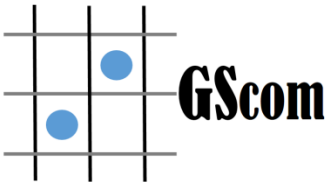


Mapper & De-Mapper System Document

Mapper / De-Mapper

Table of Contents

1. High Level System and Function Block

As can be seen from the Figure 1, high level modem system consists of a transmitter and a receiver. The encoder and modulator in the transmitter receives data from the general purpose multiplexer (GPM) and creates the data frame for transmission through DAC and analog radio frequency devices to antenna. The encoder consists of a low density parity check (LDPC) encoder and a Mapper. The encoder adds low density parity check (LDPC) parity bits to input data bits to make LDPC encoded data format for forward error correction. The LDPC encoded data bits go to a Mapper to map those bits into symbols according to the selected modulation scheme. The symbols are then modulated into real signals (I) and imaginary signals (Q). The digital format of the airframe is converted to analog signals by a digital to analog converter (DAC) and is sent for transmission.

At the receiver, the analog to digital converter (ADC) receives the analog signal, converts the received analog signal to digital signal and sends it to the demodulator block. The digital signals are processed by demodulator, sent to De-mapper for converting the demodulated symbols to bits according to selected modulation scheme, and then sent to the decoder for error correction and for decoding.

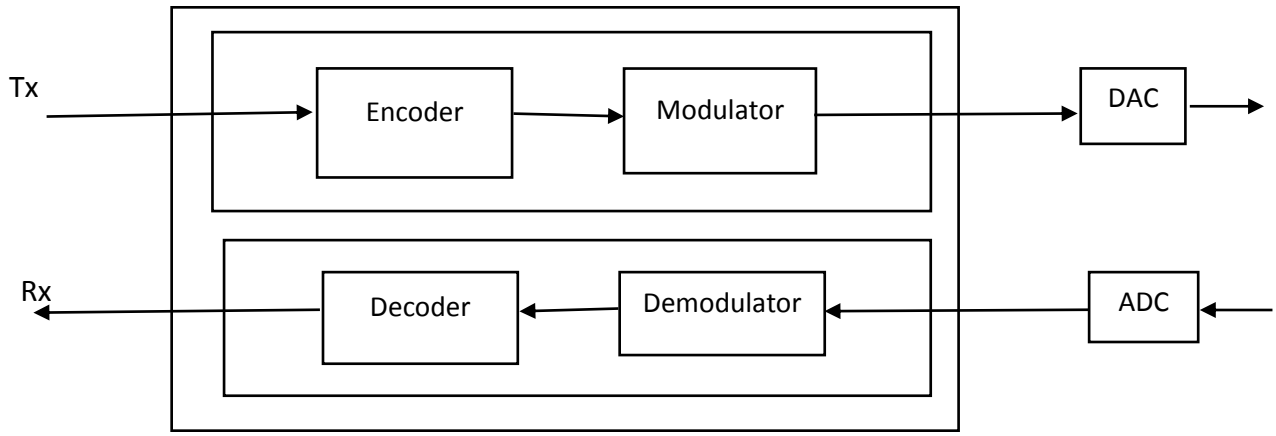


Figure 1, High level modem system

1.1. Mapper

The Mapper module takes a stream of coded bits, divides them into groups of *m* bits, maps each group of *m* bits in one of M (=2^m) signal constellations using Gray code mapping rule, and generates the stream of corresponding symbols as follows;

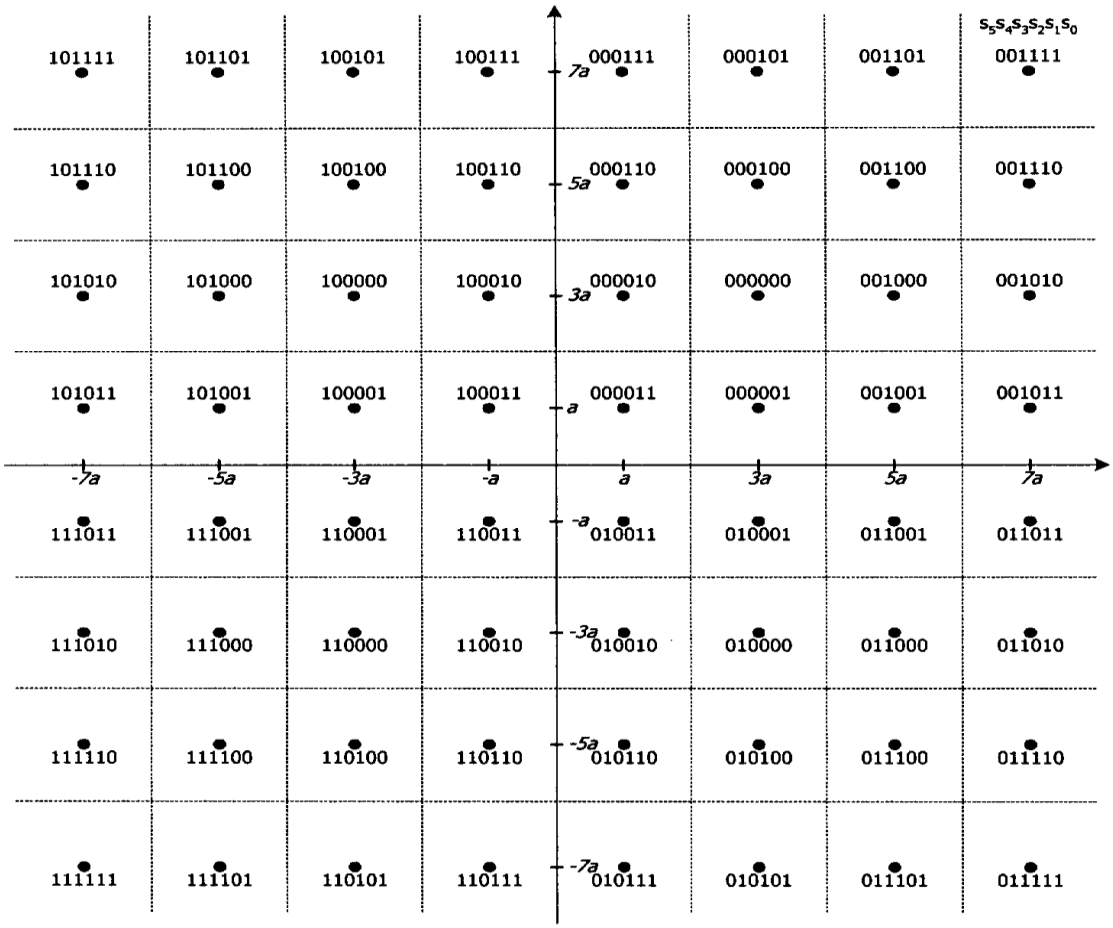
$$s_{k,m-1}s_{k,m-2}s_{k,m-3}.....s_{k,0} \xrightarrow{\text{map}} I_k, Q_k$$

In the above equation, $s_{k,i}$ (i = 0,1,2, …., m-1) represents the i^{th} bit in the k^{th} symbol to be coded by the Gray code mapping, and I_k and Q_k represents the in-phase and quadrature component of the k^{th} symbol, respectively. The Gray mapping for mapping for coded binary bits into symbols can be done with the following recursive formula.

gray (k) = $2^{n-1} + \text{gray}\ (2^n - 1 - k)$, $2^{n-1} \leq \text{k} < 2^n$

with gray (0) = 0, gray (1) = 1, as initial values.

The Mapper is programmable to support different constellations and different modulations up to 1024 QAM. The Mapper supports five standard gray mapped constellations: QPSK, 16QAM, 64QAM, 256QAM, and 1024QAM. In case of 64QAM, as an example, m = 6 and its corresponding signal constellation is as follows;


$S_5S_4S_3S_2S_1S_0$
101111 101101 100101 100111 000111 000101 001101 001111
101110 101100 100100 100110 000110 000100 001100 001110
101010 101000 100000 100010 000010 000000 001000 001010
101011 101001 100001 100011 000011 000001 001001 001011
111011 111001 110001 110011 010011 010001 011001 011011
111010 111000 110000 110010 010010 010000 011000 011010
111110 111100 110100 110110 010110 010100 011100 011110
111111 111101 110101 110111 010111 010101 011101 011111
7a
5a
3a
a
-a
-3a
-5a
-7a
-7a -5a -3a -a a 3a 5a 7a

2. Demodulator Functional Block

The demodulator in the receiver receives a stream of symbol data from ADC (analog to digital converter). The demodulator performs frequency and timing (symbol and frame) synchronization process with keeping AGC (automatic gain control) process on the received symbols. After obtaining the synchronization, the demodulator performs an Equalization process to mitigate channel effect on the received symbols. Then the output of the Equalizer goes to De-mapper (or Slicer) and then goes to the LDPC decoder for data correction.

The following Figure 2 shows the functional block diagram of demodulator.

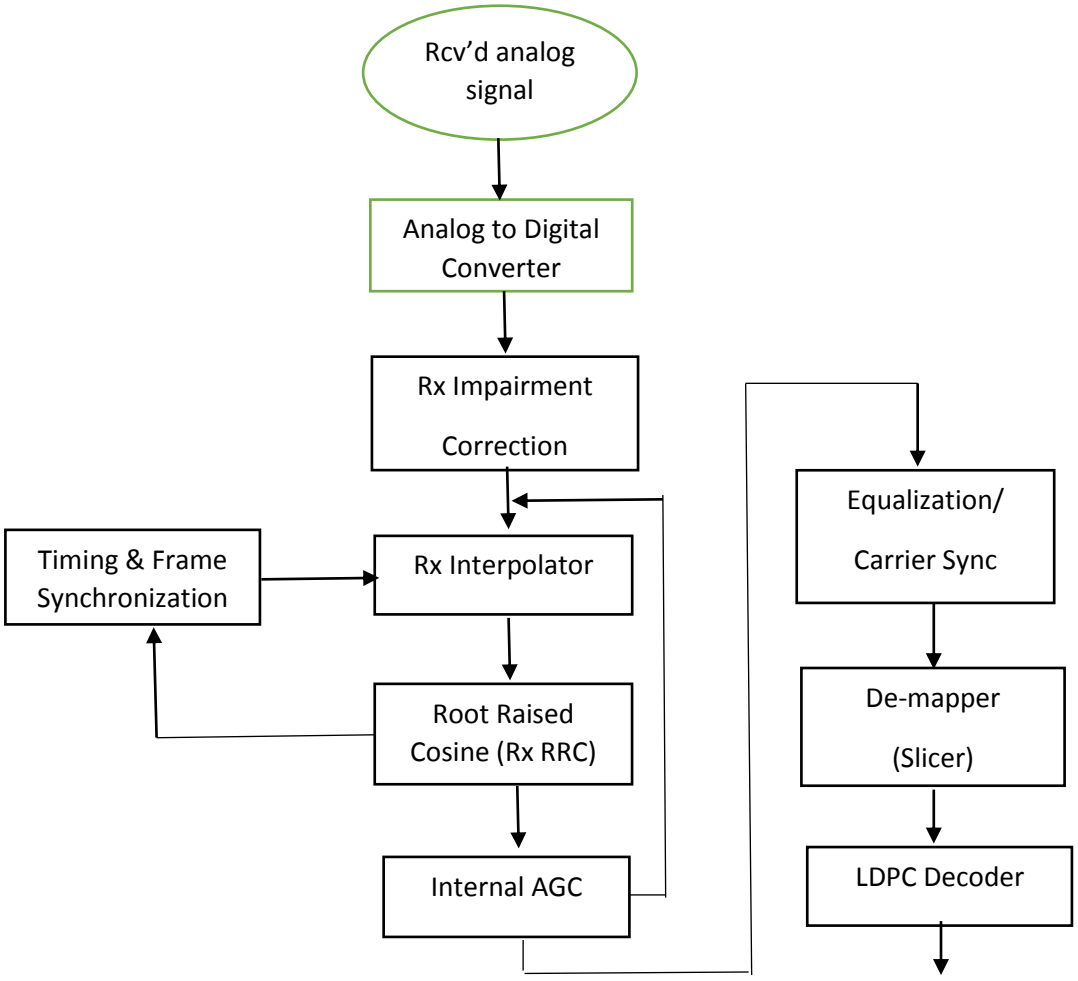


Figure 2: Demodulator Functional Block Diagram

2.1 Decoder Block

The Rx decoder consists of a De-Mapper (or Slicer) and an LDPC decoder. The Rx decoder receives demodulated symbols from demodulator, converts the demodulated symbols to bit stream through a de-mapping process, and then performs the error correction on the received bit streams of data.

2.1.1. De-Mapper

The Mapper module takes the coded bits and maps the bit stream into symbol streams. The Mapper is programmable to support different constellations and different modulations up to 1024QAM. The Mapper supports five standard gray mapped constellations: QPSK, 16QAM, 64QAM, 256QAM, and 1024QAM.

The modulator in the transmitter receives a stream of symbol data from Mapper. The symbols are then modulated into real signals (I) and imaginary signals (Q). The digital format of the airframe is converted to analog signals by digital to analog converter (DAC) and is sent for transmission.

Receiver receives the transmitted signal through an analog to digital converter (ADC) and demodulates the received signal symbols. The De-Mapper receives the demodulated symbols from demodulator, converts the received symbols into the soft values of the corresponding bit stream through a soft decision decoding process. The De-Mapper output signals are the soft decision value of the bit stream of the corresponding symbols. As it is well known, LDPC provides its best performance with the soft value as its input.

We can represent the demodulated received signal, R_k, as follows;

$$R_k = X_k + jY_k = g_k(I_k + jQ_k) + (n_k^I + jn_k^Q)$$

where X_k and Y_k are the in-phase and quadrature components of modulated signal, respectively, g_k is the transmission path gain of the signal, n_k^I and n_k^Q are independently distributed zero mean white Gaussian noise.

The estimation of bit, $S_{k,i}$ (i = 0, 1, 2 … m), in symbol S_k is obtained by taking the log likelihood ratio (LLR) of the conditional probability of the bit being correct, as follows;

$$\Lambda(S_{k,i}) = \log \frac{\Pr\{ S_{k,i}=0 \mid X_k, Y_k \}}{\Pr\{ S_{k,i}=1 \mid X_k, Y_k \}}, \quad i = 0, 1, 2, \ldots, m-1$$

The above LLR can be approximated by dual minimum metric approximation as follows;

[Equation 1]

$$\Lambda(S_{k,i}) = \log \frac{\sum_{z_i} \exp\{ -1/\sigma_n^2 [R_k - z_k (S_{k,i}=0)]^2 \}}{\sum_{z_i} \exp\{ -1/\sigma_n^2 [R_k - z_k (S_{k,i}=1)]^2 \}} .$$

$$\approx \{ \min [R_k - z_k(S_{k,i} = 1)]^2 - \min [R_k - z_k(S_{k,i} = 0)]^2 \} .$$

$$= (2n_{k,i} - 1) \{ [R_k - z_k(S_{k,i} = n_{k,i})]^2 - \min [R_k - z_k(S_{k,i} = \bar{n}_{k,i})]^2 \}$$

Where $z_k(S_{k,i}=0)$ and $z_k(S_{k,i}=1)$ are the values of I_k+jQ_k when $S_{k,i}$=0 and $S_{k,i}$=1, respectively. In the above equation, $n_{k,i}$ represent the value of the i^{th} bit of the symbol which is the closest to the received symbol, R_k. The $\bar{n}_{k,i}$ represent the negative value of $n_{k,i}$.

As can be seen from the above equation, the LLR can be calculated by finding the values of $z_k(S_{k,i} = 1)$ and $z_k(S_{k,i} = 0)$ which minimize the values of $[R_k - z_k(S_{k,i} = 1)]^2$ and $[R_k - z_k(S_{k,i} = 0)]^2$, respectively. The values of $z_k(S_{k,i} = 1)$ and $z_k(S_{k,i} = 0)$ which minimize the values of $[R_k - z_k(S_{k,i} = 1)]^2$ and $[R_k - z_k(S_{k,i} = 0)]^2$ can be determined by the range of the values of the in-phase and quadrature component of the received symbol, R_k.

The first term in the third line in [Equation 1] can be represented using the in-phase and quadrature component of received symbol (or signal) as follows;

[Equation 2]

$$[R_k - z_k(S_{k,i} = n_{k,i})]^2 = (X_k - U_k)^2 + (Y_k - V_k)^2$$

The U_k and V_k are the in-phase and quadrature components of received symbol on a signal constellation, respectively.

The second term in the third line in [Equation 1] can be represented using the in-phase and quadrature component of received symbol (or signal) as follows;

[Equation 3]

$$\min [R_k - z_k(S_{k,i} = \bar{n}_{k,i})]^2 = (X_k - U_{k,i})^2 + (Y_k - V_{k,i})^2$$

The $U_{k,i}$ and $V_{k,i}$ are the in-phase and quadrature components of received symbol on signal constellation, respectively, which minimize $[R_k - z_k(S_{k,i} = \bar{n}_{k,i})]^2$.

Using the [Equation 2] and [Equation 3] in [Equation 1], the LLR can be obtained as follows;

[Equation 4]

$$\Lambda(S_{k,i}) = (2n_{k,i}-1)\,[\,\{(X_k-U_k)^2 + (Y_k-V_k)^2\} - \{(X_k-U_{k,i})^2 + (Y_k-V_{k,i})^2\}\,] \quad .$$

$$= (2n_{k,i}-1)\,[\,(U_k+U_{k,i} - 2X_k)(U_k- U_{k,i}) + (V_k+V_{k,i} - 2Y_k)(V_k- V_{k,i})\,]$$

We can use [Equation 4] to derive the soft decision value of the LLR of any M-ary QAM for the input of the decoder. As we support from 4QAM up to 1024QAM in our modem, we need to derive 5 sets of those soft decision value of the LLRs for those of 4QAM, 16QAM, 64QAM, 256QAM, and 1024QAM.

As can be seen in [Equation 4], we need to find the values of U_k, $U_{k,i}$, V_k, and $V_{k,i}$ for the corresponding received symbol $S_{k,i}$ (= X_k + jY_k) as a function of bit position of i (= 0, 1, ….., *m-1*) to obtain the soft decision value of the LLR, $\Lambda(S_{k,i})$, of received symbol.

LLR Derivation For 256QAM

We illustrate the derivation methodology for the soft decision value of the LLR for 256QAM as an example. For the 256QAM, the following two tables (Table 1 & Table 2) show the range of the quadrature component Y_k and in-phase component X_k of the received symbol R_k, their corresponding bit values, and their corresponding representative values of V_k and U_k in those ranges.

[Table 1]

Y_k	$(n_{k,7}\ n_{k,6}\ n_{k,5}\ n_{k,4})$	V_k
$Y_k > 14a$	(0000)	15a
$12a < Y_k < 14a$	(0001)	13a
$10a < Y_k < 12a$	(0011)	11a
$8a < Y_k < 10a$	(0010)	9a
$6a < Y_k < 8a$	(0110)	7a
$4a < Y_k < 6a$	(0111)	5a
$2a < Y_k < 4a$	(0101)	3a
$0 < Y_k < 2a$	(0100)	a
$-2a < Y_k < 0$	(1100)	-a
$-4a < Y_k < -2a$	(1101)	-3a
$-6a < Y_k < -4a$	(1111)	-5a
$-8a < Y_k < -6a$	(1110)	-7a
$-10a < Y_k < -8a$	(1010)	-9a
$-12a < Y_k < -10a$	(1011)	-11a
$-14a < Y_k < -12a$	(1001)	-13a
$Y_k < -14a$	(1000)	-15a

[Table 2]

X_k	$(n_{k,3}\ n_{k,2}\ n_{k,1}\ n_{k,0})$	U_k
$X_k > 14a$	(0000)	15a
$12a < X_k < 14a$	(0001)	13a
$10a < X_k < 12a$	(0011)	11a
$8a < X_k < 10a$	(0010)	9a
$6a < X_k < 8a$	(0110)	7a
$4a < X_k < 6a$	(0111)	5a
$2a < X_k < 4a$	(0101)	3a
$0 < X_k < 2a$	(0100)	a
$-2a < X_k < 0$	(1100)	-a
$-4a < X_k < -2a$	(1101)	-3a
$-6a < X_k < -4a$	(1111)	-5a
$-8a < X_k < -6a$	(1110)	-7a
$-10a < X_k < -8a$	(1010)	-9a
$-12a < X_k < -10a$	(1011)	-11a
$-14a < X_k < -12a$	(1001)	-13a
$X_k < -14a$	(1000)	-15a

The following Table 3 shows the bit sequence $\{m_{k,7}, m_{k,6}, \ldots\ldots, m_{k,2}, m_{k,1}, m_{k,0}\}$ in the symbol which minimize $[R_k - z_k(S_{k,i} = \bar{n}_{k,i})]^2$ for each bit position of i (i=0,1,2,….,7) , their representation in terms of $\{n_{k,7}, n_{k,6}, \ldots\ldots, n_{k,2}, n_{k,1}, n_{k,0}\}$, and their corresponding in-phase component $U_{k,i}$ and the quadrature component $V_{k,i}$ of the symbol $z_k(S_{k,i} = \bar{n}_{k,i})$.

[Table 3]

i	$\{ m_{k,7}\ m_{k,6}\ m_{k,5}\ m_{k,4}\ m_{k,3}\ m_{k,2}\ m_{k,1}\ m_{k,0} \}$	$V_{k,i}$	$U_{k,i}$
7	$\{ \bar{n}_{k,7}\ 1\ 0\ 0\ \ n_{k,3}\ n_{k,2}\ n_{k,1}\ n_{k,0} \}$	$V_{k,7}$	U_k
6	$\{ n_{k,7}\ \bar{n}_{k,6}\ 1\ 0\ \ n_{k,3}\ n_{k,2}\ n_{k,1}\ n_{k,0} \}$	$V_{k,6}$	U_k
5	$\{ n_{k,7}\ n_{k,6}\ \bar{n}_{k,5}\ 1\ \ n_{k,3}\ n_{k,2}\ n_{k,1}\ n_{k,0} \}$	$V_{k,5}$	U_k
4	$\{ n_{k,7}\ n_{k,6}\ n_{k,5}\ \bar{n}_{k,4}\ \ n_{k,3}\ n_{k,2}\ n_{k,1}\ n_{k,0} \}$	$V_{k,4}$	U_k
3	$\{ n_{k,7}\ n_{k,6}\ n_{k,5}\ n_{k,4}\ \ \bar{n}_{k,3}\ 1\ 0\ 0 \}$	V_k	$U_{k,3}$
2	$\{ n_{k,7}\ n_{k,6}\ n_{k,5}\ n_{k,4}\ \ n_{k,3}\ \bar{n}_{k,2}\ 1\ 0 \}$	V_k	$U_{k,2}$
1	$\{ n_{k,7}\ n_{k,6}\ n_{k,5}\ n_{k,4}\ \ n_{k,3}\ n_{k,2}\ \bar{n}_{k,1}\ 1 \}$	V_k	$U_{k,1}$
0	$\{ n_{k,7}\ n_{k,6}\ n_{k,5}\ n_{k,4}\ \ n_{k,3}\ n_{k,2}\ n_{k,1}\ \bar{n}_{k,0} \}$	V_k	$U_{k,0}$

The values of $V_{k,i}$ and $U_{k,i}$ obtained by using the values of $(m_{k,7}, m_{k,6}, m_{k,5}, m_{k,4})$ and $(m_{k,3}, m_{k,2}, m_{k,1}, m_{k,0})$, respectively, as a function of bit position i in the above table are shown in the next two tables (Table 4 & Table 5) as a function of the value of bits, $n_{k,i}$ ($i = 0, 1, \ldots\ldots, m-1$).

[Table 4]

$(n_{k,7}\ n_{k,6}\ n_{k,5}\ n_{k,4})$	$V_{k,7}$	$V_{k,6}$	$V_{k,5}$	$V_{k,4}$
(0000)	-a	7a	11a	13a
(0001)	-a	7a	11a	15a
(0011)	-a	7a	13a	9a
(0010)	-a	7a	13a	11a
(0110)	-a	9a	3a	5a
(0111)	-a	9a	3a	7a
(0101)	-a	9a	5a	a
(0100)	-a	9a	5a	3a
(1100)	a	-9a	-5a	-3a
(1101)	a	-9a	-5a	-a
(1111)	a	-9a	-3a	-7a
(1110)	a	-9a	-3a	-5a
(1010)	a	-7a	-13a	-11a
(1011)	a	-7a	-13a	-9a
(1001)	a	-7a	-11a	-15a
(1000)	a	-7a	-11a	-13a

[Table 5]

$(n_{k,3}\ n_{k,2}\ n_{k,1}\ n_{k,0})$	$U_{k,3}$	$U_{k,2}$	$U_{k,1}$	$U_{k,0}$
(0000)	-a	7a	11a	13a
(0001)	-a	7a	11a	15a
(0011)	-a	7a	13a	9a
(0010)	-a	7a	13a	11a
(0110)	-a	9a	3a	5a
(0111)	-a	9a	3a	7a
(0101)	-a	9a	5a	a
(0100)	-a	9a	5a	3a
(1100)	a	-9a	-5a	-3a
(1101)	a	-9a	-5a	-a
(1111)	a	-9a	-3a	-7a
(1110)	a	-9a	-3a	-5a
(1010)	a	-7a	-13a	-11a
(1011)	a	-7a	-13a	-9a
(1001)	a	-7a	-11a	-15a
(1000)	a	-7a	-11a	-13a

The values of the $\Lambda(S_{k,i})$ of the received symbol obtained by using the values of the U_k, $U_{k,i}$, V_k, and $V_{k,i}$ from the above tables ([Table 1], [Table 2], [Table 4], [Table 5]) in [Equation 4] are shown in the following two tables ([Table 6] & [Table 7]) as a function of the value of bits, $n_{k,i}$ ($i = 0, 1, \ldots, m-1$).

[Table 6]

Y_k Range	$\Lambda(S_{k,7})$	$\Lambda(S_{k,6})$	$\Lambda(S_{k,5})$	$\Lambda(S_{k,4})$
$Y_k > 14a$	$8Y_k$ - 56a	$4Y_k$ - 44a	$2Y_k$ - 26a	Y_k - 14a
$12a < Y_k < 14a$	$7Y_k$ - 42a	$3Y_k$ - 30a	Y_k - 12a	Y_k - 14a
$10a < Y_k < 12a$	$6Y_k$ - 30a	$2Y_k$ - 18a	Y_k - 12a	$-Y_k$ + 10a
$8a < Y_k < 10a$	$5Y_k$ - 20a	Y_k - 8a	$2Y_k$ - 22a	$-Y_k$ + 10a
$6a < Y_k < 8a$	$4Y_k$ - 12a	Y_k - 8a	$-2Y_k$ + 10a	Y_k - 6a
$4a < Y_k < 6a$	$3Y_k$ - 6a	$2Y_k$ - 14a	$-Y_k$ + 4a	Y_k - 6a
$2a < Y_k < 4a$	$2Y_k$ - 2a	$3Y_k$ - 18a	$-Y_k$ + 4a	$-Y_k$ + 2a
$0 < Y_k < 2a$	Y_k	$4Y_k$ - 20a	$-2Y_k$ + 6a	$-Y_k$ + 2a
$-2a < Y_k < 0$	Y_k	$-4Y_k$ - 20a	$2Y_k$ + 6a	Y_k + 2a
$-4a < Y_k < -2a$	$2Y_k$ + 2a	$-3Y_k$ - 18a	Y_k + 4a	Y_k + 2a
$-6a < Y_k < -4a$	$3Y_k$ + 6a	$-2Y_k$ - 14a	Y_k + 4a	$-Y_k$ - 6a
$-8a < Y_k < -6a$	$4Y_k$ + 12a	$-Y_k$ - 8a	$2Y_k$ + 10a	$-Y_k$ - 6a
$-10a < Y_k < -8a$	$5Y_k$ + 20a	$-Y_k$ - 8a	$-2Y_k$ - 22a	Y_k + 10a
$-12a < Y_k < -10a$	$6Y_k$ + 30a	$-2Y_k$ - 18a	$-Y_k$ - 12a	Y_k + 10a
$-14a < Y_k < -12a$	$7Y_k$ + 42a	$-3Y_k$ - 30a	$-Y_k$ - 12a	$-Y_k$ - 14a
$Y_k < -14a$	$8Y_k$ + 56a	$-4Y_k$ - 44a	$-2Y_k$ - 26a	$-Y_k$ - 14a

[Table 7]

X_k Range	$\Lambda(S_{k,3})$	$\Lambda(S_{k,2})$	$\Lambda(S_{k,1})$	$\Lambda(S_{k,0})$
$X_k > 14a$	$8X_k$ - 56a	$4X_k$ - 44a	$2X_k$- 26a	X_k - 14a
$12a < X_k < 14a$	$7X_k$ - 42a	$3X_k$ - 30a	X_k- 12a	X_k - 14a
$10a < X_k < 12a$	$6X_k$ - 30a	$2X_k$ - 18a	X_k - 12a	$-X_k$ + 10a
$8a < X_k < 10a$	$5X_k$ - 20a	X_k - 8a	$2X_k$ - 22a	$-X_k$ + 10a
$6a < X_k < 8a$	$4X_k$ - 12a	X_k - 8a	-$2X_k$ + 10a	X_k - 6a
$4a < X_k < 6a$	$3X_k$- 6a	X_k- 14a	-X_k + 4a	X_k - 6a
$2a < X_k < 4a$	$2X_k$ - 2a	$3Y_k$ - 18a	-X_k + 4a	-X_k + 2a
$0 < X_k < 2a$	X_k	$4X_k$- 20a	-$2X_k$ + 6a	-X_k + 2a
$-2a < X_k < 0$	X_k	$-4X_k$ - 20a	$2X_k$ + 6a	X_k + 2a
$-4a < X_k < -2a$	$2X_k$ + 2a	$-3Y_k$ - 18a	X_k + 4a	X_k + 2a
$-6a < X_k < -4a$	$3X_k$ + 6a	$-2Y_k$ - 14a	X_k + 4a	$-X_k$ - 6a
$-8a < X_k < -6a$	$4X_k$ + 12a	$-X_k$ - 8a	$2X_k$ + 10a	$-X_k$ - 6a
$-10a < X_k < -8a$	$5X_k$ + 20a	$-X_k$ - 8a	-$2X_k$ - 22a	X_k + 10a
$-12a < X_k < -10a$	$6X_k$ + 30a	$-2X_k$ - 18a	$-X_k$ - 12a	X_k + 10a
$-14a < X_k < -12a$	$7X_k$ + 42a	-$3X_k$ - 30a	$-X_k$ - 12a	$-X_k$ - 14a
$X_k < -14a$	$8X_k$ + 56a	-$4X_k$ - 44a	$-2X_k$ - 26a	$-X_k$ - 14a

We can obtain the LLR of the $\Lambda(S_{k,i})$ by applying the following transform variable to [Table 6] and [Table 7] which we do not need lookup table or memory for the calculation of the $\Lambda(S_{k,i})$. The transform variable used are as follows;

$Z_{1k} = |Y_k| - 8a,\ Z_{2k} = |Z_{1k}| - 4a,\ Z_{3k} = |Z_{2k}| - 2a,$

$Z'_{1k} = |X_k| - 8a,\ Z'_{2k} = |Z'_{1k}| - 4a,\ Z'_{3k} = |Z'_{2k}| - 2a,$

Under the assumption of the possibility of using sign bit to represent the sign of those value of transform variables (X_k, Y_k, Z_{1k}, Z_{2k}, Z_{3k}, Z'_{1k}, Z'_{2k}, Z'_{3k}), we can obtain the LLR tables for the calculation of $\Lambda(S_{k,i})$ of the received symbol obtained by representing the values of LLR in terms of the transform variables in the [Table 6] and [Table 7]. The resulting tables are shown in [Table 8] and [Table 9], respectively.

[Table 8]

MSB(Y_k)	MSB(Z_{1k})	MSB(Z_{2k})	MSB(Z_{3k})	$\Lambda(S_{k,7})$	$\Lambda(S_{k,6})$	$\Lambda(S_{k,5})$	$\Lambda(S_{k,4})$

0	0	0	0	$Y_k + 7Z_{1k}$	$Y_k - 2Z_{1k} + 5Z_{2k}$	$Z_{2k} + Z_{3k}$	Z_{3k}
			1	$Y_k + 7Z_{1k} - Z_{3k}$	$Y_k - Z_{2k} + 3Z_{3k}$	Z_{2k}	Z_{3k}
		1	0	$3Y_k - 2Z_{3k}$	Z_{1k}	$Z_{2k} - Z_{3k}$	Z_{3k}
			1	$3Y_k - 3Z_{3k}$	$Z_{1k} - Z_{3k}$	Z_{2k}	Z_{3k}
	1	0	0	Y_k	$2Z_{1k} - 2Z_{3k}$	$Z_{2k} + Z_{3k}$	Z_{3k}
			1	$Y_k - Z_{3k}$	$2Z_{1k} - Z_{3k}$	Z_{2k}	Z_{3k}
		1	0	$2Y_k + 2Z_{3k}$	Z_{1k}	$Z_{2k} - Z_{3k}$	Z_{3k}
			1	$2Y_k + Z_{3k}$	$Z_{1k} + Z_{3k}$	Z_{2k}	Z_{3k}
1	0	0	0	$Y_k - 7Z_{1k}$	$-Y_k - 2Z_{1k} + 5Z_{2k}$	$Z_{2k} + Z_{3k}$	Z_{3k}
			1	$Y_k - 7Z_{1k} + Z_{3k}$	$-Y_k - Z_{2k} + 3Z_{3k}$	Z_{2k}	Z_{3k}
		1	0	$3Y_k + 2Z_{3k}$	Z_{1k}	$Z_{2k} - Z_{3k}$	Z_{3k}
			1	$3Y_k + 3Z_{3k}$	$Z_{1k} - Z_{3k}$	Z_{2k}	Z_{3k}
	1	0	0	Y_k	$2Z_{1k} - 2Z_{3k}$	$Z_{2k} + Z_{3k}$	Z_{3k}
			1	$Y_k + Z_{3k}$	$2Z_{1k} - Z_{3k}$	Z_{2k}	Z_{3k}
		1	0	$2Y_k - 2Z_{3k}$	Z_{1k}	$Z_{2k} - Z_{3k}$	Z_{3k}
			1	$2Y_k - Z_{3k}$	$Z_{1k} + Z_{3k}$	Z_{2k}	Z_{3k}

[Table 9]

MSB(X_k)	MSB(Z'_{1k})	MSB(Z'_{2k})	MSB(Z'_{3k})	$\Lambda(S_{k,3})$	$\Lambda(S_{k,2})$	$\Lambda(S_{k,1})$	$\Lambda(S_{k,0})$
0	0	0	0	$X_k + 7Z'_{1k}$	$X_k - 2Z'_{1k} + 5Z'_{2k}$	$Z'_{2k} + Z'_{3k}$	Z'_{3k}
			1	$X_k + 7Z'_{1k} - Z'_{3k}$	$X_k - Z'_{2k} + 3Z'_{3k}$	Z'_{2k}	Z'_{3k}
		1	0	$3X_k - 2Z'_{3k}$	Z'_{1k}	$Z'_{2k} - Z'_{3k}$	Z'_{3k}
			1	$3X_k - 3Z'_{3k}$	$Z'_{1k} - Z'_{3k}$	Z'_{2k}	Z'_{3k}
	1	0	0	X_k	$2Z'_{1k} - 2Z'_{3k}$	$Z'_{2k} + Z'_{3k}$	Z'_{3k}
			1	$X_k - Z'_{3k}$	$2Z'_{1k} - Z'_{3k}$	Z'_{2k}	Z'_{3k}
		1	0	$2X_k + 2Z'_{3k}$	Z'_{1k}	$Z'_{2k} - Z'_{3k}$	Z'_{3k}
			1	$2X_k + Z'_{3k}$	$Z'_{1k} + Z'_{3k}$	Z'_{2k}	Z'_{3k}
1	0	0	0	$X_k - 7Z'_{1k}$	$-X_k - 2Z'_{1k} + 5Z'_{2k}$	$Z'_{2k} + Z'_{3k}$	Z'_{3k}
			1	$X_k - 7Z'_{1k} + Z'_{3k}$	$-X_k - Z'_{2k} + 3Z'_{3k}$	Z'_{2k}	Z'_{3k}
		1	0	$3X_k + 2Z'_{3k}$	Z'_{1k}	$Z'_{2k} - Z'_{3k}$	Z'_{3k}
			1	$3X_k + 3Z'_{3k}$	$Z'_{1k} - Z'_{3k}$	Z'_{2k}	Z'_{3k}
	1	0	0	Y_k	$2Z'_{1k} - 2Z'_{3k}$	$Z'_{2k} + Z'_{3k}$	Z'_{3k}
			1	$X_k + Z'_{3k}$	$2Z'_{1k} - Z'_{3k}$	Z'_{2k}	Z'_{3k}

		1	0	$2X_k - 2Z'_{3k}$	Z'_{1k}	$Z'_{2k} - Z'_{3k}$	Z'_{3k}
			1	$2X_k - Z'_{3k}$	$Z'_{1k} + Z'_{3k}$	Z'_{2k}	Z'_{3k}

The soft decision value of LLR for the upper 4 bits ($\Lambda(S_{k,7})$, $\Lambda(S_{k,6})$, $\Lambda(S_{k,5})$, $\Lambda(S_{k,4})$) can be obtained by exploiting the information in [Table 8] as follows;

$\Lambda(S_{k,7}) = \alpha Y_k + \delta(\beta Z_{1k} + \gamma Z_{3k})$

$\Lambda(S_{k,6}) = aY_k + bZ_{1k} + cZ_{2k} + dZ_{3k}$

$\Lambda(S_{k,5}) = Z_{2k} + fZ_{3k}$

$\Lambda(S_{k,4}) = Z_{3k}$

Where the value of α, δ, β, γ, a, b, c, d, and f are coefficient constants and can be obtained by exploiting the information in [Table 8].

The soft decision value of LLR for the lower 4 bits ($\Lambda(S_{k,3})$, $\Lambda(S_{k,2})$, $\Lambda(S_{k,1})$, $\Lambda(S_{k,0})$) can be obtained by exploiting the information in [Table 9] as follows;

$\Lambda(S_{k,3}) = \alpha' X_k + \delta'(\beta' Z'_{1k} + \gamma' Z'_{3k})$

$\Lambda(S_{k,2}) = \alpha' X_k + b' Z'_{1k} + c' Z'_{2k} + d' Z'_{3k}$

$\Lambda(S_{k,1}) = Z'_{2k} + f' Z'_{3k}$

$\Lambda(S_{k,0}) = Z'_{3k}$

Where the value of α', δ', β', γ', a', b', c′, d′, and f′ are coefficient constants and can be obtained by exploiting the information in [Table 9].

2.1.2. Implementation of De-Mapper into Hardware

As can be seen from the above LLR equation, the soft decision value of LLR of one symbol for 256QAM can be obtained using six bit clocks. One symbol in 256-QAM consists of eight bits. The de-mapping process developed here is a real time process. We use the same principle to derive real time mapping & de-mapping processors for QPSK, 16QAM, 64QAM, and 1024QAM.