IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-5, Issue-2 E-ISSN: 2347-2693

PER/BER Performance Evaluation of Less-Complex KVD Decoding Architecture for IEEE 802.11 a/n/ac/ah WLANs

Dutta R.[1*], Konar S.C.[2] and Mitra K.[3]

[1*]Dept.of Electronics & Communication Engg., SIEM, Maulana Abul Kalam Azad University of Tech., Kolkata, India
[2] Dept. of Electrical Engineering, Indian Institute of Engg. Science & Technology (BESU), India
[3]Dept.of Electronics & Communication Engg., SIEM, Maulana Abul Kalam Azad University of Tech., Kolkata, India

[*]*Corresponding Author: ritam_siliguri@yahoo.com, Tel.: +91-94340-61896*

Available online at: www.ijcseonline.org

Received: 09/Jan/2017 Revised: 18/Jan/2017 Accepted: 04/Feb/2017 Published: 28/Feb/2017

***Abstract* - In modern era of wireless communications Viterbi decoder (VD) is widely used to decode Binary Convolution Codes (BCC) in many Wi-Fi systems such as WLAN 802.11a/n/ac/ah, Satellite communications, Mobile communications etc. To obtain high decoding performance, the constraint length of BCCs is basically quite long. Whereas the complexity of VD is exponentially affected by BCCs constraint length. Therefore Viterbi decoding of BCCs with long constraint length cannot be used for systems or devices like Internet of Things (IoT) sensors, that require low power and less hardware cost. In this work a less-complex K_{min} Viterbi Decoder (KVD) is proposed in which the decoder's complexity is inconsiderably affected by constraint length. For the decoding performance evaluation such as Packet Error Rate (PER) and Bit Error Rate (BER) of the proposed decoder, a BCC with constraint length k = 7 is used. This code is required in Wi-Fi systems such as IEEE 802.11a/n/ac/ah. A new standard 802.11ah for IoT applications is considered for simulation. The PER performance of proposed KVD decoder in relation with several factors such as channel type, modulation type, decoder's trace-back length and packet size has been evaluated. The proposed KVD achieves the same PER performance as the orthodox VD does, while its complexity is reduced by approximately 12.80 times to 21.33 times shown in result analysis.**

***Keywords*—k_{min} Viterbi Decoder (KVD); Binary Convolution Code (BCC); IoT sensors; Packet Error Rate (PER); Bit Error Rate (BER); Wireless Transceiver; 802.11 a/n/ac/ah WLAN**

I. INTRODUCTION

In today's world, many researchers are getting attracted by internet of things (IoT) from all over the world. It is the key technology to build smart and intelligent systems, smart city, etc. [1]. The terms IoT refers to a network of things or objects that are equipped with electronic devices or sensors, to collect information of things and exchange the information with the server computer. To exchange data, IoT sensors [2] must have a wireless communication transceiver which follows one of the standards such as Bluetooth, Zigbee, Wi-Fi 802.11ah, etc. Among these standards, 802.11ah is being developed by IEEE 802.11 committee. Because of inheriting many features from the successful Wi-Fi standards such as 802.11n/ac, the 802.11ah is expected to be widely used for developing wireless transceiver in future IoT sensors. Unlike the orthodox Wi-Fi transceivers, i.e., 802.11n/ac [3], which need high data rate that is not a big issue in IoT transceiver. However, complexity becomes a critical problem because of less area, low-cost and low power consumed IoT sensors. Meanwhile, as the purpose of transferring data, packet error rate (PER) performance of transceiver is very important. Once the packet is received with error, the transceiver should transfer that packet again with higher signal power. As a result, transfer time is longer and system consumes more energy. Thus improving PER performance means reducing transfer energy.

To improve the PER performance, beside of employing Orthogonal Frequency Division Multiplexing (OFDM) technology to eliminate the interference between data sub-carriers, the implement of Error Correction Code (ECC) is necessary. Among many ECCs such as Binary Convolution Code (BCC), Turbo Code (TC), Low Density Parity Check (LDPC) Code etc., the BCC has been selected as a mandatory part in Wi-Fi systems such as 802.11a/n/ac/ah.

To recover the transmitted data encoded by BCC, Viterbi decoder (VD) is required at the receiver side. Viterbi Decoding algorithm was developed by Andrew J. Viterbi in 1967 [4]. It has been widely used in many systems such as W-CDMA, Wi-Fi 802.11a/n/ac/ah, satellite communications, digital television, etc. in recent years. The low complex VD for decoding BCC with constraint length k=3 is commonly used for education purpose on high speed applications [5]. However, in real applications the complex VD for decoding BCC with long constraint length is utilized to improve the decoding performance such as PER. The modern Wi-Fi systems IEEE 802.11a/n/ac/ah employ BCC with constraint length $k = 7$. The equivalent VD has $2^{k-1} = 64$ status nodes per layer. It is thus called as 64-state VD. Basically, the complexity of Viterbi decoder is (2^{k-1}). 64-state VD is one of the most complex block [6] at the receiver side of Physical layer of Wi-Fi system.

In this paper, a low-complexity K_{min} Viterbi decoding (KVD) architecture is proposed. The orthodox and the proposed VDs are implemented into IEEE 802.11ah simulator and the PER performance of system is simulated when using the two decoders. After thorough result and mathematical analysis, it is concluded that the complexity of the proposed KVD is smaller than that of the orthodox VD by about 12.80 times to 21.33 times while KVD achieves the same PER performance as an orthodox VD does. Although this research aims to support IoT sensors based on 802.11ah standard, the research results are also expected to be applicable in other Wi-Fi systems such as 802.11a/n/ac, or W-CDMA, satellite communications, etc., that use VD for decoding BCCs with high constraint length. The rest of this paper is organized as follows: Sect. 2 briefly describes binary convolution coding (BCC) and orthodox Viterbi decoder. Sect. 3 explains proposed KVD decoding architecture. Sect. 4 evaluates the complexity of KVD decoder in terms of number of mathematical operations and computational time. Sect. 5 describes 802.11ah simulator and exposes PER/BER simulation results. Conclusion is discussed in the final section 6. In future, the work can be extended for speed optimization of modified VD to be dumped in FPGA hardware [7].

II. BCC CODED ORTHODOX VITERBI DECODING

A. Introduction

Fig. 1 illustrates the operation of BCC encoder and Viterbi decoder in Wi-Fi 802.11a/n/ac/ah systems. At the transmitter side BCC encoder encodes the information bit-stream I into two encoded bit-streams *A* and *B.* At the receiver side Viterbi decoder generates the bit-stream I' from A' and B', which are the estimated values of the encoded bit-streams *A* and *B*, respectively [8]. Because of interference, noise and fading effect of wireless environment, the estimated bit-streams A' and B' are hardly to be identical to *A* and *B*. The purpose of Viterbi decoder is to recover the transmitted bit-stream I from the wrongly-estimated A' and B'.

It means that I' should be the same as *I*. In case I' does not identical to *I*, the VD is considered to decode the data unsuccessfully. The transfer packet of data is received wrongly and the transmitter is requested to send the data again.

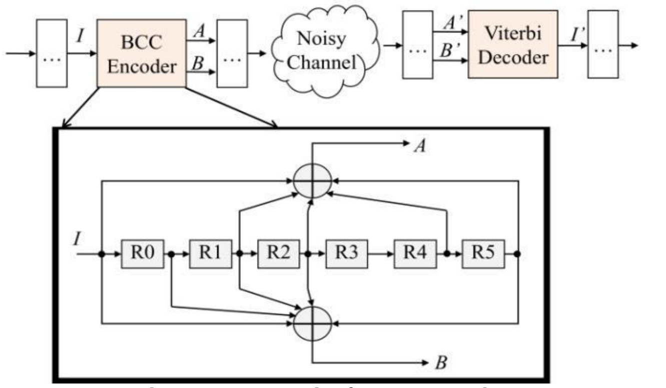


Fig.1 BCC encoder at transmitter side of 802.11a/n/ac/ah systems. constraint length $k = 7$, number of internal registers $m = k - 1 = 6$

B. Binary Convolution Coding (BCC)

The 802.11a/n/ac/ah systems [9] use BCC encoder with constraint length $k = 7$. The inside of BCC encoder is shown in Fig. 1. The encoder uses $m = k - 1 = 6$ registers, i.e., R_0 to R_5, and two XOR gates to generate two output bits *A* and *B* from every input bit I. The code rate is thus r = 1/2. To increase the transfer rate, the higher code rates such as 2/3, 3/4, 5/6, etc., may be selected by puncturing the encoded bit-streams *A* and *B* with an appropriate ratio. To implement XOR gates that generate *A* and *B*, the industry standard generator polynomials $g_0 = 133_8$ and $g_1 = 171_8$ are used, respectively. The hardware design of BCC encoder is simple and does not need much resource [10].

C. Operation of orthodox Viterbi Decoder (VD)

• Concept of Trellis Diagram

The trellis diagram of VD in 802.11a/n/ac/ah is shown in Fig. 2. It mimics the operation of BCC encoder. In which each layer of the trellis has 64 nodes showing $2^6 = 64$ status values, i.e., from 0 to 63, of $m = 6$ registers of BCC encoder. Because six registers of BCC encoder are initialized by zero value, the node with status value 0 in initial layer of VD's trellis, i.e., denoted as $node^{(0)} = 0$, is selected to start the decoding process. If the first input bit of BCC encoder $I^{(1)}$ equals to zero, or one, the next value of six registers $R_5R_4R_3R_2R_1R_0$ of BCC encoder will be 6′ b000000 = 0 or 6′ b000001 = 1 respectively. Therefore, there are two paths connecting $node^{(0)} = 0$ to $node^{(1)} = 0$ and $node^{(0)} = 0$ to $node^{(1)} = 1$. Note that $node^{(1)} = x$ refers to status node *x* in layer *l*. In layer 2, in case the previous status node $node^{(1)} = 0$, the current status nodes should be $node^{(2)} = 0$ and $node^{(2)} = 1$ if the second input-bit of BCC encoder $I^{(2)} = 0$ and $I^{(2)} = 1$, respectively. Therefore, there are two paths connecting $node^{(1)} = 0$ to $node^{(2)} = 0$ and $node^{(1)} = 0$ to $node^{(2)} = 1$. In case $node^{(2)} = 1$, the current status nodes should be $node^{(2)} = 2$ and $node^{(2)} = 3$ if $I^{(2)} = 0$ and $I^{(2)} = 1$, respectively. Thus, there are two paths connecting $node^{(1)} = 1$ to $node^{(2)} = 2$ and $node^{(1)} = 1$ to $node^{(2)} = 3$. In total, there are

4 connection paths in layer 2. Similarly, there are 8, 16, 32, and 64 paths in layers 3, 4, 5, and 6. For layer l ($l \geq 7$), there are $64 \times 2 = 128$ paths connecting 64 nodes in layer l - 1 to 64 nodes in layer l, refer to Fig. 2.

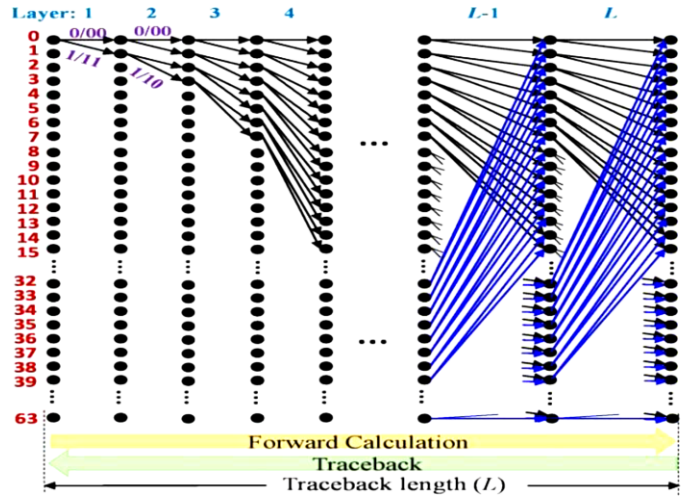


Fig.2 Trellis Diagram of orthodox VD in 802.11a/n/ac/ah systems with each layer $2^m = 64$ status nodes

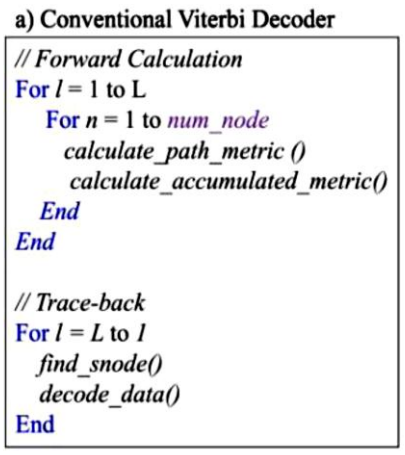
In addition, a path that connects node$^{(l-1)}$ to node$^{(l)}$ is accompanied with a pair of value $I^{(l)}/A^{(l)}B^{(l)}$, in which $I^{(l)}$, $A^{(l)}$ and $B^{(l)}$ are respectively the lth input-bit I, lth encoded-bit A and B of BCC encoder. While $I^{(l)}$ has two values zero and one, $A^{(l)}$ and $B^{(l)}$ can be calculated by eq. (1) and (2), respectively. For examples, two paths that connect node$^{(0)}$ = 0 to node$^{(1)}$ = 0 and node$^{(0)}$ = 0 to node$^{(1)}$ = 1 are respectively accompanied with values 0/00 and 1/11, see Fig. 2.

$A^{(l)} = \text{modulo}(I^{(l)} + R_1 + R_2 + R_4 + R_5, 2)$ …………….(1)

$B^{(l)} = \text{modulo}(I^{(l)} + R_0 + R_1 + R_2 + R_5, 2)$ ……………..(2)

• Decoding Process

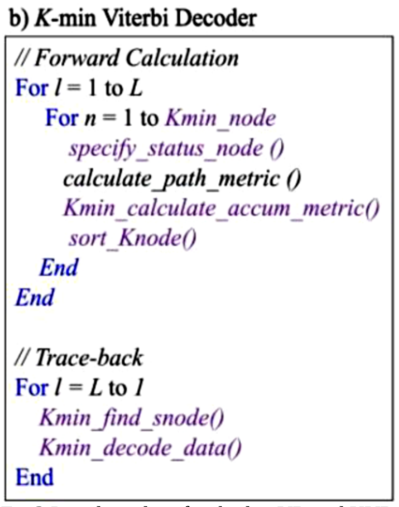
The input data of Viterbi decoder are the estimated bit-streams A' and B' of A and B. The decoding process follows two stages, i.e., forward calculation and trace-back, one after the other. The pseudo code of the decoding process is shown in Fig. 3a.



a) Conventional Viterbi Decoder

```
// Forward Calculation
For l = 1 to L
    For n = 1 to num_node
        calculate_path_metric ()
        calculate_accumulated_metric()
    End
End

// Trace-back
For l = L to 1
    find_snode()
    decode_data()
End
```



b) K-min Viterbi Decoder

```
// Forward Calculation
For l = 1 to L
    For n = 1 to Kmin_node
        specify_status_node ()
        calculate_path_metric ()
        Kmin_calculate_accum_metric()
        sort_Knode()
    End
End

// Trace-back
For l = L to 1
    Kmin_find_snode()
    Kmin_decode_data()
End
```

Fig.3 Pseudo codes of orthodox VD and KVD

Stage 1: Forward Calculation

This stage processes through a predefined L layers of the trellis. In which L is an important parameter of VD called as trace-back length. In each layer, a number of *num_node* loops will be repeated. Each loop performs two tasks such as *calculate_path_metric( )* & *calculate_accumulated_metric( ).* Note that num node presents the number of nodes should be calculated in each layer. For layer 1, 2, 3, 4, 5 and 6 values of *num_node* will be 1, 2, 4, 8, 16 and 32 respectively. For other layers, we have *num_node* = 64.

Task *calculate_ path_ metric( )* in layer l ($l \geq 1$) calculates path metric of all the paths connecting nodes in layer $l - 1$ to nodes in layer l. In case of hard-decision VD, path metric is measured as Hamming distance between the expected value $A^{(l)}$, $B^{(l)}$ and the estimated values $A'^{(l)}$, $B'^{(l)}$. The calculation is shown in eq. (3). In which $A^{(l)}$, $B^{(l)}$, $A'^{(l)}$ and $B'^{(l)}$ are all binary data. In case of soft-decision VD, path metric is measured as Euclidean distance between $A'^{(l)}$, $B'^{(l)}$ and $A'^{(l)}$, $B'^{(l)}$. The calculation is shown in eq. (4). In this case, $A^{(l)}$, $B^{(l)}$, $A'^{(l)}$ and $B'^{(l)}$ are the log likelihood ratio (LLR) values and are represented by a number of binary bits, i.e., denoted as D bits.

$p(\text{hard}) = | A'^{(l)} - A^{(l)} | + | B'^{(l)} - B^{(l)} |$ ………………(3)

$p(\text{sof t}) = (A'^{(l)} - A^{(l)})2 + (B'^{(l)} - B^{(l)})^2$ ….…………..(4)

Task *calculate_ path_ metric( )* in layer l ($l \geq 1$) calculates accumulated metric of all nodes in layer l. For layer l with $l < 7$, node j in layer l is connected with only one node, e.g., node i, in layer l - 1. The calculation follows eq. (5). For layer l with $l \geq 7$, node j in layer l is connected with two nodes, e.g., nodes i^1 and i^2, in layer l - 1. The calculation follows eq. (6) - (8). In which $m_i(l-1)$ and $m_j(l)$ denote accumulated metric of node i in layer l - 1 and of node j in layer (l) ($0 \leq i, j \leq 63$) respectively; $p_{i,j}$ denotes path metric of the path that connects node i and node j and min(a, b) function returns to the smaller value of a and b.

$m_j^{(l)} = m_i^{(l-1)} + p_{i,j}$ …………………………………(5)
$m1_j^{(l)} = m_{i1}^{(l-1)} + p_{i1,j}$ ……………………………… (6)
$m2_j^{(l)} = m_{i2}^{(l-1)} + p_{i2,j}$ ………………………………(7)
$m_j^{(l)} = \min(m1_j^{(l)}, m2_j^{(l)})$ ……………...(8)

It means that accumulated metric of a node in one layer is defined as the sum of accumulated metric of its connected node in previous layer with its path metric. For layer l with $l \geq 7$, each status node has two connected paths with previous layer's status nodes. The path that results to smaller accumulated metric is considered to be survival path. And that smaller accumulated metric will be selected.

Fig. 4 illustrates the calculation of accumulated metric for $node_{(l)} = 0$ and $node_{(l)} = 1$, with $l \geq 7$. The forward calculation is repeated from layer to layer until layer $l = L$ is completed. At layer L, node that has the smallest accumulated metric will be selected as a start point for the next stage, i.e., trace-back. Remember that L is known as trace-back length.

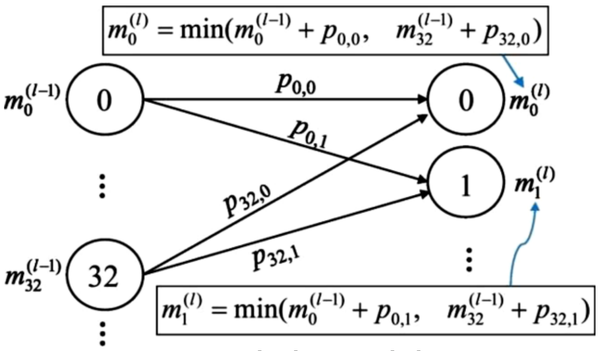

$m_0^{(l)} = \min(m_0^{(l-1)} + p_{0,0}, \quad m_{32}^{(l-1)} + p_{32,0})$
$m_0^{(l-1)}$
0
$p_{0,0}$
0
$m_0^{(l)}$
$p_{0,1}$
1
$m_1^{(l)}$
$p_{32,0}$
$m_{32}^{(l-1)}$
32
$p_{32,1}$
$m_1^{(l)} = \min(m_0^{(l-1)} + p_{0,1}, \quad m_{32}^{(l-1)} + p_{32,1})$

Fig.4 Accumulated Metric Calculation Process

Stage 2: Trace-back

Trace-back is a process in which the decoded data I' is computed bit-by-bit in the order I'^{L} , . . . , $I'^{(2)}$, $I'^{(1)}$. In each layer, there are two tasks such as *find_snode( )* and *decode_data( )* are performed, refer to Fig. 3a.

Task *find_snode( )* in layer l ($l \geq 1$) finds the most suitable node $snode^{(l)}$ among 64 nodes in that layer. For layer l = L, $snode^{(l)}$ is obtained by comparing accumulated metric of all 64 nodes in layer L. It should be the node that has the smallest accumulated metric. For the other layers, i.e., layer l with $l < L$, the most suitable node $snode^{(l)}$ is the one that has smaller accumulated metric between the two nodes having connection with $snode^{(l+1)}$ To know which nodes having connection with $snode^{(l+1)}$, the VD needs to memorize the survival paths of all 64 nodes in all L layers of the trellis. Task *decode_data( )* in layer l ($l \geq 1$) estimates $I'^{(l)}$ of the transmitted data $I^{(l)}$ from $snode^{(l)}$. To do that the VD needs to memorize the trellis architecture. It means that the accompanied data $I^{(l)}$ of all survival paths of the trellis must be stored in memory. The decoded data $I'^{(l)}$ will be the accompanied data $I^{(l)}$ of the path that connects $snode^{(l-1)}$to $snode^{(l)}$.

III. PROPOSED K_{MIN} VITERBI DECODING (KVD)

Similar to the orthodox VD, the proposed KVD also processes through L layers in which L is known as trace-back length. The process includes two stages such as forward calculation and trace-back. Unlike the orthodox VD which relies on architecture of trellis diagram, processing of KVD follows mathematical equations. Therefore, the KVD does not need to memorize trellis diagram. In addition, the KVD calculates path metrics and accumulated metrics of only K nodes (not all 64 nodes) per layer. In which K is a parameter of KVD ($K \leq 64$). The pseudo code of KVD is shown in Fig. 3b.

Stage 1: Forward Calculation

This stage processes through L layers. In each layer, a number of K_{min}*_node* loops will be repeated. Each loop performs four tasks such as *specify_status_node( )*, *calculate_path_metric( )*, *calculate_accumulated_metric( )* and *sort_Knode( )*. Let define K as a known parameter of KVD, K_{min}*_node* takes the minimum value of *num_node* and K. Remember that, for layer 1, 2, 3, 4, 5 and 6 values of *num_node* will be 1, 2, 4, 8, 16 and 32 respectively. For other layers, we have *num_node* = 64. For example, if K = 3 then K_{min}*_node* = 1 and K_{min}*_node* = 2 in layer 1 and 2 respectively and K_{min}*_node* = 3 for layer l ($l > 2$). Task *specify_status_node( )* in layer l ($l \geq 1$) specifies child nodes $cnode^{(l)}$ of layer l that have connection with each of the K_{min} *_node* parent nodes $pnode^{(l-1)}$ in layer l - 1.

In this paper, we consider $pnode^{(l-1)}$and $cnode^{(l)}$ as parent node and child node in layer l respectively. Note that $pnode^{(l-1)}$ and $cnode^{(l)}$ respectively present status values of six registers of BCC encoder in the (l - 1)th and the lth cycles. Because after each cycle value of six registers is shifted to the most significant bit (MSB) by one bit, the value should be double then be modulo by $2^6 = 64$. In addition, the least significant bit R_0 of the six register is assigned to the lth input bit $I^{(l)}$, the new value of six registers should be added to $I^{(l)}$ after being doubled. Therefore, we can specify $cnode^{(l)}$ from $pnode^{(l-1)}$as eq. (9) in which $I^{(l)}$ has two values zero and one.

$cnode^{(l)} = \text{modulo}(2 \times pnode^{(l-1)}, 64) + I^{(l)}$ ……………… (9)

For example, from parent node $pnode^{(l-1)} = 5$ there are two child nodes $cnode^{(l)} = 10$ and $cnode^{(l)} = 11$; and from $pnode^{(l-1)} = 33$ there are two child nodes $cnode^{(l)} = 2$ and $cnode^{(l)} = 3$ refer to Fig. 5a. If layer l - 1 has K parent nodes $pnode^{(l-1)}$ a maximum of 2K child nodes $cnode^{(l)}$ will be generated. In case any two parent nodes have the same connection with any two child nodes, the total number of child nodes will be smaller than $2K$. Because of the modulo function in eq. (9) we know that two parent nodes $pnode^{(l-1)} = x$ (x = 0, 1, . . . , 31) and $pnode^{(l-1)} = 32 + x$ will have the same child nodes. An example in Fig. 5a shows that there are 6 but not 8 child nodes are generated from $K = 4$ parent nodes. That is because two parent nodes $pnode^{(l-1)} = 1$ and $pnode^{(l-1)} = 31$ have the same child nodes $cnode^{(l)} = 2$ and $cnode^{(l)} = 3$. Task *calculate_path_metric( )* calculate the path metrics. From K parent nodes there are $2K$ path metrics will be calculated. The calculation follows eq. (3) or (4) as the orthodox VD does.

Task K_{min}*_calculate_accum_metric( )* in layer l ($l \geq 1$) calculates accumulated metrics of the child nodes specified in *specify_status_ node( )* task. From Fig. 5a we see that a child node $cnode^{(l)}$ may have one or two connection paths. The path may come from $pnode^{(l-1)} = x$ (x = 0, 1, . . . , 31), i.e., known as $s^{(l)} = 1$ or come from $pnode^{(l-1)} = 32 + x$, i.e., known as $s^{(l)} = 2$. For simplicity we assume that there are $2K$ different child nodes generated from K parent nodes and each child node have only one connection path. The path may be $s^{(l)} = 1$ or $s^{(l)} = 2$.

For example, Fig. 5b shows that child node $cnode^{(l)} = 2$ generated from $pnode^{(l-1)} = 1$ and from $pnode^{(l-1)} = 31$ is considered to be two different child nodes with the same status value $cnode^{(l)} = 2$ but different paths, i.e., $s^{(l)} = 1$ and $s^{(l)} = 2$ respectively.

For child node that has $s^{(l)} = 1$, the accumulated metric $m1^{(l)}$ is calculated as eq. (6) while, $m2^{(l)}$ is assigned to a very large constant *MAX* that is expected to be always larger than $m1^{(l)}$. Similarly, for child node that has $s^{(l)} = 2$, $m2^{(l)}$ is calculated as eq. (7) while $m1^{(l)}$ is assigned to *MAX*. The final accumulated metric of child node $cnode^{(l)}$ is the minimum value of $m1^{(l)}$ and $m2^{(l)}$ as eq. (8), refer to Fig. 5b.

Task *sort_Knode( )* in layer l ($l \geq 1$) will find the top child nodes that have smallest accumulated metrics. It then finds whether there are any two child nodes having the same status value $cnode^{(l)}$ but different accumulated metric $m^{(l)}$ or not. If there are, the one with larger accumulated metric will be removed from the list. Finally, only K nodes $pnode^{(l)}$ will be selected to become the parent nodes for next layer, refer to Fig. 5c and 5d.

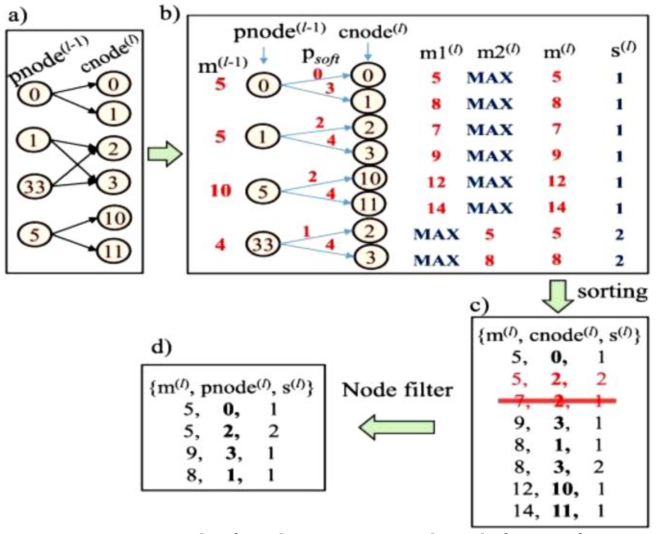


Figure 5 An example of one loop operation in layer l of KVD of $K = 4$

Stages 2: Trace-back

Similar to the orthodox VD, trace-back stage of KVD computes the decoded data I' bit-by-bit in the order $I'^{(L)}, \ldots, I'^{(2)}, I'^{(1)}$. Each layer performs two tasks such as K_{min}*_find_snode( )* and K_{min}*_decode_data( ),* refer to Fig. 3b. Task K_{min}*_find_snode( )* in layer l ($l \geq 1$) finds the most suitable node $snode^{(l)}$ among K parent nodes in that layer. For layer $l = L$, $snode^{(l)}$ is obtained by comparing accumulated metric of $2K$ child nodes $cnode^{(l)}$. It should be the node that has the smallest accumulated metric.

In addition, from eq. (9) we deduce to eq. (10) and (11) to specify two nodes $snode_1^{(l-1)}$ and $snode_2^{(l-1)}$ in layer l -1 that have connection with $snode^{(l)}$ in layer l. In which *floor*(x) function returns to the largest integer value that is smaller than or equal to x. If we call $s^{(l)}$ as the survival path of $snode^{(l)}$then $s^{(l)}$ is equal to either 1 or 2. If $s^{(l)} = 1$ the connection between $snode^{(l)}$ and $snode_1^{(l-1)}$ results to smaller accumulated metric. Otherwise, the connection between $snode^{(l)}$ and $snode_2^{(l-1)}$ results to smaller accumulated metric. Therefore, we propose eq. (12) to trace back the selected node $snode^{(l-1)}$ from $snode^{(l)}$. By using eq. (12) KVD needs to memorize only K status nodes $pnode^{(l)}$ and their survivor paths $s^{(l)}$ per layer.

$$snode_1^{(l-1)} = floor(snode^{(l)}/2) \quad \ldots\ldots\ldots\ldots (10)$$

$$snode_2^{(l-1)} = floor(snode^{(l)}/2) + 32 \quad \ldots\ldots\ldots\ldots (11)$$

$$snode^{(l-1)} = floor(snode^{(l)}/2) + 32 \times (s^{(l)} - 1) \quad \ldots\ldots\ldots (12)$$

For example, if $snode^{(l)} = 1$ and $s^{(l)} = 2$, then we know that $snode^{(l-1)} = 32$.

Task *decode_data( )* in layer l ($l \geq 1$) estimates $I^{(l)}$ of the transmitted data $I^{(l)}$. Instead of memorizing the accompanied data $I^{(l)}$ of all paths as the orthodox VD does, we propose a mathematics equation to derive $I'^{(l)}$ from $snode^{(l)}$. From eq. (9) we see that if $I^{(l)} = 0$, value of $node^{(l)}$ will be even. Otherwise, value of $node^{(l)}$ will be odd. Based on the even/odd property of $snode^{(l)}$, we can know whether $I'^{(l)} = 0$ or $I'^{(l)} = 1$. In short, we propose eq. (13) to decode $I'^{(l)}$ from $snode^{(l)}$.

$I'^{(l)}$ = modulo($snode^{(l)}$, 2) ……………………………(13)

Fig. 6 shows a simple example of KVD with $K = 3$ and trace-back length $L = 8$. In most of layers $2K = 6$ child nodes are generated and only $K = 3$ nodes will be selected to become parent nodes of next layer. In layer 7, only 4 child nodes are generated because the parent nodes 8 and 40 have the same child nodes 16 and 17. In the trace-back stage, we assume that path from node 40 to 16 is the survival path. Therefore, node 40 (but not node 8) will be selected.

Finally, the trace-back line goes through nodes 33, 16, 40, 20, 10, 5, 2, 1 and 0 see Fig. 6. Therefore, the decoded data will be $I^{(8)} \rightarrow I^{(1)}$ = 1, 0, 0, 0, 0, 1, 0, 1.

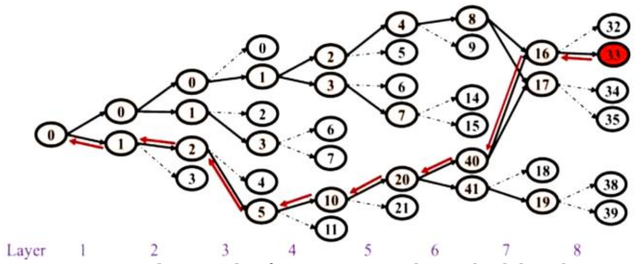


Fig.6 A simple example of KVD. $K = 3$ and trace-back length $L = 8$

IV. Complexity Analysis of Both Decoding Architecture

In this section the complexity of orthodox VD and KVD in terms of number of mathematical operations and processing time is thoroughly analyzed.

4.1 Mathematical operations in VD and KVD

We denote $N_{sub/add}$, N_{abs} and N_{sqr} respectively as the total number of subtract/adder, absolute and square operations will be used by Viterbi decoder. To calculate a path metric, eq.(3) and (4) show that a hard decision VD requires 3 subtract/adder and 2 absolute operations while a soft decision VD requires 3 subtract/adder and 2 square operations. To calculate accumulated metric for a node, eq. (6), (7) and (8) show that 2 subtract/adder operations are needed. Because the trace-back length L of VD is commonly much larger than 6, the following conclusions are approximately true.

• For the orthodox VD, there are $64 \times 2 = 128$ path metrics and $2^{(k-1)} = 64$ ($k = 7$) accumulated metrics will be calculated per layer. Therefore, we have $N_{sub/add} = (3 \times 128 + 2 \times 64) \times L = 512 \times L = 8 \times 2^{(k-1)} \times L$. In addition, there are $N_{abs} = 2 \times 128 \times L = 256 \times L = 4 \times 2^{(k-1)} \times L$ and $N_{sqr} = 4 \times 2^{(k-1)} \times L$ operations are respectively required in case of hard decision and soft decision.

• For the KVD, we have $N_{sub/add} = (6K+2K) \times L = 8 \times K \times L$ and $N_{abs} = 4 \times K \times L$ and $N_{sqr} = 4 \times K \times L$ are required in case of hard decision and soft decision respectively. We see that the numbers of subtract/adder, absolute and square operations of KVD does not affected by BCC's constraint length k or the number of status nodes of decoder, while the orthodox VD does. Table 1 shows the number of arithmetic operations of orthodox VD and KVD in case trace-back length $L = 60$. At the outset, it can surely be concluded that the number of arithmetic operations is lower by $64/K$ times if using KVD instead of orthodox VD.

4.2 Total computational time analysis report

The proposed system that includes BCC encoder ($k = 7$) and Viterbi decoding is done in MATLAB. The VD is whether orthodox VD or KVD with several values of K such as K = 1, 3, 5, 10 and 32. The number of transfer bits per packet is 4000 and the number of packets is 5000, the trace-back length $L = 60$. Each packet is divided into (4000/60) = 67 blocks. The processing time of system in case of using orthodox VD and KVD with K = 1, 3, 5, 10 and 32 is shown in Table 2. In case of soft-decision, $D = 3$ bits is selected.

Table 1 Number of Arithmetic Operations with $L = 60$

VD Type	$N_{sub/add}$	N_{abs} (Hard)	N_{sqr} (Soft)
Orthodox VD	30720	15369	15369
KVD, K= 32	15360	7680	7680
KVD, K= 10	4800	2400	2400
KVD, K= 5	2400	1200	1200
KVD, K= 3	1440	720	720
KVD, K= 1	480	240	240

Table 2 Processing time of system when using orthodox VD and KVD with $L = 60$ and $D = 3$

VD Type	Hard Decision		Soft Decision	
	Time	Unit	Time	Unit
Orthodox VD	1 Hr. 9m 45s	58.1	1 Hr. 15m 30s	58.5
KVD, K= 32	29m 45s	24.8	32m 53s	25.6
KVD, K= 10	8m 41s	7.2	9m 11s	7.2
KVD, K= 5	4m 26s	3.7	4m 58s	3.9

KVD, K= 3	2m 50s	2.4	3m 12s	2.5
KVD, K= 1	1m 12s	1	1m 17s	1

In table 2, the Time column shows the processing time in term of hour(h)/minute(m)/second(s). The time h/m/s is then converted into the number of seconds and divides to the processing time of KVD with K = 1. The result is shown in Unit column. From Table 2 we see that the processing time in both cases, i.e., hard-decision and soft-decision, is almost linear proportional to K value. It means that we can reduce the computational resource by approximately 64/K times if using KVD instead of orthodox VD. The processing of soft-decision VD is a little bit longer than that of hard-decision VD. That is because the calculation of square operation takes more clock cycles than the calculation of absolute operation.

V. PER/BER Simulation Results

To simulate performance of the orthodox VD and the proposed KVD, a draft version of 802.11ah simulator is used. Block diagram of the simulator is shown in Fig. 7. At the transmitter side, 'PSDU GEN' block generates a stream of random data for transmitting. 'SIG GEN' block generates the data for signal field of the transfer packet. 'Scrambler' block performs the scrambling to avoid sequence of zeros or ones. 'BCC Encoder' block encodes information bits. 'OFDM SYM SPLIT' block splits the stream of data into a number of orthogonal frequency division multiplexing (OFDM) symbols. 'Interleave' block changes the bit-order in each OFDM symbols. Its purpose is to deal with burst error that may appear during the transmission. 'Mapper' block maps the data into constellation. The 802.11a/n/ac/ah support several types of modulation such as binary phase shift keying (BPSK), quadrature phase shift keying (QPSK), 16 quadrature amplitude modulation (16-QAM), 64-QAM and 256-QAM. 'Pilot' block generates the pilot values. 'Preamble Memory' stores preamble data of the transfer packet. This data is used by receiver to estimate channel condition. 'Subcarrier Allocate' block allocates output of 'Mapper' to data subcarriers, and output of 'Pilot' to pilot subcarriers of OFDM symbols. Each OFDM symbol has 64 subcarriers which include 48 data subcarriers, 4 pilot subcarriers and 8 null subcarriers. 'IFFT' blocks perform the invert fast Fourier transforms so that subcarriers within symbol become orthogonal. It also converts data from frequency-domain to time-domain. 'GII' block inserts guard interval (GI) to protect data of a symbol from interference with data of adjacent symbols. In this simulator we apply normal guard interval. 'OFDM SYM CONCAT' block concatenates the OFDM symbols into a continuous stream. This stream of data is sent to the receiver via a wireless channel which can be additive white Gaussian noise (AWGN) channel or Rayleigh fading channel.

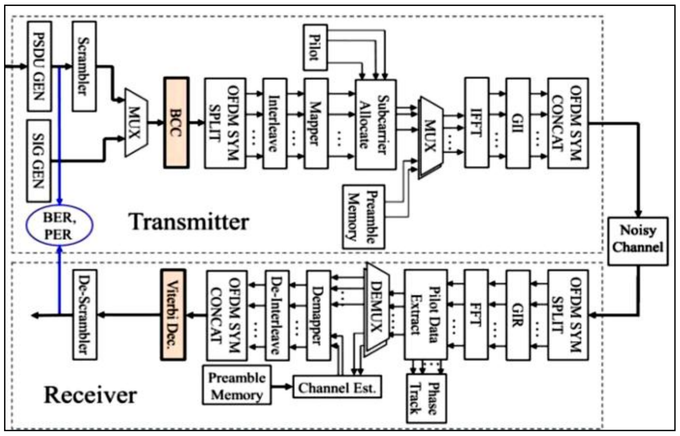


Fig.7 Block Diagram of 802.11ah PHY simulator

The blocks at receiver side do the opposite tasks as those of transmitter side do. In detail, 'OFDM SYM SPLIT' splits stream of data into several OFDM symbols. 'GIR' removes the GI subcarriers from OFDM symbols. 'FFT' performs fast Fourier transfer to convert data from time domain to frequency domain. 'Pilot Data Extract' extracts data and pilot subcarriers for further processing. The pilot subcarriers are used for phase tracking which is not implemented in this simulator. In the channel training phase, the data subcarriers are used by 'Channel Est.' block to estimate the channel condition. In the data phase, the data subcarriers are passed to 'Demapper'. In case of hard decision, 'Demapper' estimates the input values of 'Mapper'. In case of soft decision, 'Demapper' calculates the LLR values of input data of 'Mapper'. 'De-Interleave' does the opposite task of 'Interleave'. In which it returns the data subcarriers into the original position. 'OFDM SYM CONCAT' concatenates the data from OFDM symbols into one stream before giving to 'Viterbi Dec.' The 'Viterbi Dec.' decodes the received data to obtain the transmitted information. 'De-Scrambler' does the opposite task of 'Scrambler'. 'Scrambler' and 'Descrambler' are used to avoid long sequence of zeros or ones. Thus, they can partly solve the high peak to average power ratio (PAPR) problem of a communication system. The output of 'Descrambler' at receiver side is compared with the input of 'Scrambler' at transmitter side to check whether the receiver can recover the transmitted data correctly or not. In this work, the BER and PER performance of K_{min} Viterbi decoder is evaluated with several values of K such as 1, 3, 5, 10, 32 and 64. Note that, K = 64 of KVD have the same performance as orthodox Viterbi decoder. The simulation parameters are shown in Table 3.

Table 3 Simulation parameters

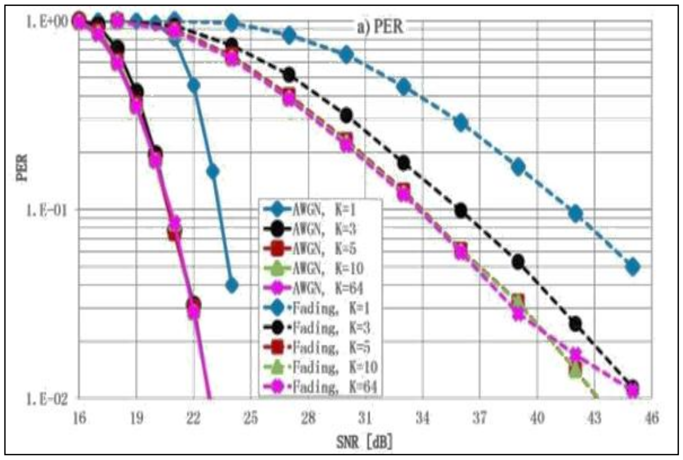
Parameters	Values
Simulator	IEEE 802.11ah draft version
Number of iterations	5000
Number of special streams in T_x x R_x	1 x 1

Channel type	AWGN, Rayleigh fading
Channel estimation	Ideal
Modulation types	BPSK, 16 – 256 QAM
Code rate	½, ¾
Transfer data type	Random
Viterbi Decoders data bit width	3-bit soft-decision

5.1 KVD versus Channel Type

In this subsection, performance of KVD in AWGN and Rayleigh fading channels is evaluated. In case of fading channel, the number of channel taps is set to 5. The PER performance is shown in Fig. 8a. In addition, the BER performance is also provided in Fig. 8b for reference. From Fig. 8a we can see that the PER performance of KVD in AWGN channel is better than that of in Fading channel. In case of AWGN channel, KVD with K = 3 achieves the same PER as the orthodox VD, i.e., KVD with K = 64, does. In case of fading channel, KVD with K = 5 achieves the same PER as the orthodox VD does. However, the PER performance of KVD with K = 1 is very worse in both channels. Amounts of PER performance degradation (K = 1) are 3dB and 8dB in cases of AWGN and Fading channels respectively. Therefore, the KVD with $3 \leq K \leq 5$ is recommended for real hardware implementation to reduce the complexity of decoder by approximately 64/5 = 12.80 times to 64/3 = 21.33 times.

For applications in which BER performance is more important than PER performance, the BER results in Fig. 8b are shown for reference. This figure shows that the BER performance of KVD is eventually degraded if K value reduces. However, the degradation of BER performance of KVD is less significant in case of AWGN channel. Because of the degradation of BER performance, the using of KVD is a trade-off between complexity and BER performance.



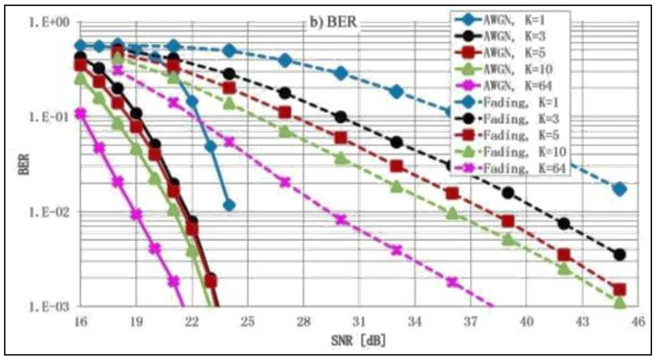
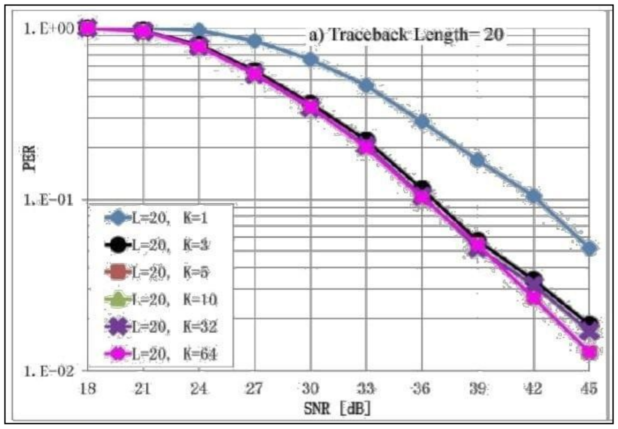


Fig. 8 PER & BER performance: KVD versus channel types. 64QAM, L = 60, PS = 100 bytes/packet.

5.2 KVD versus trace-back length L

In this subsection, performance of KVD in relation with several values of decoder's trace-back length L is evaluated. Basically, selecting L value is a trade-off between PER/BER performance and hardware cost. There is a fact that increasing the value of trace-back length L will result to better PER/BER performance but require more hardware cost and power consumption. Fig. 9 shows the PER performance of 802.11ah system when using KVD with several values of trace-back length L such as L = 20, L = 60, L = 100, and L = 800. These results prove for the fact that increasing L value will result to better PER performance. For example, to achieve the same PER performance (PER = 0.1), the orthodox VD (with K = 64) can reduce the signal-to-noise ratio (SNR) from SNR = 36 dB to SNR = 33.5 dB to SNR = 31.5 dB and to SNR = 27 dB by increasing the trace-back length from L = 20 to L = 60 to L = 100 and to L = 800 respectively. However, the Viterbi decoder with smaller value of L is preferred for applications which require low-cost and low-power such as IoT sensors Wireless LANs [11], [12]. Fig. 9 also shows that when using Viterbi decoder with smaller trace-back length L, the proposed KVD decoding one is more robust in terms of reducing complexity [13].



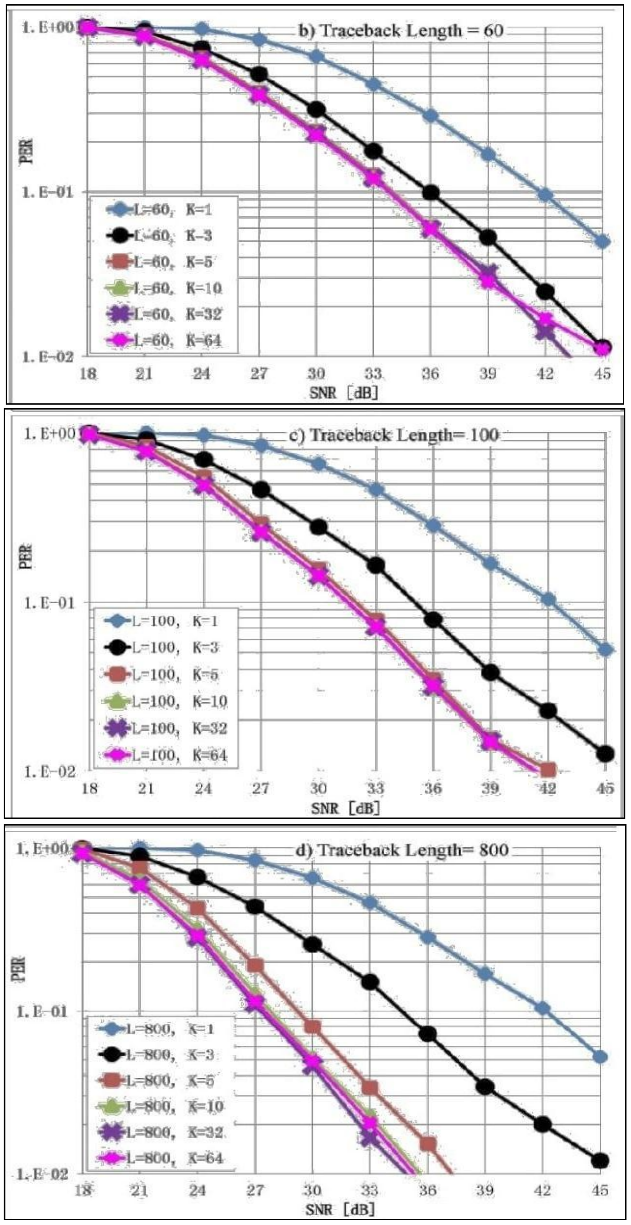


Fig. 9 PER performance: KVD versus decoder's trace-back length. 64QAM, *PS* = 100 bytes/packet.

For examples, in cases L = 20, L = 60, L = 100 and L = 800, the proposed KVD with K = 3, K = 5, K = 5 and K = 10 respectively achieve the same PER performance as the orthodox VD does. It means that the complexity can be reduced by 21.33, 12.80, 12.80 and 6.4 times respectively. For all the cases, PER performance of KVD with K = 1 is very worse. The PER performance degradation is above 5dB in all cases. Therefore the KVD with $20 \le L \le 60$ and $3 \le K \le 5$ is recommended for developing IoT sensors. In addition, within an amount of acceptable hardware cost, using the proposed KVD decoding one with larger trace-back length L will provide better PER performance as compared to the orthodox one with smaller L does.

For examples, Fig. 10 shows that PER performance of the KVD with K = 5, L = 800 is better than that of the orthodox one with K = 64, L = 100; and PER performance of the KVD with K = 5, L = 60 is better than that of the orthodox VD with K = 64, L = 20.

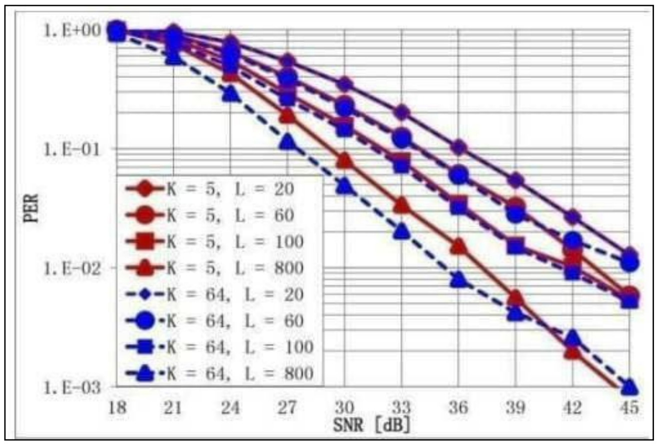
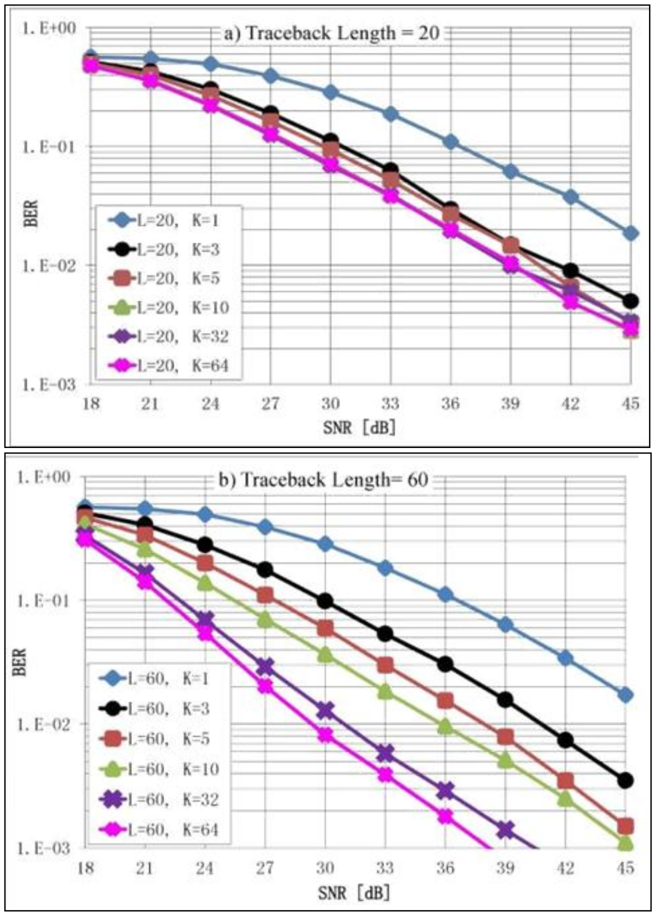
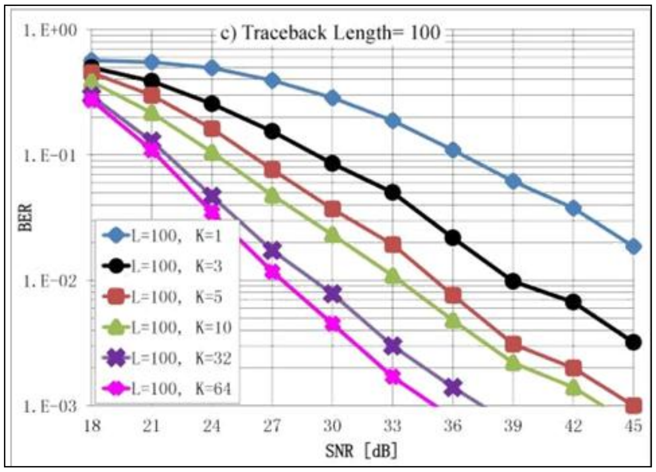


Fig. 10 PER performance: VD with K = 5 and K = 64 versus trace-back length L = 20, 60, 100, 800. 64-QAM, PS = 100 bytes/packet





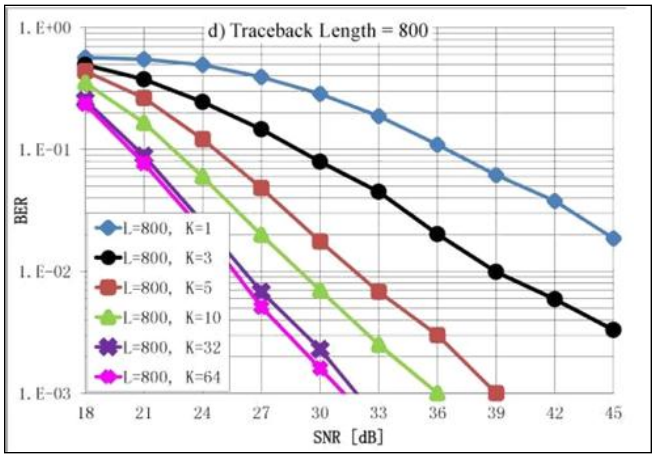
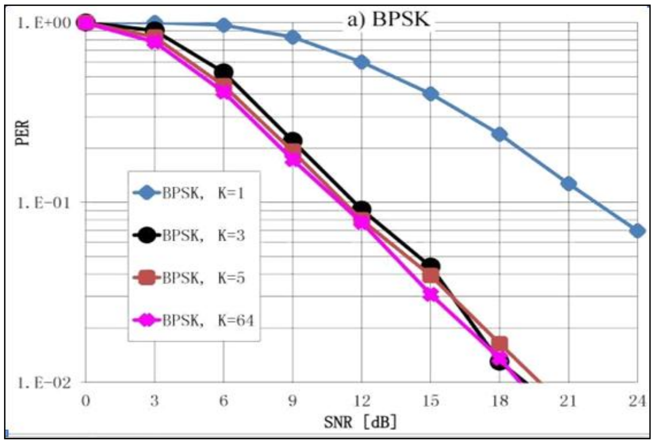


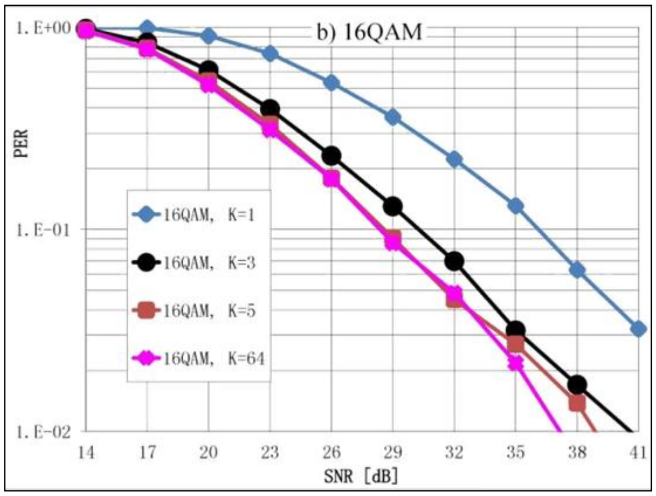
Fig. 11 BER performance: KVD versus decoder's trace-back length. 64QAM, PS = 100bytes/packet.

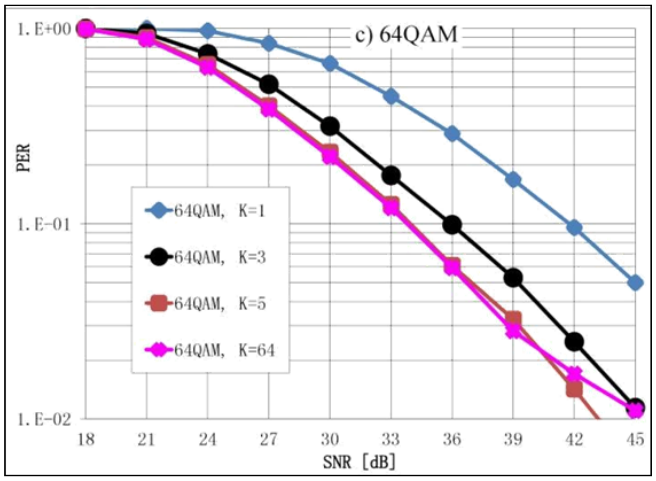
For reference purpose, the BER performance of KVD is shown in relation with L in Fig. 11. This figure shows that in case of small L, e.g., L = 20, the degradation of BER performance when K = 5 as compared to K = 64 is insignificant. However, for large L such as $L \geq 60$ the BER performance is eventually degraded and amount of degradation is significant. In these cases, the proposed KVD decoding one may be used in consideration the trade-off between BER performance and complexity.

5.3 KVD versus modulation type

In this subsection, we evaluate the PER performance of KVD in relation with several modulation types such as BPSK, 16QAM, 64QAM and 256QAM. Our simulation results are shown in Fig. 12. The PER performance of K_{min} one with K = 1 is very worse in all cases. As compared to case K = 64, the PER performance degradation is about 10dB, 8dB, 8dB and 7dB if modulation type is BPSK, 16QAM, 64QAM and 256QAM respectively (at PER = 0.1). If K = 3 the PER performance is significantly improved. As compared to case K = 64, the PER performance degradation is only about 0.2dB, 1dB, 1.8dB and 1.3dB in cases of BPSK, 16QAM, 64QAM and 256QAM respectively. In all modulation types, the K_{min} one with K = 5 achieves the same PER performance as the orthodox one (with K = 64) does. Selecting K value in the range $3 \leq K \leq 5$ is again recommended to reduce the complexity of decoder by 64/5 = 12.80 times to 64/3 = 21.33 times.







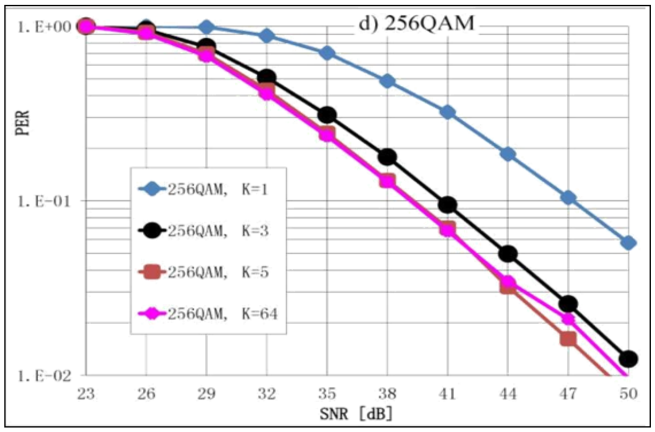
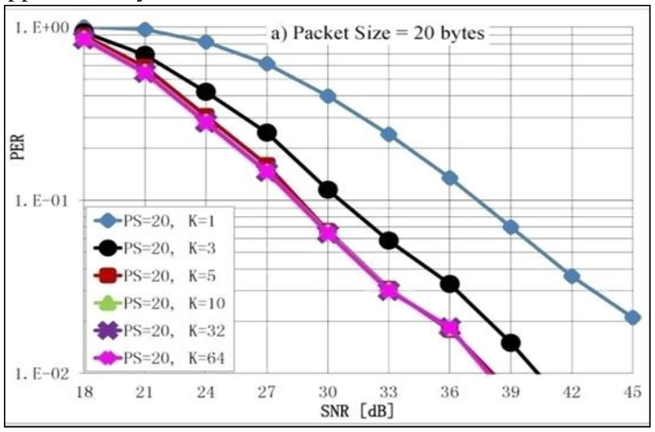
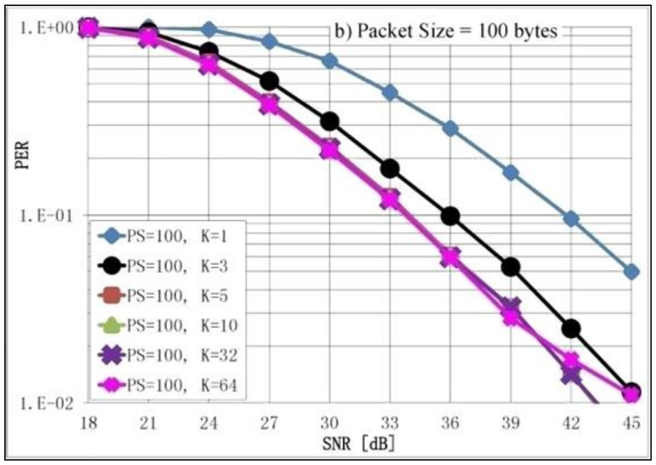


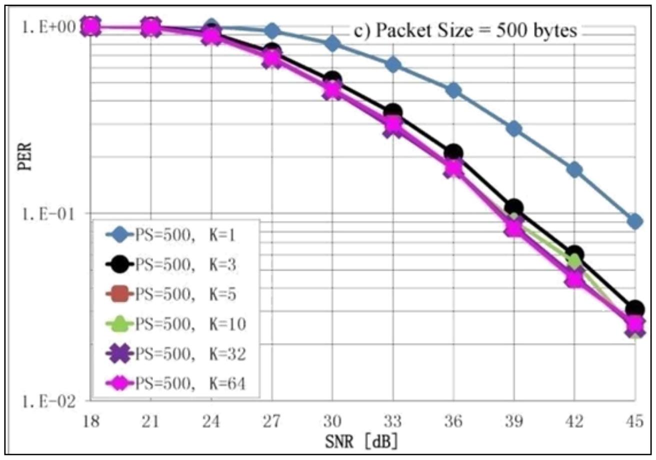
Fig. 12 PER performance: KVD versus modulation types. Fading channel with 5 taps, L = 60, PS = 100 bytes/packet.

5.4 KVD versus packet size (PS)

In this subsection the PER performance of KVD in relation with several packet sizes PS is evaluated such as PS = 20, PS = 100 and PS = 500 bytes/packet. The simulation results are shown in Fig. 13. From the results in Fig.13a, b and c, we can see that PER performance of KVD with K = 1 is very worse in all cases. The PER degradation is above 5dB (at PER = 0.1). When K = 3 the PER performance is much more improved. As compared to the orthodox decoder with K = 64, the PER degradation is about 1.7dB, 1.5dB and 0.3dB (at PER = 0.1) in cases PS = 20, PS = 100 and PS = 500 bytes/packet, respectively. In all PS cases, the KVD with K = 5 achieves the same PER performance as the orthodox VD (K=64) does. In addition, Fig. 13d shows that the PER performance is degraded if packet size PS increases. This is a reasonable result because once the packet size increases, the probability that the entire received packet cannot successfully recovered from noise/interference will increase. The interesting result is that even the PER performance is degraded as PS increases, the KVD with K = 5 always achieves the same PER performance as the orthodox VD does. For IoT sensor, value of K in the range $3 \leq K \leq 5$ is recommended for reducing the decoder complexity by approximately 12.80 times to 21.33 times.







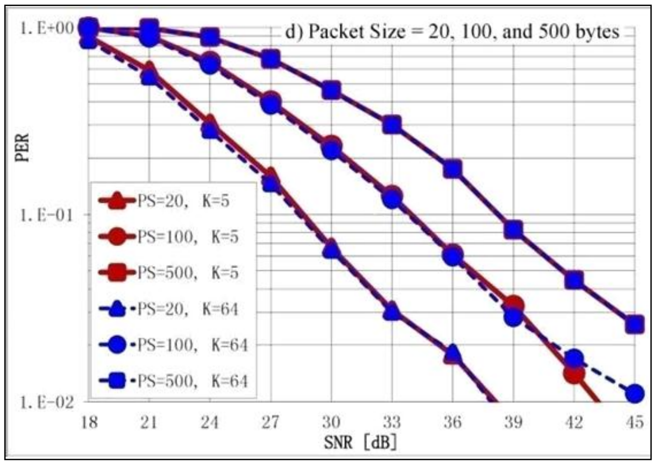


Fig. 13 PER performance: KVD versus packet size. 64QAM, Fading channel with 5 taps, L = 60

VI. CONCLUSION

In this paper, a less-complex K_{min} Viterbi decoder (KVD) for Wi-Fi 802.11a/n/ac/ah systems is proposed. Especially, the decoder aims to support the development of low-cost low-power 802.11ah IoT sensor. The complexity of KVD in terms of number of arithmetic operations and decoder processing time has been thoroughly evaluated. The evaluation results show that the complexity of KVD is reduced by 64/*K* times as compared to the orthodox VD. In the aspect of decoding performance, IEEE 802.11ah simulator simulated the PER and BER performance of 802.11ah system when using KVD with several *K* values such as 1, 3, 5, 10, 32 and 64. In which KVD with *K* = 64 is identical to orthodox VD. A lot of simulation conditions have been considered, for example, both AWGN and Rayleigh fading channel types, a wide range of decoder's trace-back length (*L* = 20, 60, 100, 800) most of modulation types (BPSK, 16-256QAM), and several packet sizes (*PS* = 20, 100, 500 bytes). The simulation results show that PER performance of KVD with *K* = 1 is too worse to be applicable. Its PER performance degradation from orthodox VD is above 5dB in most of the simulation conditions. However, by a little bit increasing *K* value to $3 \le K \le 5$ PER performance of KVD reaches to the same as that of the orthodox VD. Therefore it is recommended KVD with $3 \le K \le 5$ for real hardware implement of the decoder. By doing so it reduces the complexity of VD by approximately 64/5 = 12.80 times to 64/3 = 21.33 times while guaranteeing the same PER performance as the orthodox VD does.

At the outset, the proposed KVD is very suitable for low-cost low-power 802.11ah transceiver in IoT sensors. Designing hardware circuits of KVD with *K* = 3 and *K* = 5 for 802.11ah transceiver is the work can be done in future. The proposed decoding can also be useful for the high throughput Wi-Fi systems such as 802.11a/n/ac.

References

[1] Liu T.Y., Zhou G.: *"Key Technologies and Applications of Internet of Things"*. In the Proceedings of fifth Int. Conf. on Intelligent Computation Technology and Automation (ICICTA-2012) , Xian, pp.197–200, 2012.

[2] Girau R, Martis S, Atzori L, "*Lysis: a platform for IoT distributed applications over socially connected objects*", IEEE Internet of Things Journal (IoTJ), Vol.4(1), pp. 1-12, 2016.

[3] Tran T. H., Nagao Y., Kurosaki M., Sai, B., Ochi, H.: *"ASIC Implement of 600Mbps IEEE 802.11n 4x4 MIMO Wireless LAN System"*. In the Proceedings of 14th IEEE Int. Conf. on Advan. Commu. Tech. (ICACT-2012), Korea, pp. 360–363, 2012.

[4] Viterbi, A.: *"Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory"* Vol.1(1) pp. 260-269, 1967.

[5] Dutta R., Van Der Mast C., *"Virtex 4 FPGA Implementation of Viterbi Decoded 64-bit RISC for High Speed Application using Xilinx"*, International Journal of Computer Applications (IJCA), Vol.88(14), pp. 30-35, 2014.

[6] Maharatna K., Troya A., Krstic M., Grass E.: *"On the implementation of a low-power IEEE 802.11a compliant Viterbi decoder"*. In the Proceedings of 19th Int. Conference on VLSI Design (DOI:10.1109/VLSID). Bombay, pp.124, 2006.

[7] Chakraborty D, Raha P, Bhattacharya A, Dutta R; *"Speed Optimization of a FPGA based modified Viterbi Decoder"*, in the Proc. of IEEE Int. Conf. on Computer, Communication and Informatics (ICCCI-2013), Coimbatore, pp. 321-326; 04-06 January, 2013. ISBN: 978-1-4673-2907-1.

[8] Sandesh, Y.M., Kasetty, R.: "*Implementation of Convolution Encoder and Viterbi Decoder for Constraint Length 7 and Bit Rate ½"*, Int. Journal of Engineering Research and Applications Vol.3(4), pp. 42-46, 2013.

[9] IEEE 802.11 committee: Part 11: *"Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications"*. IEEE Std. 802.11a, Vol.38(6), 1999.

[10] Tran T.H., Nagao Y., Ochi H.: *"Algorithm and Hardware Design of A 2D Sorter-based K-best MIMO Decoder"*. EURASIP Journal on Wireless Communications and Networking DOI: 10.1186/1687-1499-2014-93, Vol.8(3), 2014.

[11] Dutta R, Pradhan PC, Sharma P, Guha S: "*VoIP Technology based modified CAC scheme for IEEE 802.11 Wireless LANs"*, In the Proceedings of Int. Conf. on Computation and Communication Advancement (IC3A) – 2013, West Bengal, pp. 174-180, 2013.

[12] Dutta R, Chaudhuri D, Mohanto B, Das M; "*A Proposed DLCCS Algorithm for High Speed Operation & Implementation on FPGA*", International Journal Nanotechnology & Applications (IJNA), Vol.8(1), pp. 01-12, 2014.

[13] Liu Y.S., Tsai Y.Y.: *"Minimum decoding trellis length and truncation depth of wrap-around Viterbi algorithm for TBCC in mobile WiMAX"*, EURASIP Journal on Wireless Communications and Networking, Vol. 44(8) DOI: 10.1186/1687-1499-2011-111, 2011.

Authors Profile

Ritam Dutta pursued his Bachelor of Technology from Sikkim Manipal Institute of Technology, Sikkim Manipal University, Sikkim, India in 2007 and Master of Technology from SRM University, Chennai, India in year 2009. He is currently pursuing Ph.D. from Sikkim Manipal University since 2017 and currently working as Assistant Professor (Sr.) in Department of Electronics & Communication Engineering, Surendra Institute of Engineering & Management, MAKAUT (Kolkata) since 10 July 2009. He is a member of IAENG, IRED, ISTE and many more of repute. He has published a Book and more than 20 research papers in reputed international journals including Thomson Reuters and conferences including IEEE and it's also available online. His main research work focuses on Low Power VLSI, Solid State Devices, Sensors and Computational Intelligence based education. He has almost 9 years of teaching experience.

Sukumar Chandra Konar is Professor. He is Ex-Head, Dept. of Electrical Engineering, Indian Institute of Engineering Science & Technology (IIEST), Shibpur, India. He pursued his Bachelor of Engineering from Calcutta University, earned his Master of Engineering from Calcutta University, India. He earned his Ph.D. from Indian Institute of Engineering Science & Technology (IIEST), Shibpur. He has published many research papers in reputed international journals including Thomson Reuters (SCI) and conferences including IEEE. His main research work focuses on Power Systems, Robust Stability. He has almost 35 years of teaching experience. He is a member of Institute of Engineers (India), ISTE and many more of repute.

Krishanu Mitra pursued his Bachelor of Technology from Asansol Engg College, WBUT, India in 2010 and Master of Technology from Bengal Inst. Of Technology & Management, Chennai, India in year 2014. He is currently working as Assistant Engineer in Department of Electronics & Communication Engineering, Surendra Institute of Engineering & Management, MAKAUT (Kolkata) since 2010. He has published many research papers in reputed international & national journals and conferences and it's also available online. His main research work focuses on VLSI architecture, Electronic Devices and Systems.