FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 22, no. 1, April 2009, 1-33

Closed-System Quantum Logic Network Implementation of the Viterbi Algorithm

Anas N. Al-Rabadi

Abstract: New convolution-based multiple-stream error-control coding and decoding schemes are introduced. The new coding method applies the reversibility property in the convolution-based encoder for multiple-stream error-control encoding and implements the reversibility property in the new reversible Viterbi decoding algorithm for multiple-stream error-correction decoding. The complete design of quantum circuits for the quantum realization of the new quantum Viterbi cell in the quantum domain is also introduced. In quantum mechanics, a closed system is an isolated system that can't exchange energy or matter with its surroundings and doesn't interact with other quantum systems. In contrast to open quantum systems, closed quantum systems obey the unitary evolution and thus they are reversible. Reversibility property in error-control coding can be important for the following main reasons: (1) reversibility is a basic requirement for low-power circuit design in future technologies such as in quantum computing (QC), (2) reversibility leads to super-speedy encoding/decoding operations because of the superposition and entanglement properties that emerge in the quantum computing systems that are naturally reversible and therefore very high performance is obtained, and (3) it is shown in this paper that the reversibility 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 irreversible Viterbi algorithm.

Keywords: Error-Correcting Codes, Error-Control Coding, Coding, Low-Power Computing, Low-Power Circuits and Systems, Noise, Quantum Circuits, Quantum Computing, Reversible Circuits, Reversible Logic.

Manuscript received on October 12, 2008.

The author is currently an Associate Professor with the Computer Engineering Department, The University of Jordan, Jordan & is associated with the Office of Graduate Studies and Research (OGSR), Portland State University, U.S.A. (e-mail: alrabadi@yahoo.com).

1 Introduction

Due to the anticipated failure of Moore's law around the year 2020, quantum computing (QC) will play an increasingly crucial role in building more compact and less power consuming computers [4, 66, 67]. Due to this fact, and because all quantum computer gates (i.e., building blocks) should be reversible [4, 11, 48, 67, 72, 77, 92], reversible computing will have an increasingly more existence in the future design of regular, compact, and universal circuits and systems. (k,k) reversible 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 be always uniquely reconstructed from the vector of output states. A (k,k) conservative circuit has the same number of inputs k and outputs k and has the same number of values (states) in inputs and outputs (e.g., the same number of ones and twos in inputs and outputs for ternary) [4, 67, 72]. The importance of the conservativeness property stems from the fact that this property reflects the physical law of energy preservation: no energy can be created or destroyed, but can be transformed from one form to another. Thus, conservative logic will incorporate the fundamental law of energy preservation into the logic design of circuits and systems.

Other motivations for pursuing the possibility of implementing circuits and systems using reversible logic (RL) and QC would include items such as: (1) *power*: the fact that, theoretically, the internal computations in RL systems consume no power. It is shown in [48] that the amount of energy (heat) dissipated for every irreversible bit operation is given by $K \times T \ln(2)$ where $K = 1.3806505 \times 10^{-23}$ JK^{-1} 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 reversible logical components. Thus, reversible logic circuits are information-lossless. For this reason, different technologies have been studied to implement reversible logic in hardware such as in [4, 66, 67, 72, 77, 92]: bioinformatics, nanotechnology-based circuits and systems, adiabatic CMOS VLSI circuit design, optical systems, and quantum circuits. Fully reversible digital systems will greatly reduce the power consumption (theoretically eliminate) through three conditions: (i) *logical reversibility*: the vector of input states can always be uniquely reconstructed from the vector of output states, (ii) *physical reversibility*: the physical switch operates backwards as well as forwards, and (iii) the use of *"ideal-like" switches* that have no parasitic resistances; (2) *size*: since the newly emerging quantum computing technology must be reversible [4, 11, 48, 66, 67, 92], the current trends related to more dense hardware implementations are heading towards 1 Angstrom (atomic size), at which quantum mechanical effects have to be accounted for; and (3) *speed (performance)*: if the properties of superposition and entanglement of quantum mechanics can be usefully employed in the design of circuits and systems, significant computational speed enhancements can be expected [4, 67].

Therefore, while in the classical (irreversible) systems the frequency-to-power ratio (f/p), or equivalently power-to-frequency ratio (p/f), doesn't improve much after certain threshold (level) since the increase in frequency (i.e., more speed; better performance) leads to the increase in power consumption, this doesn't exist in the quantum domain; in the quantum system, speed of processing is very high (due to the properties of quantum superposition and entanglement) and power consumption is inversely very low, i.e., $(f/p) \to \infty$ or equivalently $(p/f) \to 0$.

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 reversible error correction of communicated batch of data (i.e., parallel data streams) are highly important tasks in situations where noise occurs. 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* [30, 31] 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 [1–3, 5–10, 12–18, 20–47, 49–65, 69–71, 74–76, 78–91, 93–96, 98–101].

For example, the manufacturers of integrated circuits (ICs) have recently started to produce error-correcting circuits, and one such circuit is the TI 74LS636 [19] which is an 8-bit error detection and correction circuit that corrects any single-bit memory read error and flags any two-bit error which is called single error correction / double error detection (SECDED). This IC is currently found in high-end computer systems because of the cost of implementing a system that uses error correction, and the newest computer systems are now using DDR memory with error-correction code (ECC). When a single error is detected, the 74LS636 goes through an error-correction cycle; the 74LS636 checks the single-error flag (SEF) to determine whether an error has occurred, and if it has then a correction cycle causes the single-error defect to be corrected, and if a double-error occurs then an interrupt request is generated by the double-error flag (DEF) output. Since the introduction of the Intel Pentium® microprocessor, the modern microprocessor design incorporates the logic circuitry to detect/correct errors provided that the memory can store the extra eight bits required for storing the ECC code, in which the ECC memory is 72-bits wide using the eight additional bits to store the ECC code (i.e., memory width is 64 data bits + 8 bits for ECC code), and if an error occurs then the microprocessor runs the correction cycle to correct the error. Recently, some memory devices such as Samsung® memory also perform an internal error check in which Samsung® ECC uses three bytes to check every 256 bytes of memory.

The main contributions of this paper are the introduction of new convolution-based multiple-stream error-control encoding and decoding schemes that apply the reversibility property in both the convolution-based encoder for multiple-stream error-control encoding and in the new reversible Viterbi decoding algorithm for multiple-stream error-control decoding. Also, the complete design of quantum circuits for the quantum implementation of the new quantum Viterbi cell (i.e., quantum trellis node) in the quantum domain is introduced. It is also introduced in this paper that the reversibility relationship between multiple-streams of parallel 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 (or more) that are uncorrectable using the irreversible Viterbi algorithm.

Basic background in error-control coding, reversible logic and quantum computing is presented in Section 2. The new reversible error correction method in data communication is introduced in Section 3. The design of quantum circuits for the quantum implementation of the new quantum Viterbi cell is introduced in Section 4. Conclusions and future work are presented in Section 5.

2 Fundamentals

This Section presents basic background in the topics of error-correction coding, reversible logic, and quantum computing. The fundamentals presented in this section will be utilized in the development of the new results introduced in Sections $3-4$.

2.1 Error correction

In the data communication context, noise usually exists and is generated from the channel in which transmitted data are 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

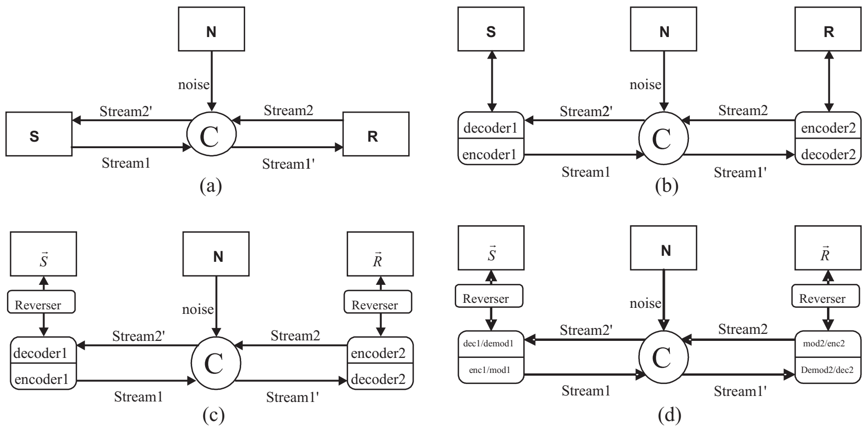


Fig. 1. Modeling data communication in the existence of noise: (a) model of a noisy data communication where C is the channel, (b) model of the solution to the noise problem using encoder / decoder schemes, (c) the application of reversibility using the reverser block for parallel multiple-input multiple-output (MIMO) bijectivity (uniqueness) in the data streams, and (d) the communication model in which coding and modulation are combined. In this model, stream1' = stream1 + noise and stream2' = stream2 + noise.

counterpart, noise must be modeled [22, 68, 97] and accordingly an appropriate encoding / decoding communication schemes must be implemented [1–3, 5–10, 12–18, 20–47, 49–65, 69–71, 74–76, 78–91, 93–96, 98–101]. Various coding schemes have been proposed and one very important family is the convolutional codes [1–3, 5, 12, 17, 21, 26, 32, 35, 37, 39, 44, 47, 49, 50, 54, 56, 60, 61, 63, 71, 74, 79, 87, 89, 91, 93, 94, 96, 98, 99]. Figure 1 illustrates the modeling of data communication in the existence of noise, the solution to the noise problem using an encoder / decoder scheme, and the utilization of a new block called the reverser for bijectivity (uniqueness) in multiple-stream (i.e., parallel data) communication.

Each of the two nodes sides in the system shown in Figure 1 consists of three major parts: (1) encoding (e.g., generating a convolutional code using a convolutional encoder) to generate an encoded transmitted decision (message), (2) channel noise, and (3) decoding (e.g., generating the correct convolution code using the corresponding decoding algorithm (cf. Viterbi algorithm)) 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 (FSM) 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 [1–3, 5, 12, 17, 21, 26, 32, 35, 37, 39, 44, 47, 49, 50, 54, 56, 60, 61, 63, 71, 74, 79, 87, 89, 91, 93, 94, 96, 98, 99].

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:

$$r = \frac{L}{n(L+M)} \quad \text{bits / symbol}$$

and for the typical case of $L \gg M$, the code rate reduces to $r \approx (1/n)$ 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 $r \approx (k/n)$ by using k shift registers, n modulo-2 adders, an input multiplexer, and an output multiplexer. An example of a convolutional encoder with constraint length = 3 and rate = $^1/_2$ is the one shown in Figure 2.

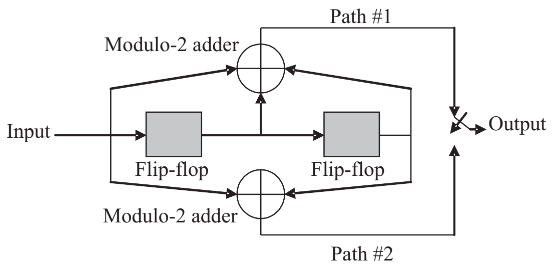


Fig. 2. Convolutional encoder with constraint length = 3 and rate = $^1/_2$. The flip-flop is a unit-delay element, and the modulo-2 adder is the logic Boolean difference (XOR) operation.

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:

$$\begin{aligned} g_1(D) &= 1\cdot D^0 + 1\cdot D^1 + 1\cdot D^2 \\ &= 1 + D + D^2 \\ g_2(D) &= 1\cdot D^0 + 0\cdot D^1 + 1\cdot D^2 \\ &= 1 + D^2 \end{aligned}$$

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

$$\begin{aligned} m(D) &= 1\cdot D^0 + 0\cdot D^1 + 0\cdot D^2 + 1\cdot D^3 + 1\cdot D^4 \\ &= 1 + D^3 + D^4 \end{aligned}$$

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:

$$\begin{aligned} c_1(D) = g_1(D)m(D) &= (1 + D + D^2)(1 + D^3 + D^4) \\ &= 1 + D + D^2 + D^3 + D^6 \\ c_2(D) = g_2(D)m(D) &= (1 + D^2)(1 + D^3 + D^4) \\ &= 1 + D^2 + D^3 + D^4 + D^5 + D^6 \end{aligned}$$

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:

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

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 3a), (4) trellis (cf. Figure 3b), and (5) state diagram (cf. Figure 3c).

An important decoder that uses the trellis representation to correct received erroneous messages is the Viterbi decoding algorithm [26,89–91]. 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 (HMMs) [73]. The Viterbi algorithm forms a subset of information theory [1,22], 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 network (LAN) communications.

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

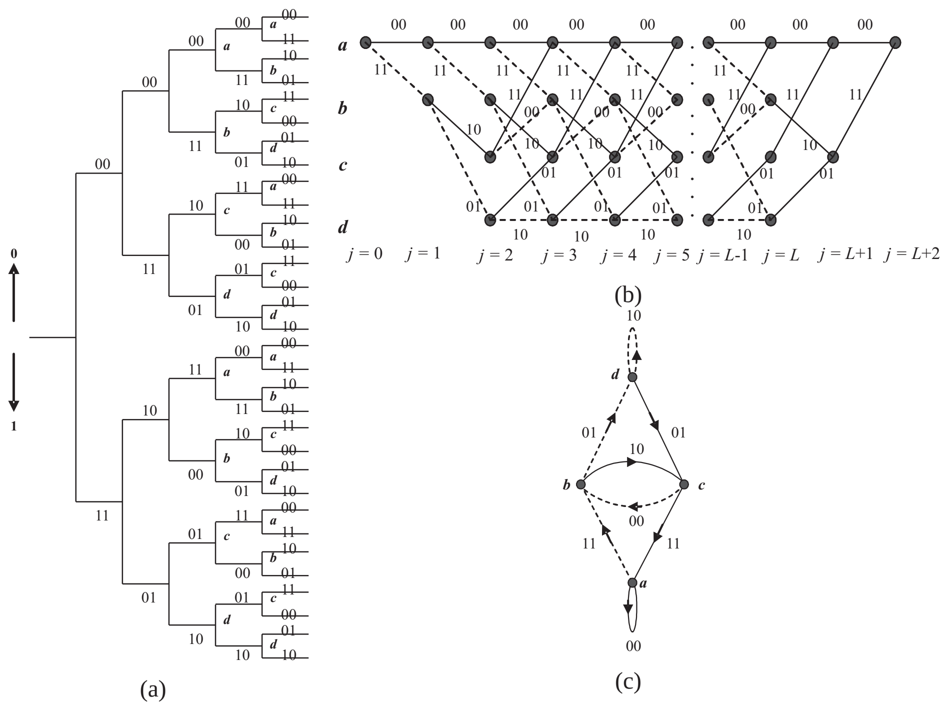


Fig. 3. Various representations for the circuit of the convolutional encoder in Figure 2: (a) code tree, (b) trellis, and (c) state diagram. Solid line is the input of value "0" and the dashed line is the input of value "1". The binary label on each branch is the encoder's output as it moves from one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

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 $M \leq j \leq L$, where $M = (K-1)$ is the encoders 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 (i.e., flipping a fair coin). The Viterbi algorithm is a maximum likelihood sequence estimator, and the following procedure and Examples 3 - 5 illustrate the detailed steps for the implementation of this algorithm [1, 2, 5, 12, 17, 21, 26, 39, 40, 44, 54, 60, 61, 63, 71, 89–91, 96].

Algorithm 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 [39] 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:

$$\begin{aligned} g_1(D) &= 1 \cdot D^0 + 1 \cdot D^1 + 1 \cdot D^2 \\ &= 1 + D + D^2 \\ g_2(D) &= 1 \cdot D^0 + 0 \cdot D^1 + 1 \cdot D^2 \\ &= 1 + D^2 \end{aligned}$$

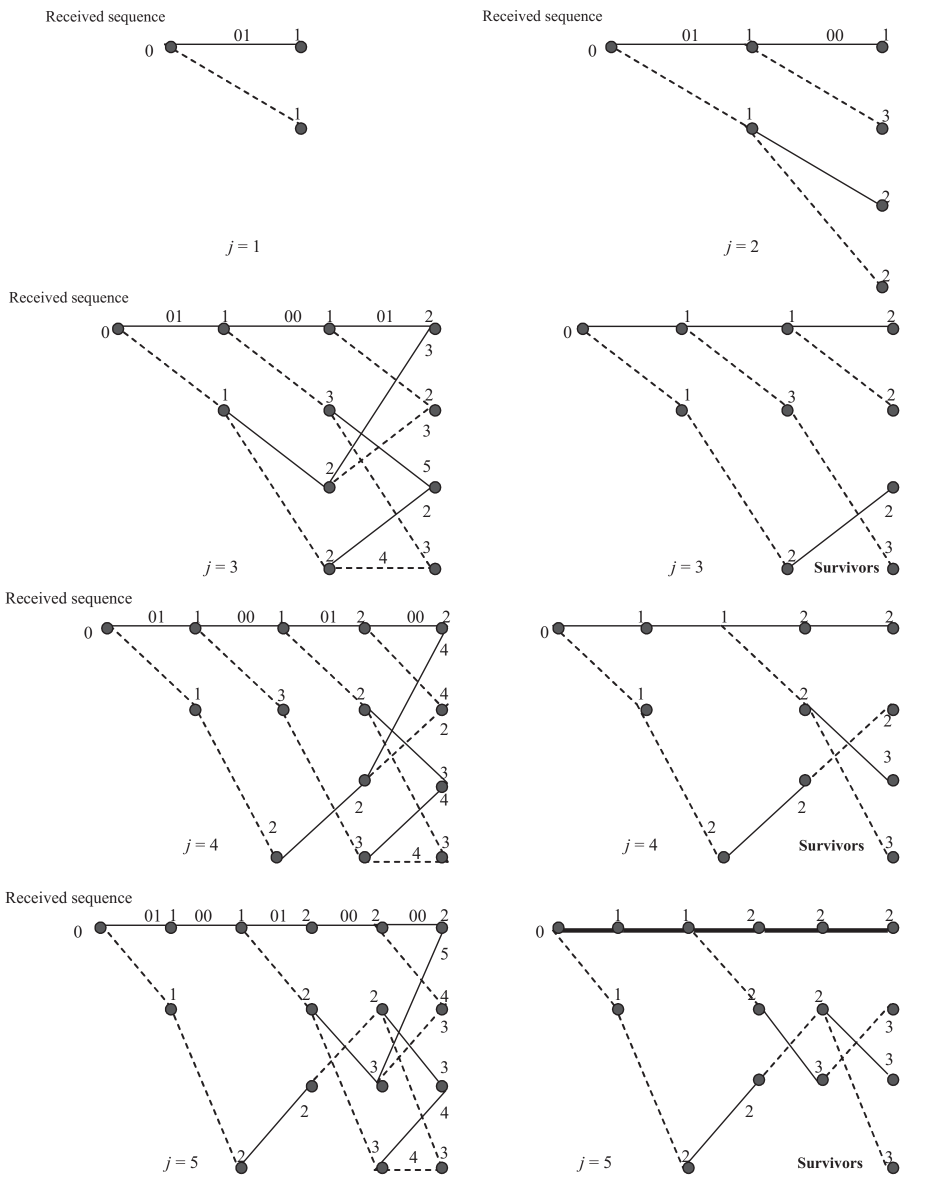


Fig. 4. The illustration of the steps of the Viterbi algorithm when applied for Example 3, where the bold path (in level $j = 5$) is the survivor path.

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

$$\begin{aligned} m(D) &= 1 \cdot D^0 + 0 \cdot D^1 + 1 \cdot D^2 \\ &= 1 + D^2 \end{aligned}$$

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:

$$\begin{aligned} c_1(D) &= g_1(D)m(D) = (1+D+D^2)(1+D^2) \\ &= 1+D+D^3+D^4 \\ c_2(D) &= g_2(D)m(D) = (1+D^2)(1+D^2) \\ &= 1+D^4 \end{aligned}$$

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, the following is the resulting survivor path which generates the correct sent message $c = (11, 10, 00, 10, 11)$.

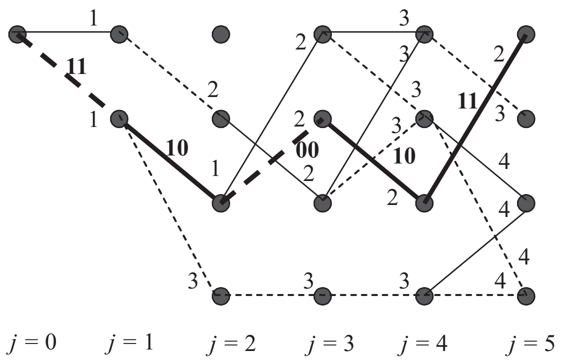


Fig. 5. The resulting survivors of the Viterbi algorithm when applied for Example 4, where the bold path is the survivor path.

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 $r = {}^{1}/_{2}$ 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 $> K$.

Example 5. Suppose an all-zero sequence $c = (0000000000)$ is generated by the convolutional encoder in Figure 2. For a received sequence containing three errors $c' = (1100010000)$, Figure 6 shows the breakdown of the Viterbi algorithm when implemented to the convolutional encoder in Figure 2 ($K = 3$ and $r = {}^{1}/_{2}$) as it fails to correct for a triple-error pattern.

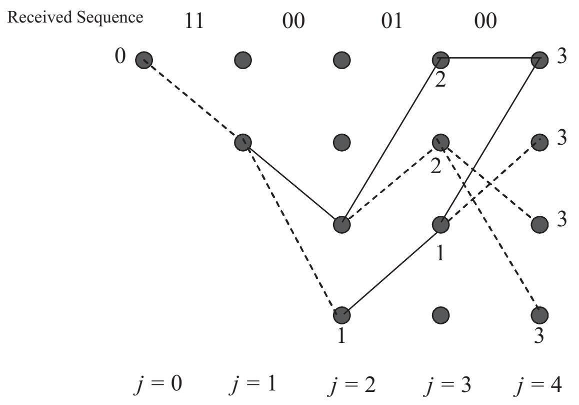


Fig. 6. The illustration of the failure of the Viterbi algorithm in Example 5, where the correct path has been eliminated in level $j = 3$.

2.2 Reversible logic

In quantum mechanical systems, a closed system is an isolated system that doesn't exchange energy or matter with its surroundings (i.e., doesn't dissipate power) and doesn't interact with other quantum systems. Closed quantum systems obey the unitary evolution and therefore they are reversible.

In general, an (n,k) reversible 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 [4,11,48,66,67,72,77,92]. Thus, a (k,k) reversible map is a bijective function which is both (1) injective (one-to-one or (1:1)) and (2) 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 reversibility are called garbage outputs. These are auxiliary outputs from which a reversible map is constructed (cf. Example 6). Therefore, reversible circuits (systems) are information-lossless.

Geometrically, achieving reversibility leads to value space-partitioning that leads to spatial partitions of unique values. Algebraically and in terms of systems representation, reversibility leads to multi-input multi-output (MIMO) bijective maps (i.e., bijective functions). An algorithm called reversible Boolean function (RevBF) that produces a reversible form from an irreversible Boolean function is as follows [4].

Algorithm RevBF

1. To achieve (k,k) reversibility, 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 1a, column W_1).
3. For the next auxiliary output, **If** non-reversibility still exists, **Then** assign for identical output tuples (irreversible map entries) values which are half zeros and half ones, and then assign a constant for the remainder that are already reversible.
4. **Do** step 3 until all map entries are reversible.

Example 6. The standard two-variable Boolean equivalence (XNOR): $W = c \otimes d$ is irreversible. 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 RevBF algorithm, the following are four possible reversible two-variable Boolean maps for the XNOR function:

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

c	d	W	W_1
0	0	1	**0**
0	1	0	**0**
1	0	0	**1**
1	1	1	**1**

(a)

c	d	W	W_1
0	0	1	**1**
0	1	0	**1**
1	0	0	**0**
1	1	1	**0**

(b)

c	d	W	W_1
0	0	1	**0**
0	1	0	**1**
1	0	0	**0**
1	1	1	**1**

(c)

c	d	W	W_1
0	0	1	**1**
0	1	0	**0**
1	0	0	**1**
1	1	1	**0**

(d)

For example, using the RevBF algorithm, the construction of the reversible map in Table 1a is obtained as follows: since W is irreversible, assign auxiliary (“garbage”) 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 reversible. This gate is also called the inverted Feynman gate or inverted Controlled-NOT (inverted C-NOT) gate in which: $W_1 = c$ and $W = c \otimes d = (c \oplus d)'$ (cf. Feynman gate in Figure 11a.)

2.3 Quantum computing

Quantum computing (QC) is a method of computation that uses a closed-system dynamic process governed (for a single particle) by the Schröinger Equation (SE) [4, 67]. The single-particle one-dimensional time-dependent SE (TDSE) takes the following general form:

$$-\frac{(h/2\pi)^2}{2m}\frac{\partial^2|\psi\rangle}{\partial x^2} + V|\psi\rangle = i\frac{h}{2\pi}\frac{\partial|\psi\rangle}{\partial t} \tag{2}$$

or

$$H|\psi\rangle = i\hbar\frac{\partial|\psi\rangle}{\partial t} \tag{3}$$

where h is Planck constant (6.626×10^{-34} Js), $\hbar = h/(2\pi)$ is the reduced Planck constant, $V(x,t)$ is the potential, m is particle mass, i is the imaginary number, $|\psi(x,t)\rangle$ is the quantum state, H is the Hamiltonian operator ($H = -[(h/2\pi)^2/2m]\nabla^2 + V$), and ∇^2 is the Laplacian operator. While the above holds for all physical systems, in the quantum computing (QC) context, the time-independent SE (TISE) is normally used [4, 67]:

$$\nabla^2|\psi\rangle = \frac{2m}{\hbar^2}(V-E)|\psi\rangle \tag{4}$$

where the solution $|\psi\rangle$ is an expansion over orthogonal basis states $|\phi_i\rangle$ defined in Hilbert space H as follows:

$$|\psi\rangle = \sum_i c_i|\phi_i\rangle \tag{5}$$

where the coefficients c_i are called probability amplitudes, and $|c_i|^2$ is the probability that the quantum state $|\psi\rangle$ will collapse into the (eigen) state $|\phi_i\rangle$. The probability is equal to the inner product $|\langle\phi_i|\psi\rangle|^2$, with the unitary condition $\sum|c_i|^2 = 1$.

In QC, a linear and unitary operator $\mathscr{T}$ is used to transform an input vector of $\underline{\text{qu}}$antum $\underline{\text{bits}}$ (qubits) into an output vector of qubits [4, 67]. In two-valued QC, a qubit is a vector of bits defined as follows:

$$\text{qubit}_0 \equiv |0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \quad \text{qubit}_1 \equiv |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{6}$$

A two-valued quantum state $|\psi\rangle$ is a superposition of quantum basis states $|\phi_1\rangle$ such as those defined in Equation (6). Thus, for the orthonormal computational basis states $\{|0\rangle, |1\rangle\}$, one has the following quantum state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{7}$$

where $\alpha\alpha^* = |\alpha|^2 = p_0 \equiv$ the probability of having state $|\psi\rangle$ in state $|0\rangle$, $\beta\beta^* = |\beta|^2 = p_1 \equiv$ the probability of having state $\psi\rangle$ in state $|1\rangle$, and $|\alpha|^2 + |\beta|^2 = 1$. The calculation in QC for multiple systems (e.g., the equivalent of a register) follow the tensor product ($\otimes$) [4]. For example, given two states $|\psi_1\rangle$ and $|\psi_2\rangle$ one has the following QC:

$$\begin{aligned} |\psi_{12}\rangle &= |\psi_1\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \\ &= (\alpha_1|0\rangle + \beta_1|1\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle) \\ &= \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle \end{aligned} \tag{8}$$

A physical system, describable by the following equation [4, 67]:

$$|\psi\rangle = c_1|\text{Spinup}\rangle + c_2|\text{Spindown}\rangle \tag{9}$$

(e.g., the hydrogen atom), can be used to physically implement a two-valued QC. Another common alternative form of Equation (9) is:

$$|\psi\rangle = c_1\left|+\frac{1}{2}\right\rangle + c_2\left|-\frac{1}{2}\right\rangle \tag{10}$$

Many-valued QC (MVQC) can also be accomplished [4, 67]. For the three-valued QC, the *qubit* becomes a 3-dimensional vector *qudit* ($\underline{\text{qu}}$antum $\underline{\text{d}}$iscrete dig$\underline{\text{it}}$), and in general, for MVQC the qudit is of dimension many. For example, one has for 3-state QC (in Hilbert space **H**) the following qudits:

$$\text{qudit}_0 \equiv |0\rangle = \begin{bmatrix}1\\0\\0\end{bmatrix}, \quad \text{qudit}_1 \equiv |1\rangle = \begin{bmatrix}0\\1\\0\end{bmatrix}, \quad \text{qudit}_2 \equiv |2\rangle = \begin{bmatrix}0\\0\\1\end{bmatrix} \tag{11}$$

A three-valued quantum state is a superposition of three quantum orthonormal basis states (vectors). Thus, for the orthonormal computational basis states $\{|0\rangle, |1\rangle, |2\rangle\}$, one has the following quantum state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle \tag{12}$$

where $\alpha\alpha^* = |\alpha|^2 = p_0 \equiv$the probability of having state $|\psi\rangle$ in state $|0\rangle$, $\beta\beta^* = |\beta|^2 = p_1 \equiv$the probability of having state $|\psi\rangle$ in state $|1\rangle$, $\gamma\gamma^* = |\gamma|^2 = p_2 \equiv$the probability of having state $|\psi\rangle$ in state $|2\rangle$, and $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$.

In general, for an *n*-valued logic, a quantum state is a superposition of *n* quantum orthonormal basis states (vectors). Thus, for the orthonormal computational basis states $\{|0\rangle, |1\rangle, \ldots, |n-1\rangle\}$, one has the following quantum state:

$$|\psi\rangle = \sum_{k=0}^{n-1} c_k|q\rangle_k \tag{13}$$

where: $\sum_{k=0}^{n-1} c_k c_k^* = \sum_{k=0}^{n-1} |c_k|^2 = 1$.

The calculation in QC for many-valued multiple systems follow the tensor product in a manner similar to the one demonstrated for two-valued QC in Equation (8).

As stated previously, while an open quantum system does interact with its environment (i.e., its surroundings or bath) and thus dissipate power resulting in a non-unitary evolution, a closed quantum system is an isolated system that doesn't exchange energy or matter with its surroundings and therefore doesn't dissipate power resulting in a unitary evolution (i.e., unitary transformation or unitary matrix) and hence they are reversible. A physical system comprising trapped ions under multiple laser excitations can be used to reliably implement MVQC [66]. A physical system in which an atom (particle) is exposed to a specific potential field (function) $V(x)$ can also be used to implement MVQC (two-valued being a special case) [4, 67]. In such an implementation, the (resulting) *distinct energy states* are used as the orthonormal basis states. The latter is illustrated in Example 7 below which is an example of implementing MVQC by exposing a particle to a potential field V where the distinct energy states are used as the *orthonormal basis states*.

Example 7. We assume the following constraints: (1) spring potential $V(x) = (1/2)kx^2$, where m is a particle, $k = m\omega^2$ is spring constant, and ω is the angular frequency ($\omega = 2\pi\cdot$ frequency), and (2) boundary conditions. Also, assuming the solution of the TISE in Equation (4) for these constraints is of the following form (i.e., the Gaussian function):

$$\psi(x) = Ce^{-\alpha\frac{x^2}{2}}$$

where $\alpha = m\omega/\hbar$. The general solution for the wave function $|\psi\rangle$, (for a spring potential) is:

$$C = \left[\frac{\alpha}{\pi}\right]^{\frac{1}{4}} \frac{1}{\sqrt{2^n n!}} H_n(\sqrt{\alpha}x)$$

where $H_n(x)$ are the Hermite polynomials. This solution leads to the sequence of evenly spaced energy levels (eigenvalues) E_n characterized by a quantum number n as follows:

$$E_n = (n + \frac{1}{2})\hbar\omega$$

The distribution of the energy states (eigenvalues) and their associated probabilities are shown in Figure 7.

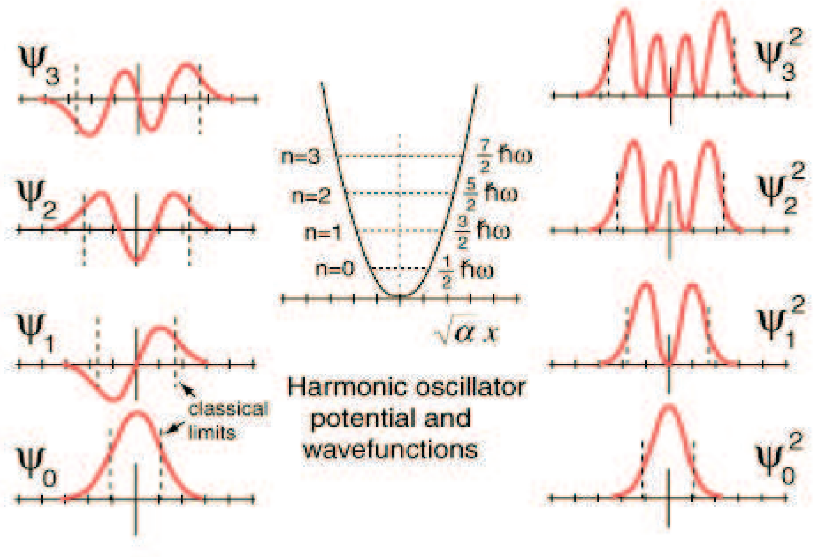


Fig. 7. Harmonic oscillator (HO) potential and wavefunctions: (a) wavefunctions for various energy levels (subscripts), (b) spring potential $V(x)$ and the associated energy levels E_n, and (c) probabilities for measuring particle m in each energy state (E_n).

A closed-system quantum circuit is a composition of quantum gates with the following properties [4, 67]: (1) must be reversible, (2) must have an equal number of inputs k and outputs k, (3) doesn't allow fan-out, (4) is constrained to be acyclic (i.e., feedback (loop) is not allowed), and (5) the transformation performed is unitary (i.e., a unitary matrix). The quantum Viterbi circuit design in the quantum domain using the corresponding basic quantum primitives will be completely shown in Section 4.

3 Reversible Error Correction via Reversible 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 introduces reversible error correction of communicated batch (parallel) of data in multiple-input multiple-output (MIMO) systems. Reversibility in parallel-based data communication is directly observed since:

$$\vec{O}_1 = \vec{I}_2 \tag{14}$$

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 irreversibility in data communication (i.e., irreversibility in data mapping) since $\vec{O}_1 \neq \vec{I}_2$. As will be introduced in this and the following sections respectively, the implementation of reversible error correction can be performed (1) in software using the new reversible error-correction algorithm and (2) in hardware using quantum error correction hardware. The following algorithm, called Reversible Viterbi (RV) Algorithm, introduces the implementation of reversible error correction in the parallel data communication.

Algorithm RV

```
1. Use the RevBF 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 re-
   ceived erroneous message.
5. Generate the total maximum-likelihood trellis resulting from the iterative
   application of the Viterbi decoding algorithm.
6. Generate the corrected communicated batch of data messages.
7. End
```

The convolutional encoding for the RV 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 8 in which several s convolutional encoders operate in parallel for encoding s number of simultaneously

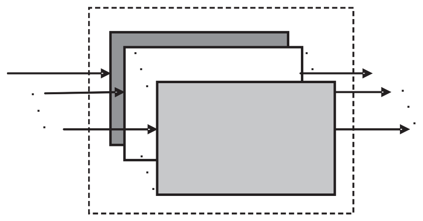


Fig. 8. General MIMO encoder circuit for the parallel generation of convolutional codes where each box represents a single SISO convolutional encoder such as the one shown in Figure 2.

submitted messages (i.e., data message set of cardinality (size) equal to s) generated from s nodes.

Example 8. The reversibility implementation (e.g., RevBF Algorithm) upon the following input bit stream $\{m_1 = 1, m_2 = 1, m_3 = 1\}$ produces the following reversible set of message sequences:

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

For the convolutional encoder in Figure 8, 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, respectively:

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

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 RV algorithm, Figure 9 shows the resulting survivor paths which generate the correct sent messages: $\{c_1 = (1110001011),\ c_2 = (0000111011),\ c_3 = (0011010111)\}$.

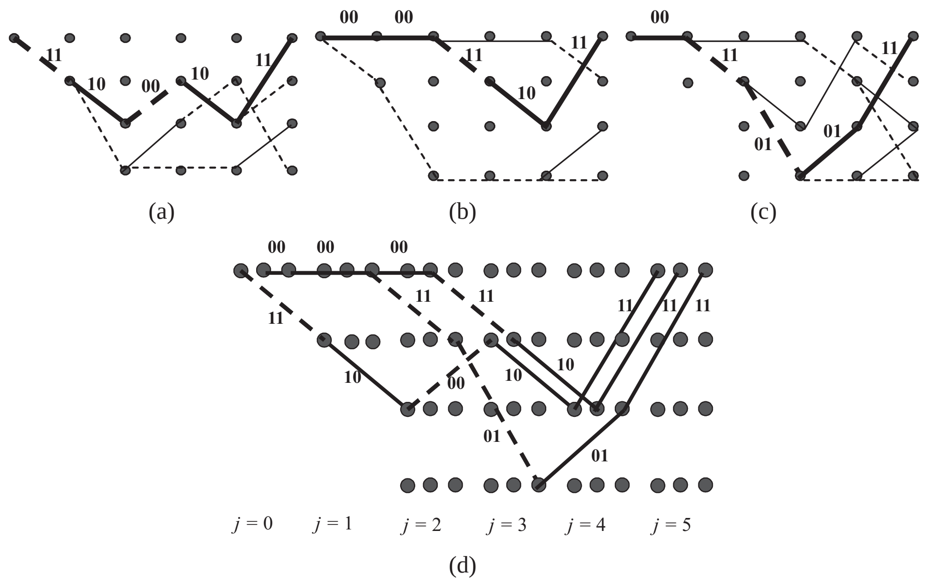


Fig. 9. The resulting survivor paths of the RV algorithm when applied to Example 8.

As in the irreversible Viterbi Algorithm, in some cases, applying the reversible Viterbi (RV) 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 reversible 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 $r = {}^{1}/_{2}$ and $K = 3$ cannot produce a correctable decoded message from the incoming erroneous 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 RV algorithm in case the RV 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) reversibility relationship between the parallel submitted data streams and this relationship exists from applying a known reversible mapping such as the RevBF algorithm, or (3) combination of parity and reversibility properties. The reversibility property in the RV algorithm produces a reversibility relationship between the sent parallel streams of data, and this known reversibility mapping can be used to correct the uncorrectable errors (e.g., triple errors) which the RV algorithm fails to correct.

Example 9. The following is a version of the RevBF algorithm that produces reversibility as follows:

Algorithm RevBF (Version 1)

```
1. To achieve $(k,k)$ reversibility, 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 mappings 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-reversibility still exists, Then assign
   for identical output tuples (irreversible map entries) values which are half
   ones and half zeros, and then assign a constant for the remainder that are
   already reversible which is the ones complement (NOT; inversion) of the
   previously assigned constant to that remainder.
4. Do step 3 until all map entries are reversible.
```

For the parallel sent bit stream $\{1,1,1\}$ in Example 8 in which the reversibility implementation (using Version 1 of the RevBF Algorithm) produces the following reversible 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 10 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)

Fig. 10. Tables for possible errors in data stream m_3 that is generated by the RevBF Algorithm V1: (a) original sent correct (uncorrupted) m_3 that resulted from the application of the RevBF Algorithm V1, and (b)–(h) possibilities of the erroneous received m_3.

Note that the erroneous m_3 is Figures 10b-10e and 10g-10h are correctable using the RV algorithm since less than triple-errors exits, but the triple error as in Figure 10f is (usually) uncorrectable using the RV algorithm. Yet, the existence of the reversibility property using the RevBF algorithm adds information that can be used to correct m_3 as follows: By applying the RevBF Algorithm (Version 1) from right-to-left in Figure 10f 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 RevBF 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 RevBF algorithm (Version 1) the 3rd cell of the last column from right must have the value “0” which is the ones 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.

4 Quantum Circuit Design of the New RV Algorithm

The reversible hardware implementation for each trellis node in the (reversible) Viterbi algorithm requires the following reversible components: reversible modulo-2 adder, reversible arithmetic adder, reversible subtractor (RS) and reversible selector (i.e., reversible multiplexer) to be both used in one possible design of the corresponding reversible comparator (RC). Table 2 shows the truth tables of an irreversible half-adder (HA), irreversible subtractor, and irreversible full-adder (FA).

Table 2. Truth tables: (a) irreversible half-adder (HA) and irreversible subtractor, and (b) irreversible full-adder (FA).

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

(a)

Inputs			Outputs	
a	b	c_i	s	c_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
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

(b)

While each quantum circuit is reversible, not each reversible circuit is quantum [4, 67]. Figure 11 shows the various quantum circuits for the quantum realization of each quantum trellis node in the corresponding (reversible) Viterbi algorithm. Figures 11a-11c present fundamental quantum gates [4, 67]. Figures

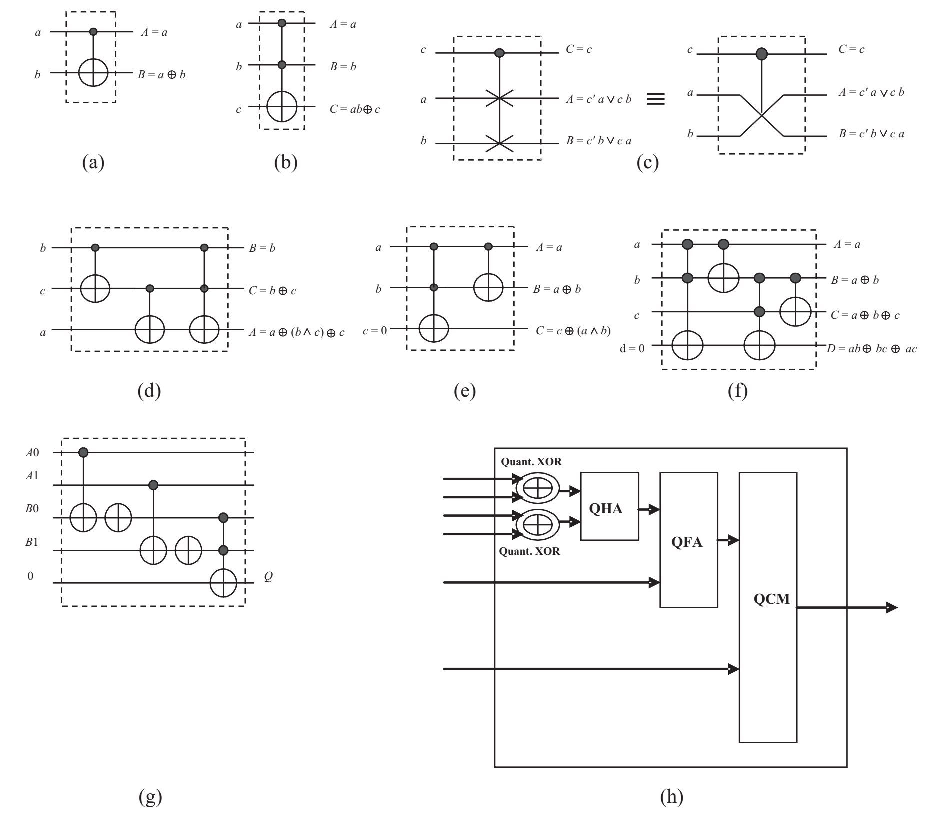


Fig. 11. Quantum reversible circuits for the quantum realization of each trellis node in the corresponding (reversible) Viterbi algorithm: (a) quantum XOR gate (Feynman gate; Controlled-NOT (C-NOT) gate), (b) quantum Toffoli gate (Controlled-Controlled-NOT (C^2-NOT) gate), (c) quantum multiplexer (Fredkin gate; Controlled-Swap (C-Swap) gate), (d) quantum subtractor, (e) quantum half-adder (QHA), (f) quantum full-adder (QFA), (g) quantum equality-based comparator that compares two 2-bit numbers where an isolated XOR symbol means a quantum NOT gate, and (h) basic quantum reversible Viterbi (QV) cell (i.e., quantum reversible trellis node) which is made of two Feynman gates, one QHA, one QFA and one quantum comparator with multiplexing (QCM). The quantum comparator can be synthesized using a quantum subtractor (QS) and a Fredkin gate. The symbol $\oplus$ is logic XOR (exclusive OR; modulo-2 addition), $\wedge$ is logic AND, $\vee$ is logic OR, and $'$ is logic NOT.

11d-11g show basic quantum arithmetic circuits of: quantum subtractor (Figure 11d), quantum half-adder (Figure 11e), quantum full-adder (Figure 11f), and the quantum equality-based comparator (Figure 11g) [4, 67]. Figure 11h introduces the basic quantum Viterbi cell (i.e., quantum trellis node) which is made of two Feynman gates, one QHA, one QFA and one quantum comparator with multiplexing (QCM).

Figure 12 shows the logic circuit design of an iterative network to compare two 3-digit binary numbers: $X = x_1 x_2 x_3$ and $Y = y_1 y_2 y_3$, and Figure 13 presents the detailed synthesis of a comparator circuit which is made of a comparator cell (Figure 13a) and a comparator output circuit (Figure 13b). The extension of the circuit in Figure 12 to compare two *n*-digit binary numbers is straightforward by utilizing *n*-cells and the same output circuit.

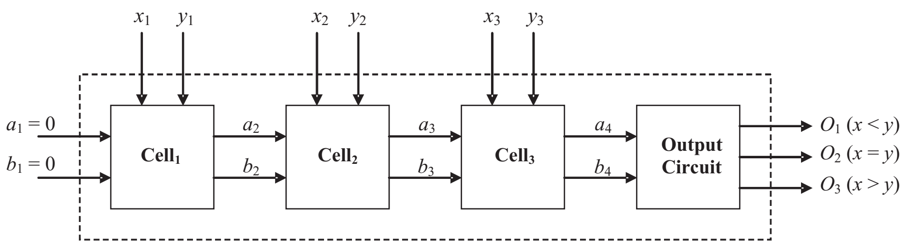


Fig. 12. An iterative network to compare two 3-digit binary numbers: $X = x_1x_2x_3$ and $Y = y_1y_2y_3$.

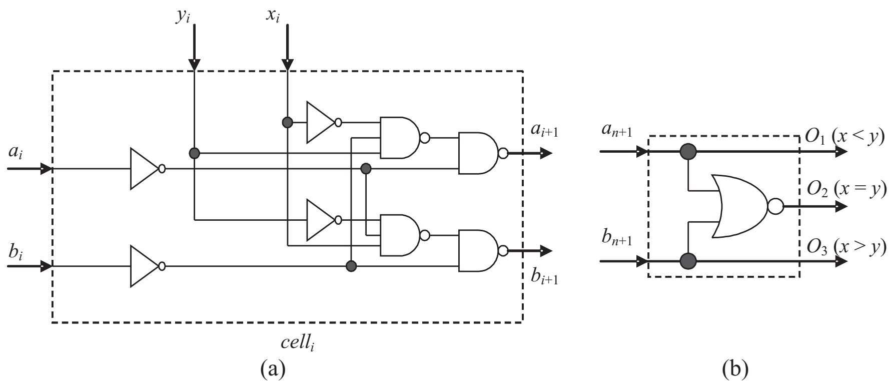
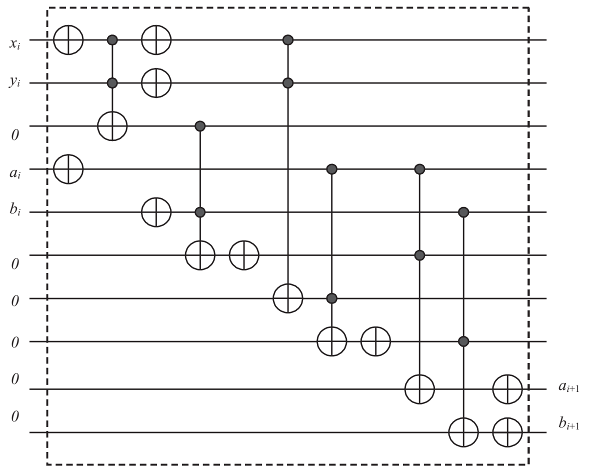
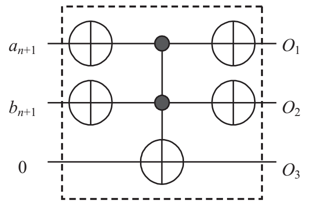


Fig. 13. Designing a comparator circuit: (a) comparator cell and (b) comparator output circuit.

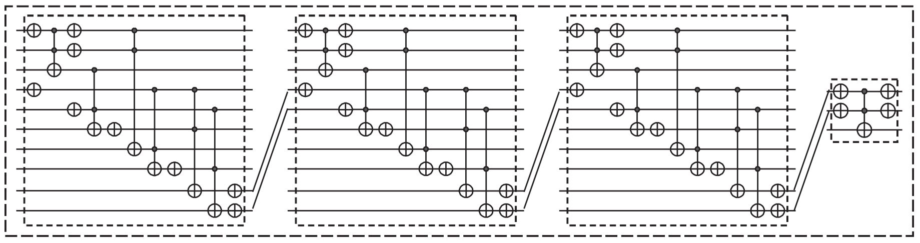
Figure 14 illustrates the quantum circuit synthesis for the comparator cell and the output circuit (which were shown in Figure 13), and Figure 15 shows the design of a quantum comparator with multiplexing (QCM) where Figure 15a shows an iterative quantum network to compare two 3-digit binary numbers and Figure 15c shows the complete design of the QCM. The extension of the quantum circuit in Figure 15a to compare two *n*-digit binary numbers is straightforward by utilizing *n* quantum cells (from Figure 14a) and the same output quantum circuit (in Figure 14b).



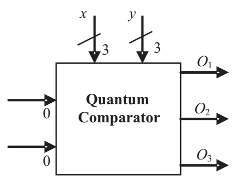


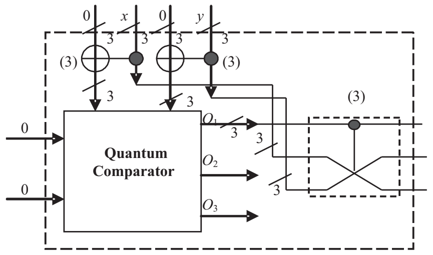
(a) (b)

Fig. 14. Quantum circuit synthesis for the comparator cell and output circuit in Figure 13: (a) quantum comparator cell and (b) quantum comparator output circuit.



(a)





(b) (c)

Fig. 15. Designing a quantum comparator with multiplexing (QCM): (a) an iterative quantum network to compare two 3-digit binary numbers, (b) symbol of the quantum comparator circuit in (a), and (c) complete design of QCM 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 16 shows the complete design of a quantum trellis node (i.e., quantum Viterbi cell) in the irreversible and reversible Viterbi algorithms that was shown in Figure 11h. The design of the quantum trellis node shown in Figure 16f pro-

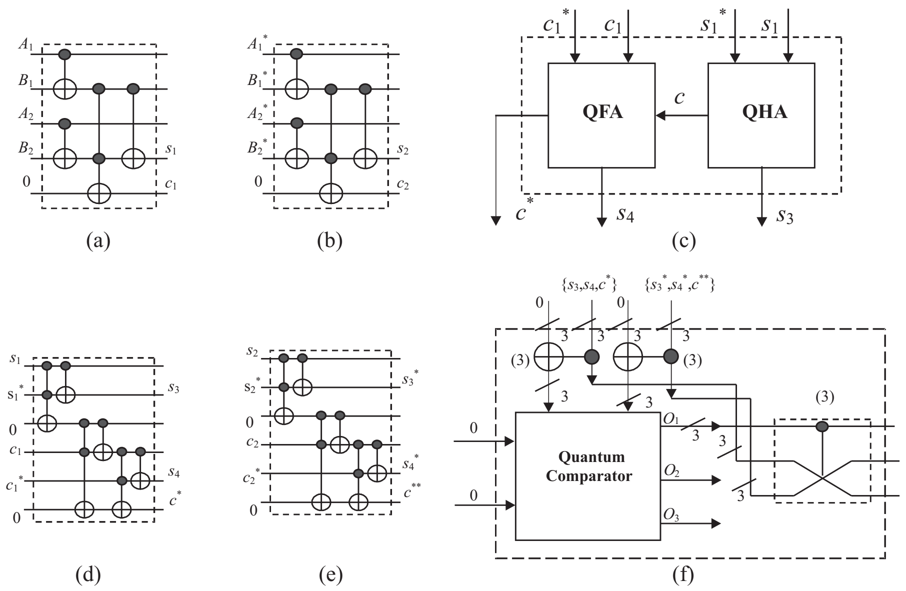


Fig. 16. The complete design of a quantum trellis node in the irreversible and reversible Viterbi algorithms that was shown in Figure 11h: (a) quantum circuit that is made of two Feynman gates (i.e., two quantum XORs) to produce the difference between incoming received bits (A_1A_2) and trellis bits (B_1B_2) followed by quantum half-adder (QHA) to produce the corresponding sum (s_1c_1) which is the Hamming distance for the first line entering the trellis node, (b) quantum circuit that is made of two Feynman gates (i.e., two quantum XORs) to produce the difference between incoming received bits ($A_1^*A_2^*$) and trellis bits ($B_1^*B_2^*$) followed by quantum half-adder (QHA) to produce the corresponding sum (s_2c_2) which is the Hamming distance for the second line entering the trellis node, (c) logic circuit composed of QHA and quantum full-adder (QFA) that adds the current Hamming distance to the previous Hamming distance, (d) quantum circuit in the first line entering the trellis node for the logic circuit in (c) that is made of a QHA followed by a QFA, (e) quantum circuit in the second line entering the trellis node for the logic circuit in (c) that is made of a QHA followed by a QFA, and (f) quantum comparator with multiplexing (QCM) in the trellis node that compares the two entering metric numbers: $X = s_3s_4c^*$ and $Y = s_3^*s_4^*c^{**}$ and selects using control line O_1 the path that produces the minimum entering metric (i.e., $X < Y$).

ceeds as follows: (1) two quantum circuits for the first and second lines entering the trellis node each is made of two Feynman gates (i.e., two quantum XORs) to produce the difference between incoming received bits and trellis bits followed by quantum half-adder (QHA) to produce the corresponding sum (which is the Hamming distance) are shown in Figures 16a and 16b, (2) logic circuit composed of a QHA and a quantum full-adder (QFA) that adds the current Hamming distance to the previous Hamming distance is shown in Figure 16c, (3) two quantum circuits for the first and second lines entering the trellis node each is synthesized according to the logic circuit in Figure 16c (which is made of a QHA followed by a QFA) are shown in Figures 16d and 16e, (4) quantum comparator with multiplexing (QCM) in the trellis node 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 16f.

In Figures 16c-16e, the current Hamming metric $\{s_1, c_1\}$ for the first entering path of the trellis node and the current Hamming metric $\{s_2, c_2\}$ for the second entering path of the trellis node 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 trellis node (e.g., $(5)_{10} = (101)_2$), then extra QFAs are added in the logic circuit in Figure 16c and consequently in the quantum circuits shown in Figures 16d-16e. Also, in the case that when the paths entering a quantum trellis node (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 quantum circuit in Figure 16f 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 ($\{s_3, s_4, c^*\} = \{s_3^*, s_4^*, c^{**}\}$).

5 Conclusions and Future Work

This paper introduces new convolution-based multiple-stream error-correction encoding and decoding methods that implement the reversibility property in the convolution-based encoder for multiple-stream error-control encoding and in the new reversible Viterbi (RV) decoding algorithm for multiple-stream error-control decoding. This paper also introduces the complete synthesis of quantum circuits in the quantum domain for the quantum implementation of the new quantum trellis node (i.e., quantum Viterbi cell). It is also shown in this paper that the relationship of reversibility 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 RV algorithm in correcting for more than two errors.

While an open quantum system interacts with its environment (i.e., its surroundings or bath) and thus dissipates power which result in a non-unitary evolution, a closed quantum system doesn’t exchange energy or matter with its surroundings and therefore doesn’t dissipate power which results in a unitary evolution (i.e., unitary matrix) and hence it is reversible. Since power reduction has become the current main concern for digital logic designers after performance (speed), reversibility property in error-control coding is highly important because reversibility is a main requirement for low-power circuit synthesis of future technologies such as in quantum computing, and reversibility property results in super-speedy encoding/decoding operations because of the superposition and entanglement properties that emerge in the closed quantum computing systems that are inherently reversible.

Future work will include items such as the investigation of using the introduced reversibility property in more advanced multi-error coding schemes to correct the corresponding corrupted multi-stream communicated data, and also the investigation of the corresponding optimal quantum circuit design of such new reversible systems.

References

[1] N. Abramson, *Information theory and coding*, McGraw-Hill, New York, 1963.

[2] J. J. Adamk, *Foundations of coding*, Wiley, New York, 1991.

[3] Y. Akaiwa, *Introduction to digital mobile communication*, Wiley, New York, 1997.

[4] A. N. Al-Rabadi, *Reversible logic synthesis: From fundamentals to quantum computing*, Springer-Verlag, New York, 2004.

[5] J. B. Anderson and S. Mohan, *Source and channel coding: An algorithmic approach*, Kluwer Academic, Boston, Mass., 1991.

[6] J. B. Anderson and D. P. Taylor, *A bandwidth-efficient class of signal space codes*, IEEE Transactions on Information Theory **IT-24** (1978), 703–712.

[7] B. S. Atal and M. R. Schroeder, *Stochastic coding of speech signals at very low bit rates*, IEEE International Conference on Communications, May 1984.

[8] L. R. Bahi, J. Cocke, F. Jelinek, and J. Raviv, *Optimal decoding of linear codes for minimizing symbol error rate*, IEEE Transactions on Information Theory **IT-20** (1974), 284–287.

[9] G. Battail, *Coding for the Gaussian channel: the promise of weighted output decoding*, International J. Satellite Communications **7** (1989), 183–192.

[10] S. Benedetto and G. Montorsi, *Unveiling turbo codes: Some results on parallel concatenated coding schemes*, IEEE Transactions on Information Theory **42** (1996), 409–428.

[11] C. Bennett, *Logical reversibility of computation*, IBM Journal of Research and Development **17** (1973), 525 – 532.

[12] E. R. Berlekamp, *Algebraic coding theory*, McGraw-Hill, New York, 1968.

[13] C. Berrou and A. Glavieux, *Near optimum error correcting coding and decoding: Turbo codes*, IEEE Transactions on Communications **44** (1996), 1261–1271.

[14] ______, *Reflections on the prize paper: Near optimum error-correcting coding and decoding turbo codes*, IEEE Information Theory Society Newsletter **48** (1998), no. 2, 24 – 31.

[15] C. Berrou, A. Glavieux, and P. Thitmajshima, *Near Shannon limit error-correction coding and decoding: Turbo codes*, International Conference on Communications (Geneva, Switzerland), May 1993, pp. 1064–1090.

[16] V. K. Bhargava, *Forward error correction schemes for digital communications*, IEEE Communications Magazine **21** (1983), no. 1, 11–19.

[17] V. K. Bhargava, D. Haccoun, R. Matyas, and P. Nuspl, *Digital communications by satellite: Modulation, multiple access, and coding*, Wiley, New York, 1981.

[18] R. C. Bose and D. K. Ray-Chaudhuri, *On a class of error correcting binary group codes*, Information and Control **3** (1960), 68–79.

[19] Barry B. Brey, *The intel microprocessors: Architecture, programming, and interfacing*, 7th ed., Pearson Education Inc., 2006.

[20] A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, *Speech coding based upon vector quantization*, IEEE Transactions on Acoustics, Speech, and Signal Processing **ASSP-28** (1980), 562–574.

[21] G. C. Clark, Jr. and J. B. Cain, *Error-correction coding for digital communications*, Plenum Publishers, New York, 1981.

[22] T. M. Cover and J. A. Thomas, *Elements of information theory*, Wiley, New York, 1991.

[23] F. Daneshgaran and M. Mondin, *Design of interleavers for turbo codes: Iterative interleaver growth algorithms of polynomial complexity*, IEEE Transactions on Information Theory **45** (1999), 1845–1859.

[24] D. Divsalar, *Turbo codes*, MILCOM tutorial (San Diego), November 1996.

[25] P. Elias, *Coding for noisy channels*, IRE Convention Record, Part 4 (1955), 37–46.

[26] G. D. Forney, Jr., *The Viterbi algorithm*, Proceedings of the IEEE **61** (1973), no. 3, 268–278.

[27] G. D. Forney, Jr. and M. V. Eyuboglu, *Combined equalization and coding using precoding*, IEEE Communications Magazine **29** (1991), no. 12, 25–34.

[28] B. J. Frey and D. J. C. MacKay, *Irregular turbocodes*, Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing (Allerton House, Illinois), September 1999.

[29] D. Gabor, *Theory of communication*, Journal of IEE **93** (1946), 429–457, Part III.

[30] R. G. Gallager, *Low-density parity-check codes*, IRE Transactions on Information Theory **8** (1962), 21–28.

[31] ______, *Low-density parity-check codes*, M.I.T. Press, Cambridge, Mass., 1963.

[32] R. D. Gitlin, J. F. Hayes, and S. B. Weinstein, *Data communications principles*, Plenum, New York, 1992.

[33] M. J. E. Golay, *Note on digital coding*, Proceedings of the IRE **37** (1949), 657.

[34] ______, *Binary coding*, IRE Transactions on Information Theory **PGIT-4** (1954), 23 – 28.

[35] S. W. Golomb, *Shift register sequences*, Holden-Day, San Francisco, 1967.

[36] D. W. Hagelbarger, *Recurrent codes: Easily mechanized burst-correcting binary codes*, Bell System Tech. J. **38** (1959), 969 – 984.

[37] J. Hagenauer, E. Offer, and L. Papke, *Iterative decoding of binary block and convolutional codes*, IEEE Transactions on Information Theory **42** (1996), 429 – 445.

[38] R. W. Hamming, *Error detecting and error correcting codes*, Bell System Tech. J. **29** (1950), 147 – 160.

[39] S. Haykin, *Communication systems*, 4th ed., John Wiley & Sons, 2001.

[40] C. Heegard and S. B. Wicker, *Turbo coding*, Kluwer Academic Publishers, Boston, Mass., 1999.

[41] D. A. Huffman, *A method for the construction of minimum redundancy codes*, Proceedings of the IRE **40** (1952), 1098 – 1101.

[42] I. M. Jacobs, *Practical applications of coding*, IEEE Transactions on Information Theory **IT- 20** (1974), 305 – 310.

[43] N. S. Jayant, *Coding speech at low bit rates*, IEEE Spectrum **23** (1986), no. 8, 58 – 63.

[44] N. S. Jayant and P. Noll, *Digital coding of waveforms: Principles and applications to speech and video*, Prentice-Hall, Englewood Cliffs, N. J., 1984.

[45] H. Kobayashi, *Correlative level coding and maximum-likelihood decoding*, IEEE Transactions on Information Theory **IT-17** (1971), 586 – 594.

[46] F. R. Kschischang and B. J. Frey, *Interactive decoding of compound codes by probability propagation in graphical models*, IEEE Journal on Selected Areas in Communication **16** (1998), 219 – 230.

[47] P. Lafrance, *Fundamental concepts in communication*, Prentice-Hall, Englewood Cliffs, N. J., 1990.

[48] R. Landauer, *Irreversibility and heat generation in the computational process*, IBM Journal of Research and Development **5** (1961), 183 – 191.

[49] B. P. Lathi, *Modern digital and analog communication systems*, 2nd ed., Oxford University Press, 1995.

[50] E. A. Lee and D. G. Messerschmitt, *Digital communication*, 2nd ed., Kluwer Academic Publishers, Boston, Mass., 1994.

[51] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kohn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, *A brief history of the internet*, Commun. ACM **40** (1997), 102 – 108.

[52] A. Lender, *Correlative level coding for binary-data transmission*, IEEE Spectrum **3** (1966), no. 2, 104 – 115.

[53] S. Lin, D. J. Costello, and M. J. Miller, *Automatic-repeat-request error control schemes*, IEEE Communications Magazine **22** (1984), no. 12, 5 – 16.

[54] S. Lin and D. J. Costello, Jr., *Error control coding: Fundamentals and applications*, Prentice-Hall, Englewood Cliffs, N. J., 1983.

[55] J. Lodge, R.Young, P. Hoeher, and J. Hagenauer, *Separable map 'filters' for the decoding of product and concatenated codes*, Proceedings of the IEEE International Conference on Communications (Geneva, Switzerland), May 1993, pp. 1740 – 1745.

[56] R. W. Lucky, J. Salz, and E. J. Weldon, Jr., *Principles of data communication*, McGraw-Hill, New York, 1968.

[57] D. J. C. MacKay, *Good error-correcting codes based on very sparse matrices*, IEEE Transactions on Information Theory **45** (1999), 399 – 431.

[58] D. J. C. MacKay and R. M. Neal, *Near Shannon limit performance of low density parity check codes*, Electronics Letters **33** (1997), no. 6, 457 – 458, and **32** (1996), no. 18, 1645 - 1646.

[59] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, *Comparison of constructions of irregular Gallager codes*, IEEE Transactions on Communications **47** (1999), 1449 – 1454.

[60] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*, North-Holland, Amsterdam, 1977.

[61] R. J. McEliece, *The theory of information and coding: A mathematical framework for communication*, Addison-Wesley, Reading, Mass., 1977.

[62] R. J. McEliece, D. J. C. MacKay, and J. F. Cheng, *Turbo decoding as an instance of pearl's "belief propagation" algorithm*, IEEE Journal on Selected Areas of Communication **16** (1998), no. 2, 140–152.

[63] A. M. Michelson and A. H. Levesque, *Error-control techniques for digital communication*, Wiley, New York, 1985.

[64] D. Middleton, *An introduction to statistical communication theory*, McGraw-Hill, New York, 1960.

[65] M. L. Moher and T. A. Gulliver, *Cross-entropy and iterative decoding*, IEEE Transactions on Information Theory **44** (1998), 3097 – 3104.

[66] A. Muthukrishnan and C. R. Stroud, *Multivalued logic gates for quantum computation*, Phy. Rev. A **62** (2000), 052309.

[67] M. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2000.

[68] A. Papoulis, *Probability, random variables, and stochastic processes*, 2nd ed., McGraw-Hill, New York, 1984.

[69] S. Pasupathy, *Correlative coding: a bandwidth-efficient signaling scheme*, IEEE Communications Magazine **15** (1977), no. 4, 4 – 11.

[70] A. J. Paulraj and C. B. Papadias, *Space-time processing for wireless communications*, IEEE Signal Processing Magazine (1997), 49 – 83.

[71] W. W. Peterson and E. J. Weldon, Jr., *Error correcting codes*, 2nd ed., M.I.T. Press, Cambridge, Mass., 1972.

[72] P. Picton, *Optoelectronic multi-valued conservative logic*, Int. J. of Optical Computing **2** (1991), 19 – 29.

[73] L. R. Rabiner, *A tutorial on hidden Markov models and selected applications in speech recognition*, Proceedings of the IEEE **77** (1989), no. 2, 257–286.

[74] T. S. Rappaport, *Wireless communications: Principles and practice*, IEEE Press, Piscataway, N. J., 1996.

[75] I. S. Reed and G. Solomon, *Polynomial codes over certain finite fields*, Journal of SIAM **8** (1960), 300 – 304.

[76] W. L. Root, *Remarks, mostly historical, on signal detection and signal parameter estimation*, Proceedings of the IEEE **75** (1987), 1446 – 1457.

[77] K. Roy and S. Prasad, *Low-power cmos vlsi circuit design*, John Wiley & Sons Inc., 2000.

[78] A. Ruiz, J. M. Cioffi, and S. Kasturia, *Discrete multiple tone modulation with coset coding for the spectrally shaped channel*, IEEE Transactions on Communications **40** (1992), 1012 – 1029.

[79] C. Schlegel, *Trellis coding*, IEEE Press, Piscataway, N. J., 1997.

[80] C. E. Shannon, *A mathematical theory of communication*, Bell System Tech. J. **27** (1948), 379 – 423, 623 – 656.

[81] ______, *Communication in the presence of noise*, Proceedings of the IRE **37** (1949), 10 – 21.

[82] ______, *Communication theory of secrecy systems*, Bell System Tech. J. **28** (1949), 656 – 715.

[83] C. E. Shannon and W. Weaver, *The mathematical theory of communication*, University of Illinois Press, Urbana, 1949.

[84] B. Sklar, *A primer on turbo code concepts*, IEEE Communications Magazine **35** (1997), 94 – 102.

[85] R. Steele, *Delta modulation systems*, Wiley, New York, 1975.

[86] A. S. Tanenbaum, *Computer networks*, 2nd ed., Prentice-Hall, Englewood Cliffs, N. J., 1995.

[87] T. M. Thompson, *From error-correcting codes through sphere packings to simple groups*, The Mathematical Association of America, Washington D. C., 1983.

[88] G. Ungerboeck, *Channel coding with multilevel/phase signals*, IEEE Transactions on Information Theory **IT-28** (1982), 55 – 67.

[89] A. J. Viterbi, *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*, IEEE Transactions on Information Theory **IT-13** (1967), 260 – 269.

[90] ______, *Wireless digital communication: A view based on three lessons learned*, IEEE Communications Magazine **29** (1991), no. 9, 33 – 36.

[91] A. J. Viterbi and J. K. Omura, *Principles of digital communication and coding*, McGraw-Hill, New York, 1979.

[92] A. De Vos, *Reversible computing*, Progress in Quantum Electronics **23** (1999), 1 – 49.

[93] L. F. Wei, *Rotationally invariant convolutional channel coding with expanded signal space - part I:* 180°, IEEE Journal on Selected Areas in Communications **SAC-2** (1984), 659 – 671.

[94] ______, *Rotationally invariant convolutional channel coding with expanded signal space - part II: Nonlinear codes*, IEEE Journal on Selected Areas in Communications **SAC-2** (1984), 672 – 686.

[95] S. B. Wicker and V. K. Bhargava (editors), *Reed-Solomon codes*, IEEE Press, Piscataway, N. J., 1994.

[96] S. G. Wilson, *Digital modulation and coding*, Prentice-Hall, Englewood Cliffs, N. J., 1996.

[97] R. D. Yates and D. J. Goodman, *Probability and stochastic processes: A friendly introduction for electrical and computer engineers*, Wiley, New York, 1999.

[98] J. H. Yuen (ed.), *Deep space telecommunications systems engineering*, Plenum, New York, 1983.

[99] R. E. Ziemer and W. H. Tranter, *Principles of communications*, 3rd ed., Houghton Miflin, Boston, Mass., 1990.

[100] J. Ziv and A. Lempel, *A universal algorithm for sequential data compression*, IEEE Transactions on Information Theory **IT-23** (1977), 337 – 343.

[101] ______, *Compression of individual sequences via variable-rate coding*, IEEE Transactions on Information Theory **IT-24** (1978), 530 – 536.