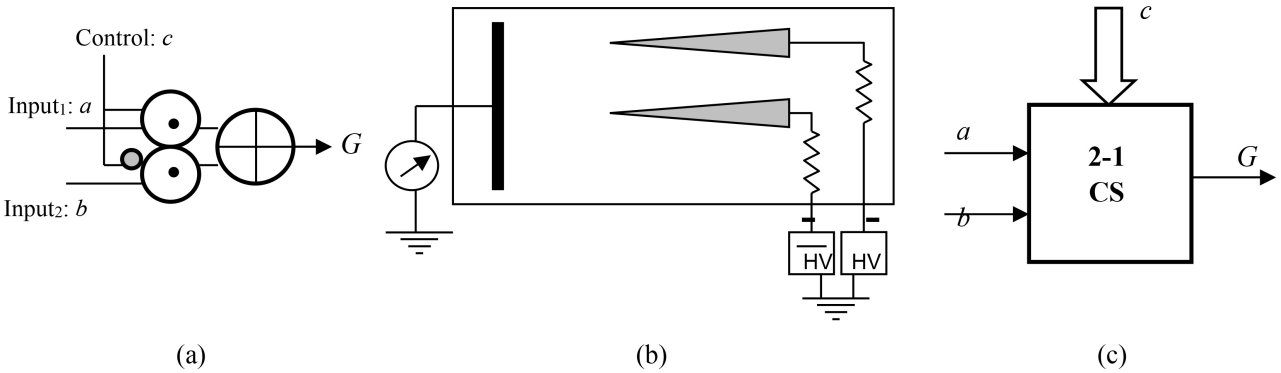


Figure 20. The carbon field emission-based device implementing the operation of the two-to-one controlled switching (CS): (a) two-to-one multiplexer ($G = ac + b\bar{c}$), (b) the carbon field emission–based two-to-one CS, and (c) block diagram for the new two-to-one multiplexer.

The experimental results show that the distance d which is required between the cathodes and the facing anode must be generally around 10 mm, else beam distortion will occur and will be affecting the collected current at the facing anode screen.

Since the equations that relate the electric field intensity (E), work (*i.e.*, energy) function (Φ) in Joules (J), distance (d), voltage (V), and the current density (J), are shown in Equations (17)-(20):

$$\Phi = e \cdot V \tag{17}$$

$$V = E \cdot d \tag{18}$$

$$d = V/E \tag{19}$$

$$J = I/(a/\Omega) = I \cdot (\Omega/a) \tag{20}$$

where e is the electron charge $\cong 1.602 \times 10^{-19}$ C, a is the tip area and Ω is the emission angle, then the equation that models the current value on the anode screen can be derived as shown in Equation (21):

$$I = \left(\frac{aA}{\Omega d^2} \mathrm{e}^{-\frac{Bd}{V}} \right) V^2 \tag{21}$$

From Equation (21), one can clearly observe the corresponding proportionality relation between the current value I and the voltage difference V, proportionality relation between the current value I and the tip area a, and the inverse relation between the current value I and the emission angle Ω. The typical used value for the work function Φ is 4.5 eV for tungeston and 4.9 eV for graphene, with utilized electric field intensity of around 3×10^9 V/m for a distance of 10 mm between the carbon nanotube cathode and the facing anode screen with applied voltage of 3×10^7 V. The Fowler-Nordheim analysis also showed that the effective emitting area for the tungsten tip in the Leybold tube correspond to a hemisphere with a radius of 290 nm.

5. Nano Circuit Design of the Homeomorphic Viteribi Algorithm

This section introduces the new implementation of nano circuit design of the homeomorphic Viterbi algorithm. The functions inside the design are implemented regularly using the regular lattice networks as was shown and presented in Section 3. The multiplexing nodes in the lattice networks are then practically achieved utilizing carbon field emission-based controlled switching that was presented in Section 4. The homeomorphic hardware implementation for each Viterbi cell in the (homeomorphic) Viterbi algorithm requires the following homeomorphic components: homeomorphic modulo-2 adder, homeomorphic arithmetic adder, homeomorphic subtractor and homeomorphic selector (*i.e.*, homeomorphic multiplexer) to be both used in one possible design of the corresponding homeomorphic comparator. Table 2 shows the truth tables of an non-homeomorphic half-adder, non-homeomorphic subtractor and non-homeomorphic full-adder.

Table 2. Truth tables: (a) non-homeomorphic half-adder and subtractor and (b) non-homeomorphic full-adder.

(a)

Inputs		Half-Adder		Subtractor	
		Outputs		Outputs	
a	b	$a + b$	Carry	$a - b$	Borrow
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	0	1	0
1	1	0	1	0	0

(b)

Inputs			Outputs	
a	b	c_i	s	c_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0

Continued

0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 21 shows the various nano circuits for the nano realization of each nano Viterbi cell in the corresponding (homeomorphic) Viterbi algorithm. Figures 21(a)-(c) present fundamental nano gates. Figures 21(d)-(g) show basic nano arithmetic circuits of nano subtractor (Figure 21(d)), nano half-adder (Figure 21(e)), nano full-adder (Figure 21(f)), and the nano equality-based comparator (Figure 21(g)). Figure 21(h) introduces the basic nano Viterbi cell which is made of two Controlled-NOT gates, one nano half-adder, one nano full-adder and one nano comparator with multiplexing.

Figure 22 shows the logic circuit design of an iterative network to compare two 3-digit binary numbers: $X = x_1x_2x_3$ and $Y = y_1y_2y_3$, and Figure 23 presents the detailed synthesis of a comparator circuit which is made of a comparator cell (Figure 23(a)) and a comparator output circuit (Figure 23(b)). The extension of the circuit in Figure 22 to compare two n-digit binary numbers is straightforward by utilizing n-cells and the same output circuit.

Figure 24 illustrates the nano circuit synthesis for the comparator cell and the output circuit (which were shown in Figure 23), and Figure 25 shows the design of a nano comparator with multiplexing where Figure 25(a) shows an iterative nano network to compare two 3-digit binary numbers and Figure 25(c) shows the complete design of the nano comparator with multiplexing. The extension of the nano circuit in Figure 25(a) to compare two n-digit binary numbers is done by utilizing n nano cells (from Figure 24(a)) and the output nano circuit (in Figure 24(b)).

Figure 26 shows the complete design of a nano Viterbi cell in the Viterbi algorithm that was shown in Figure 21(h). The design of the nano Viterbi cell shown in Figure 26(f) proceeds as follows: 1) two nano circuits for the first and second lines entering the Viterbi cell each is made of two nano XORs to produce the difference between incoming received bits and trellis bits followed by nano half-adder to produce the corresponding sum (which is the Hamming distance) are shown in Figure 26(a) and Figure 26(b), 2) logic circuit composed of a nano half-adder and a nano full-adder that adds the current Hamming distance to the previous Hamming distance is shown in Figure 26(c), 3) two nano circuits for the first and second lines entering the Viterbi cell each is synthesized according to the logic circuit in Figure 26(c) (which is made of a half-adder followed by a full-adder) are shown in Figure 26(d) and Figure 26(e), 4) nano comparator with multiplexing in the Viterbi cell that compares the two entering metric numbers (*i.e.*, two entering Hamming distances) and selects using the control line O_1 the path that produces the minimum entering metric (*i.e.*, minimum entering Hamming distance) is shown in Figure 26(f).

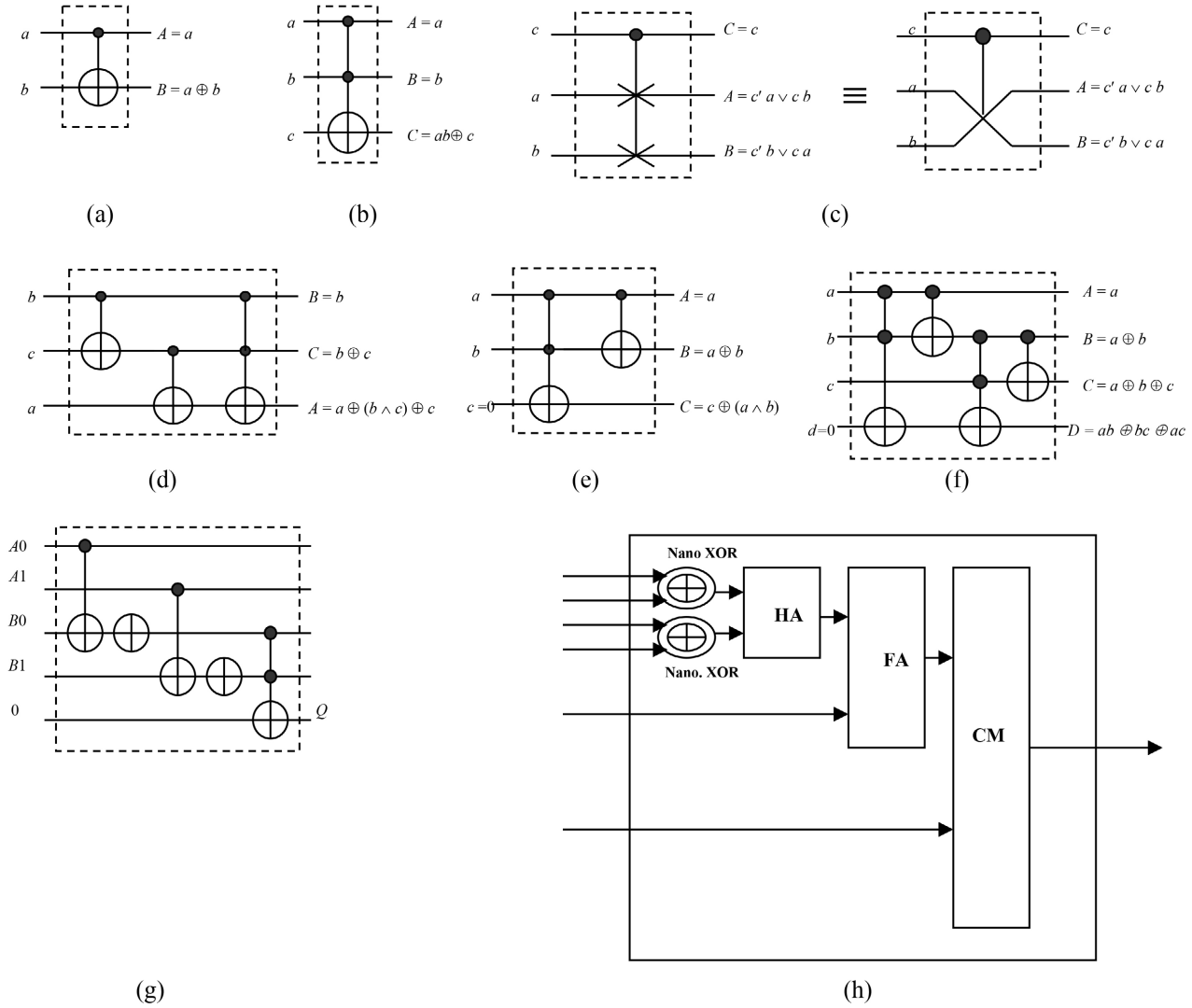


Figure 21. Nano homeomorphic circuits for the nano realization of each Viterbi cell in the corresponding homeomorphic Viterbi algorithm: (a) nano XOR gate (Controlled-NOT gate), (b) nano Controlled-Controlled-NOT gate, (c) nano multiplexer (Controlled-Swap gate), (d) nano subtractor, (e) nano half-adder (HA), (f) nano full-adder (FA), (g) nano equality-based comparator that compares two 2-bit numbers where an isolated XOR symbol means a nano NOT gate, and (h) basic nano homeomorphic Viterbi cell which is made of two Controlled-NOT gates, one nano HA, one nano FA and one nano comparator with multiplexing. The nano comparator can be synthesized using a nano subtractor and a multiplexer gate. The symbol ⊕ is logic XOR (exclusive OR; modulo-2 addition), ∧ is logic AND, ∨ is logic OR, and ′ is logic NOT.

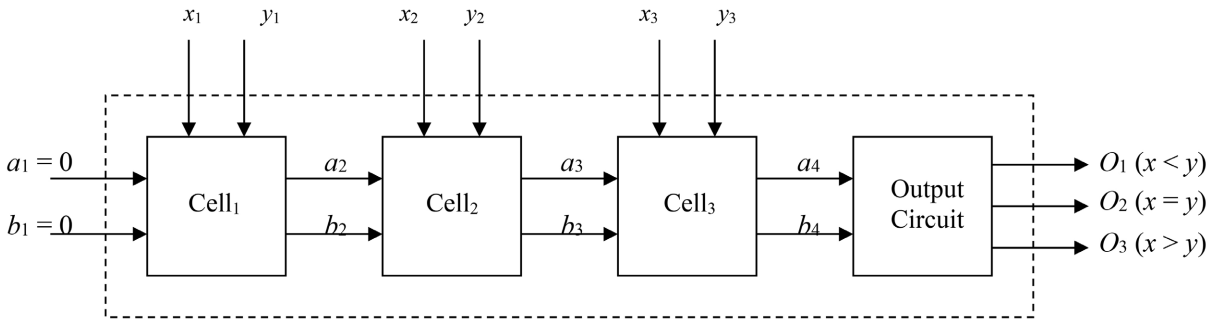


Figure 22. An iterative network to compare two 3-digit binary numbers: $X = x_1x_2x_3$ and $Y = y_1y_2y_3$.

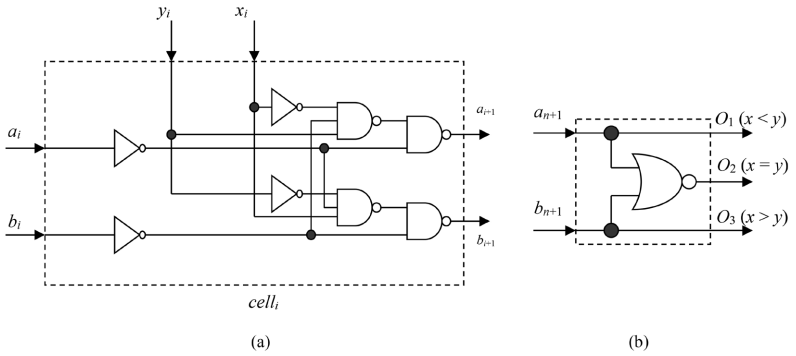


Figure 23. Designing a comparator circuit: (a) comparator cell and (b) comparator output circuit.

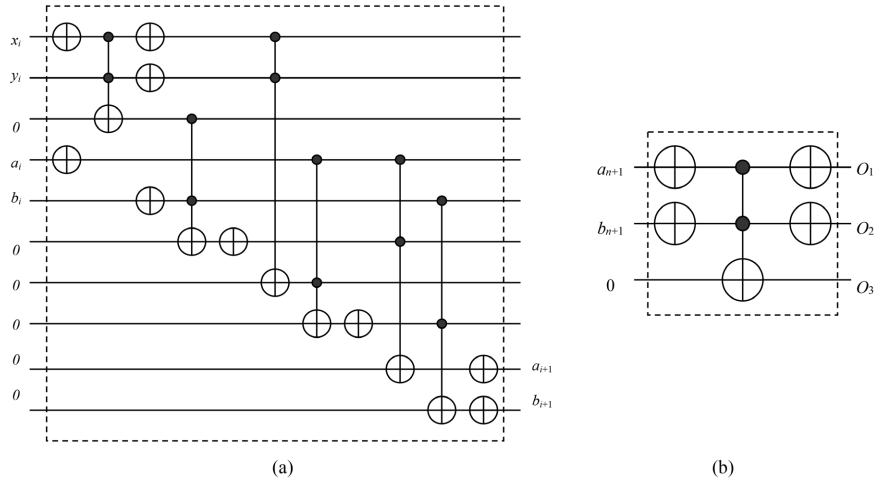


Figure 24. Nano circuit synthesis for the comparator cell and output circuit in **Figure 23**: (a) nano comparator cell and (b) nano comparator output circuit.

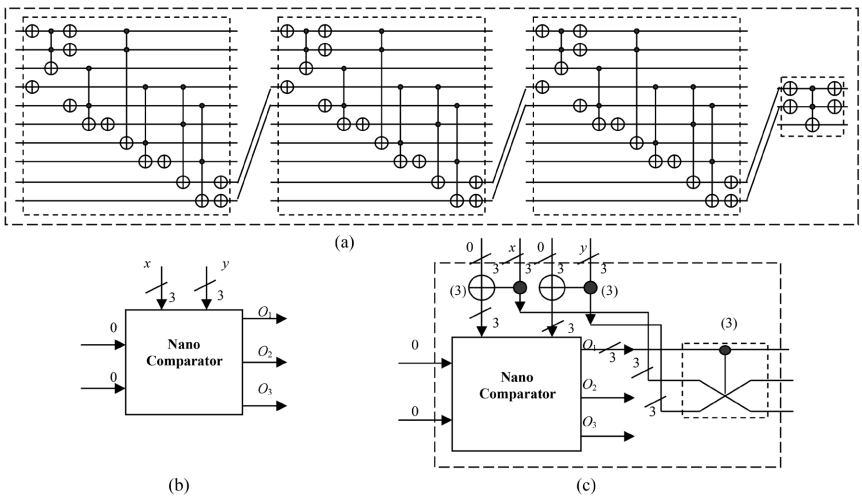


Figure 25. Designing a nano comparator with multiplexing: (a) an iterative nano network to compare two 3-digit binary numbers, (b) symbol of the nano comparator circuit in (a), and (c) complete design of nano comparator with multiplexing where the number 3 on lines indicates triple lines and (3) beside sub-circuits indicates triple circuits (*i.e.*, three copies of each sub-circuit for the processing of the triple-input triple-output lines.)

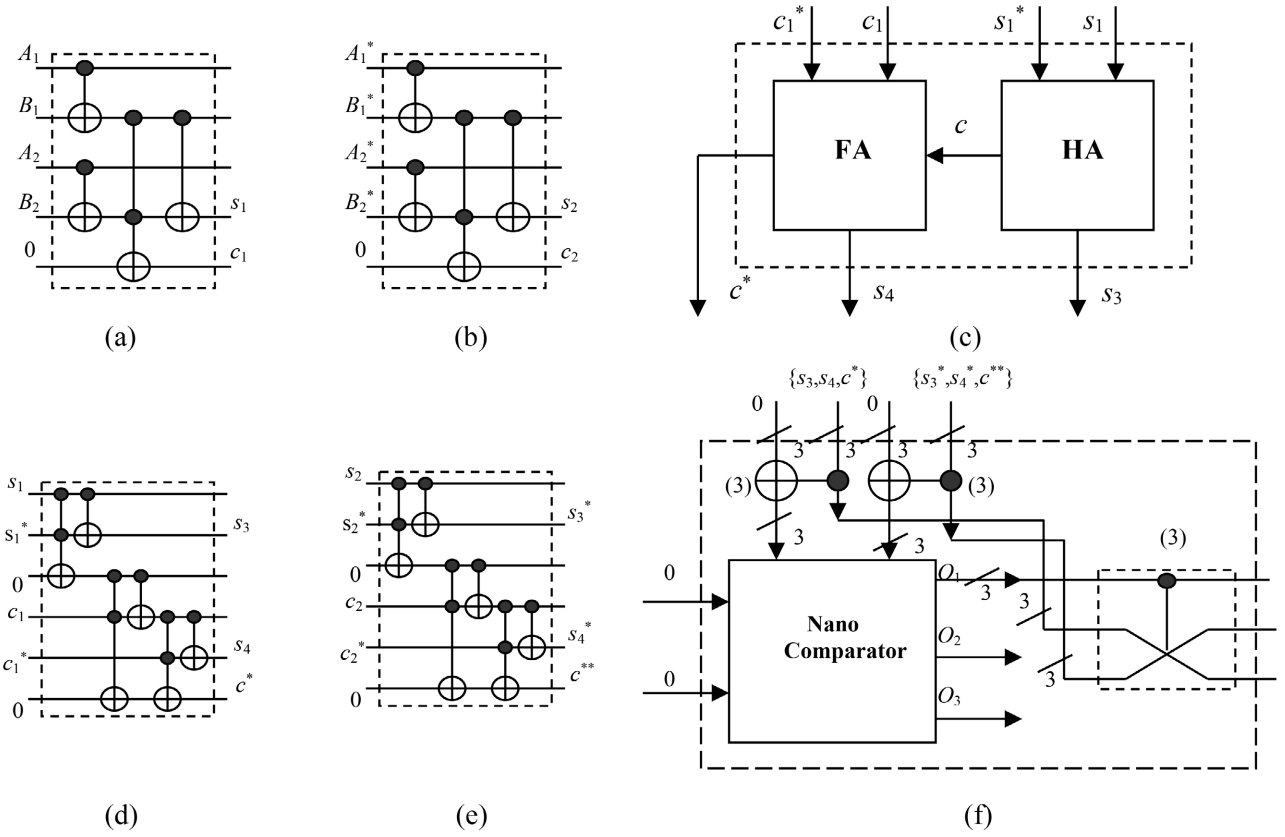


Figure 26. The complete design of a nano Viterbi cell in the Viterbi algorithm that was shown in **Figure 21(h)**: (a) nano circuit that is made of two nano XORs to produce the difference between incoming received bits (A_1 A_2) and trellis bits (B_1 B_2) followed by nano half-adder to produce the corresponding sum (s_1 c_1) which is the Hamming distance for the first line entering the Viterbi cell, (b) nano circuit that is made of two nano XORs to produce the difference between incoming received bits (A_1^* A_2^*) and trellis bits (B_1^* B_2^*) followed by nano half-adder to produce the corresponding sum (s_2 c_2) which is the Hamming distance for the second line entering the Viterbi cell, (c) logic circuit composed of nano half-adder and nano full-adder that adds the current Hamming distance to the previous Hamming distance, (d) nano circuit in the first line entering the Viterbi cell for the logic circuit in (c) that is made of a nano half-adder followed by a nano full-adder, (e) nano circuit in the second line entering the Viterbi cell for the logic circuit in (c) that is made of a nano half-adder followed by a nano full-adder, and (f) nano comparator with multiplexing in the Viterbi cell that compares the two entering metric numbers: $X = s_3\ s_4\ c^*$ and $Y = s_3^*\ s_4^*\ c^{**}$ and selects using control line O_1 the path that produces the minimum entering metric (*i.e.*, $X < Y$).

In **Figures 26(c)-(e)**, the current Hamming metric {s_1, c_1} for the first entering path of the Viterbi cell and the current Hamming metric {s_2, c_2} for the second entering path of the Viterbi cell is always made of two bits (00, 01, or 10). If more than two digits (two bits) is needed to represent the previous Hamming metric for the first or second entering paths of the Viterbi cell (e.g., $(5)_{10} = (101)_2$), then extra nano full-adders are added in the logic circuit in **Figure 26(c)** and consequently in the nano circuits shown in **Figure 26(d)** and **Figure 26(e)**. Also, in the case that when the paths entering a nano Viterbi cell (state) are compared and their metrics are found to be identical then a choice is made by making a guess to choose any of the two entering paths, and this is automatically performed in the nano circuit in **Figure 26(f)** since if ($\{s_3, s_4, c^*\} < \{s_3^*, s_4^*, c^{**}\}$) then O_1 = "1" and

thus chooses $X = \{s_3, s_4, c^*\}$, else O_1 = "0" and then it chooses $Y = \{s_3^*, s_4^*, c^{**}\}$ in both cases of ($\{s_3, s_4, c^*\} > \{s_3^*, s_4^*, c^{**}\}$) or ($\{s_3, s_4, c^*\} = \{s_3^*, s_4^*, c^{**}\}$).

As stated previously, this section introduced the new implementation of nano circuit design of the homeomorphic Viterbi algorithm. Each function inside the introduced design is implemented using a regular lattice network as was illustrated and presented in Section 3. This is done in either one of two ways: 1) *implementing each function separately using a lattice network and then connecting the resulting single lattice networks to produce the total function*, or 2) *producing the total function from sub-functions utilizing logic synthesis methods* [68] *and then implementing the resulting total function at once using the corresponding lattice network*. In either way, the multiplexing nodes in the resulting lattice network(s) are then practically achieved and implemented utilizing carbon field emission-based controlled switching that was demonstrated and presented from Section 4.

The property of homeomorphism in multiple-streams of communicated parallel data can be used, as was shown previously in Sub-section 2.3, for further correction of errors that are uncorrectable using the implemented decoding algorithm such as in the cases of the failure of the classical Viterbi algorithm in correcting for more than two errors. These errors can be caused from various reasons such as fading and burst errors. In general, fading occurs in wireless communication where the received signal strength fluctuates and varies due to various factors, such as multi-path propagation, leading to unreliable data transmission. In addition, a burst error is a sequence of corrupted bits in a data stream that are close to one another where unlike random errors (which occur independently) burst errors are correlated where an error in one bit increases the probability of errors in other neighboring bits. When a signal experiences deep fading, its strength drops significantly, which can corrupt not just one but a cluster of consecutive transmitted bits where this concentration of errors creates a burst error condition. Therefore, and since homeomorphism in Viterbi decoding can be used to correct for several bit errors, the new introduced design technique within the homeomorphism property can be useful to be used for correcting sequence of existing errors (in addition to using standard conventional techniques such as interleaving and burst-error-correcting codes).

6. Conclusion and Future Work

Since power reduction has become a main concern for modern digital systems besides speed and reliability, homeomorphism property in error-control coding is highly important because homeomorphism is a main requirement for low-power high-performance circuit synthesis of future technologies such as in nano and quantum computing. The homeomorphic design of multiple-stream Viterbi error-correcting codes using regular lattice networks and utilizing carbon field emission-based nano switching devices is introduced in this article. This is performed through the realization of operations that utilize the two-to-one basic con-

trolled-switching elements, where each of these basic elements can be directly implemented using the presented carbon-based field emission devices. It has been shown that the property of homeomorphism in multiple-streams of communicated parallel data can be used for further correction of errors that are uncorrectable using the implemented decoding algorithm such as in the cases of the failure of the classical Viterbi algorithm in correcting for more than two errors. Lattice networks possess the important property of high regularity which is useful in fault testing and localization, self-repair, compactness and ease of manufacturability. Further, and because of using only local interconnects and the utilization of efficient carbon-based nano switching, the introduced nano-based lattice homeomorphic Viterbi architecture can be utilized within several applications where higher reliability, more speed and minimal power consumption are required such as in high-performance reliable low-power wireless data communications. This article establishes fundamental and new foundations in the architectural implementation of convolutional homeomorphic error-correction coding using nano controlled-selectors and regular lattice networks, where future work will include further comprehensive and full-scale quantitative evaluation and comparison for various design parameters such as bit-error rate, total power consumption, time and area costs for typical constraint lengths that will be performed in simulation and prototyping to compare between the new introduced nano homeomorphic Viterbi design against the conventional and standard Viterbi decoder.

Acknowledgements

This research was performed during sabbatical leave in 2024-2025 granted to the author from The University of Jordan and spent at Zaytoonah University of Jordan.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Alnawasreh, S., Mousa, M.S. and Al-Rabadi, A.N. (2015) Investigating the Effects of Sample Conditioning on Nano-Apex Carbon Fiber Tips for Efficient Field Electron Emission. *Jordan Journal of Physics*, **8**, 51-57.

[2] Al-Rabadi, A.N. (2004) Reversible Logic Synthesis: From Fundamentals to Quantum Computing. Springer-Verlag.

[3] Al-Rabadi, A.N. (2009) Circuits For *m*-valued Classical, Reversible and Quantum Optical Computing with Application to Regular Logic Design. *International Journal of Intelligent Computing and Cybernetics*, **2**, 52-101. https://doi.org/10.1108/17563780910939255

[4] Al-Rabadi, A.N. (2009) Carbon NanoTube (CNT) Multiplexers, Circuits, and Actuators, United States Patent and Trademark Office, Patent No. US 7,508,039 B2.

[5] Al-Rabadi, A.N., Mousa, M.S., Al-Rabadi, R.F., Alnawasreh, S., Altrabsheh, R.F. and Madanat, M.A. (2016) Parallel Bijective Processing of Regular Nano Systolic Grids

via Carbon Field Emission Controlled-Switching. *International Journal of Computer Science and Information Technology*, **8**, 77-101. https://doi.org/10.5121/ijcsit.2016.8307

[6] Al-Rabadi, A.N. (2020) Concurrency within Ternary Galois Processing of Highly-Regular 3D Networks via Controlled Nano Switching. *International Journal of Computer Science and Information Technology*, **12**, 1-23. https://doi.org/10.5121/ijcsit.2020.12101

[7] N. Al-Rabadi, A. (2020) Concurrent Ternary Galois-Based Computation Using Nano-Apex Multiplexing Nibs of Regular Three-Dimensional Networks, Part II: Formalistic Architecture Realization. *International Journal of VLSI Design & Communication Systems*, **11**, 1-19. https://doi.org/10.5121/vlsic.2020.11601

[8] Amaratunga, G. (2003) Watching the Nanotube. *IEEE Spectrum*, **40**, 28-32. https://doi.org/10.1109/mspec.2003.1228005

[9] Bonard, J., Dean, K.A., Coll, B.F. and Klinke, C. (2002) Field Emission of Individual Carbon Nanotubes in the Scanning Electron Microscope. *Physical Review Letters*, **89**, Article 197602. https://doi.org/10.1103/physrevlett.89.197602

[10] Bonard, J.-M., Salvetat, J.-P., Stöckli, T., Forró, L. and Châtelain, A. (1999) Field Emission from Carbon Nanotubes: Perspectives for Applications and Clues to the Emission Mechanism. *Applied Physics A*: *Materials Science & Processing*, **69**, 245-254. https://doi.org/10.1007/s003390050998

[11] Brugat, M., Mousa, M.S., Sheshin, E.P. and Hagmann, M.J. (2002) Measurement of Field Emission Current Variations Caused by an Amplitude Modulated Laser. *Materials Science and Engineering*: *A*, **327**, 7-15. https://doi.org/10.1016/s0921-5093(01)01870-6

[12] Burke, P.J. (2002) Luttinger Liquid Theory as a Model of the Gigahertz Electrical Properties of Carbon Nanotubes. *IEEE Transactions on Nanotechnology*, **1**, 129-144. https://doi.org/10.1109/tnano.2002.806823

[13] Cleland, A.N. (2003) Foundations of Nanomechanics: From Solid State Theory to Device Applications. Springer.

[14] Collins, P.G. and Avouris, P. (2000) Nanotubes for Electronics. *Scientific American*, **283**, 62-69. https://doi.org/10.1038/scientificamerican1200-62

[15] Collins, P.G., Arnold, M.S. and Avouris, P. (2001) Engineering Carbon Nanotubes and Nanotube Circuits Using Electrical Breakdown. *Science*, **292**, 706-709. https://doi.org/10.1126/science.1058782

[16] Derycke, V., Martel, R., Appenzeller, J. and Avouris, P. (2001) Carbon Nanotube Inter- and Intramolecular Logic Gates. *Nano Letters*, **1**, 453-456. https://doi.org/10.1021/nl015606f

[17] Likharev, K. (2003) Electronics Below 10 Nm. In: Greer, J., Korkin, A. and Labanowski, J., Eds., *Nano and Giga Challenges in Microelectronics*, Elsevier, 27-68. https://doi.org/10.1016/b978-044451494-3/50002-0

[18] Roy, K. and Prasad, S. (2000) Low-Power CMOS VLSI Circuit Design. John Wiley & Sons Inc.

[19] Smith, K.C. (2002) Prospects for VLSI Technologies in MVL. Booklet of Ultra Large-Scale Integration (ULSI) Workshop, 4.

[20] Al-Rabadi, A.N. (2008) Bijective Digital Error-Control Coding, Part II: Quantum Viterbi Circuit Synthesis. In: Huang, DS., Wunsch, D.C., Levine, D.S. and Jo, KH., Eds., *Lecture Notes in Computer Science*, Springer, 23-30. https://doi.org/10.1007/978-3-540-87442-3_4

[21] Al-Rabadi, A. (2009) Closed-System Quantum Logic Network Implementation of the Viterbi Algorithm. *Facta universitatis-Series*: *Electronics and Energetics*, **22**, 1-33. https://doi.org/10.2298/fuee0901001a

[22] Al-Rabadi, A.N. (2009) Reversible Viterbi Algorithm and Its Closed-System Q-Domain Circuit Design and Computation. *Journal of Circuits*, *Systems and Computers*, **18**, 1627-1649. https://doi.org/10.1142/s0218126609005903

[23] De Vos, A. (1999) Reversible Computing. *Progress in Quantum Electronics*, **23**, 1-49. https://doi.org/10.1016/s0079-6727(99)00002-6

[24] Nielsen, M. and Chuang, I.L. (2000) Quantum Computation and Quantum Information. Cambridge University Press.

[25] Al-Rabadi, A.N. (2008) Bijective Digital Error-Control Coding, Part I: The Reversible Viterbi Algorithm. In: Huang, DS., Wunsch, D.C., Levine, D.S. and Jo, KH., Eds., *Lecture Notes in Computer Science*, Springer, 15-22. https://doi.org/10.1007/978-3-540-87442-3_3

[26] Al-Rabadi, A.N. (2008) Game Reversibility in Decision Space and Error-Control Coding. *International Journal of Mathematics*, *Game Theory and Algebra*, **18**.

[27] Bennett, C.H. (1973) Logical Reversibility of Computation. *IBM Journal of Research and Development*, **17**, 525-532. https://doi.org/10.1147/rd.176.0525

[28] Landauer, R. (1961) Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development*, **5**, 183-191. https://doi.org/10.1147/rd.53.0183

[29] Abramson, N. (1963) Information Theory and Coding. McGraw-Hill.

[30] Adamék, J.J. (1991) Foundations of Coding. Wiley.

[31] Akaiwa, Y. (1997) Introduction to Digital Mobile Communication. Wiley.

[32] Benedetto, S. and Montorsi, G. (1996) Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes. *IEEE Transactions on Information Theory*, **42**, 409-428. https://doi.org/10.1109/18.485713

[33] Berlekamp, E.R. (1968) Algebraic Coding Theory. McGraw-Hill.

[34] Bose, R.C. and Ray-Chaudhuri, D.K. (1960) On a Class of Error Correcting Binary Group Codes. *Information and Control*, **3**, 68-79. https://doi.org/10.1016/s0019-9958(60)90287-4

[35] Clark Jr. G.C. and Cain, J.B. (1981) Error-Correction Coding for Digital Communications. Plenum Publishers.

[36] Cover, T.M. and Thomas, J.A. (1991) Elements of Information Theory. Wiley.

[37] Divsalar, D. (1996) Turbo Codes. MILCOM Tutorial.

[38] Elias, P. (1955) Coding for Noisy Channels. IRE Convention Record, Part 4, 37-46.

[39] Forney, G.D. (1973) The Viterbi Algorithm. *Proceedings of the IEEE*, **61**, 268-278. https://doi.org/10.1109/proc.1973.9030

[40] Gabor, D. (1946) Theory of Communication. Part 3: Frequency Compression and Expansion. *Journal of the Institution of Electrical Engineers-Part III*: *Radio and Communication Engineering*, **93**, 445-457. https://doi.org/10.1049/ji-3-2.1946.0076

[41] Gallager, R.G. (1963) Low-Density Parity-Check Codes. The MIT Press. https://doi.org/10.7551/mitpress/4347.001.0001

[42] Golay, M. (1954) Binary Coding. *Transactions of the IRE Professional Group on Information Theory*, **4**, 23-28. https://doi.org/10.1109/tit.1954.1057463

[43] Golomb, S.W. (1967) Shift Register Sequences. Holden-Day.

[44] Hamming, R.W. (1950) Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, **29**, 147-160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x

[45] Haykin, S. (2001) Communication Systems. 4th Edition, John Wiley & Sons.

[46] Huffman, D. (1952) A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, **40**, 1098-1101. https://doi.org/10.1109/jrproc.1952.273898

[47] Lathi, B.P. (1995) Modern Digital and Analog Communication Systems. 2nd Edition, Oxford University Press.

[48] MacWilliams, F.J. and Sloane, N.J.A. (1977) The Theory of Error-Correcting Codes. North Holland.

[49] Michelson, A.M. and Levesque, A.H. (1985) Error-Control Techniques for Digital Communication. Wiley.

[50] Peterson, W.W. and Weldon Jr., E.J. (1972) Error Correcting Codes. 2nd Edition, M.I.T. Press.

[51] Rappaport, T.S. (1996) Wireless Communications: Principles and Practice. IEEE Press.

[52] Reed, I.S. and Solomon, G. (1960) Polynomial Codes over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, **8**, 300-304. https://doi.org/10.1137/0108018

[53] Schlegel, C. (1997) Trellis Coding. IEEE Press.

[54] Shannon, C.E. (1948) A Mathematical Theory of Communication. *Bell System Technical Journal*, **27**, 623-656. https://doi.org/10.1002/j.1538-7305.1948.tb00917.x

[55] Shannon, C.E. and Weaver, W. (1949) The Mathematical Theory of Communication. University of Illinois Press.

[56] Viterbi, A. (1967) Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, **13**, 260-269. https://doi.org/10.1109/tit.1967.1054010

[57] Viterbi, A.J. and Omura, J.K. (1979) Principles of Digital Communication and Coding. McGraw-Hill.

[58] Akers, S.B. (1972) A Rectangular Logic Array. *IEEE Transactions on Computers*, **21**, 848-857. https://doi.org/10.1109/tc.1972.5009040

[59] Kohavi, Z. (1978) Switching and Finite Automata Theory. McGraw-Hill.

[60] Al-Rabadi, A. (2005) Three-Dimensional Lattice Logic Circuits, Part I: Fundamentals. *Facta Universitatis-Series*: *Electronics and Energetics*, **18**, 1-13. https://doi.org/10.2298/fuee0501001a

[61] Al-Rabadi, A. (2005) Three-Dimensional Lattice Logic Circuits, Part II: Formal Methods. *Facta universitatis-Series*: *Electronics and Energetics*, **18**, 15-28. https://doi.org/10.2298/fuee0501015a

[62] Al-Rabadi, A. (2005) Three-Dimensional Lattice Logic Circuits, Part III: Solving 3D Volume Congestion Problem. *Facta universitatis-Series*: *Electronics and Energetics*, **18**, 29-43. https://doi.org/10.2298/fuee0501029a

[63] Green, D.H. (1991) Families of Reed-Muller Canonical Forms. *International Journal of Electronics*, **70**, 259-280. https://doi.org/10.1080/00207219108921277

[64] Bryant, R.E. (1986) Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, **35**, 677-691. https://doi.org/10.1109/tc.1986.1676819

[65] Karpovsky, M.G. (1976) Finite Orthogonal Series in the Design of Digital Devices.

Wiley.

[66] Forbes, R.G. (2012) Extraction of Emission Parameters for Large-Area Field Emitters, Using a Technically Complete Fowler-Nordheim-Type Equation. *Nanotechnology*, **23**, Article 095706. https://doi.org/10.1088/0957-4484/23/9/095706

[67] Fowler, R.H. and Nordheim, L.W. (1928) Electron Emission in Intense Electric Fields. *Proceedings of the Royal Society A*: *Mathematical*, *Physical and Engineering Sciences*, **119**, 137-181.

[68] Mano, M.M. and Kime, C.R. (2008) Logic and Computer Design Fundamentals. 4th Edition, Prentice-Hall.