(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 December 2009 (10.12.2009)

PCT

(10) International Publication Number
WO 2009/149429 A2

(51) International Patent Classification:
H04B 3/54 (2006.01) *H04L 27/26* (2006.01)

(21) International Application Number:
PCT/US2009/046516

(22) International Filing Date:
5 June 2009 (05.06.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/059,717 6 June 2008 (06.06.2008) US
12/478,699 4 June 2009 (04.06.2009) US

(71) Applicant *(for all designated States except US)*: **MAXIM INTEGRATED PRODUCTS, INC.** [US/US]; 120 San Gabriel Drive, Sunnyvale, CA 94086 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **RAZAZIAN, Kaveh** [US/US]; 8 Hazelbranch, Aliso Viejo, CA 92656 (US). **UMARI, Maher** [US/US]; 25 Palatine #421, Irvine, CA 92612 (US). **LOGINOV, Victor, V.** [RU/US]; 1019 Oak Glen, Irvine, CA 92618-92603 (US).

(74) Agents: **VINCENT, Lester, J.** et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 1279 Oakmead Parkway, Sunnyvale, CA 94085-4040 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

WO 2009/149429 A2

(54) Title: ROBUST WIDEBAND SYMBOL AND FRAME SYNCHRONIZER FOR POWER-LINE COMMUNICATION

(57) Abstract: An embodiment is a method and apparatus to perform symbol synchronization. A sign element obtains signs of samples in a sample vector. A correlation estimator computes a correlation of the sample vector. A synchronization detector detects symbol synchronization. Another embodiment is a method and apparatus to perform frame synchronization. A Fast Fourier Transform (FFT) processing unit computes a current FFT vector and an accumulated previous FFT vector. The current FFT vector and the accumulated previous FFT vector correspond to sample vectors associated with preamble symbols prior to symbol synchronization detection. A real and imaginary processing unit generates real and imaginary summations using the current FFT vector and the accumulated previous FFT vector. A mode processor generates mode flags representing operational modes using the real and imaginary summations.

ROBUST WIDEBAND SYMBOL AND FRAME SYNCHRONIZER FOR POWER-LINE COMMUNICATION

RELATED APPLICATIONS

[001] This application claims the benefits of the provisional application, filed on June 6, 2008, titled "ROBUST WIDEBAND SYMBOL AND FRAME SYNCHRONIZER FOR POWER-LINE COMMUNICATION", Serial No. 61/059,717.

TECHNICAL FIELD

[002] The presently disclosed embodiments are directed to the field of communication, and more specifically, to power line communication.

BACKGROUND

[003] Power line communication (PLC) is a communication technology to carry data on electrical media (e.g., wires) that are used for electrical power transmission. Typically, electrical power is transmitted over high voltage transmission lines, distributed over medium voltage, and used inside commercial or residential buildings at lower voltages. Since power line networks transmit data signals over the same electrical grid as that is used for carrying electrical power to commercial or residential buildings, electrical wires and sockets are used simultaneously for electricity and for data transmission, without causing disruption to either.

[004] Broadband technologies provide high speed data transmission. However, currently it is problematic to apply broadband technologies in PLC. Some problems include the ability to efficiently decode signals in noisy channels, achieve time and frequency diversity, remove signal interference, maintain received signals at pre-determined levels, measure channel quality for high transmission rate, provide robustness to wideband and narrow band symbol synchronization.

SUMMARY

[005] One disclosed feature of the embodiments is a method and apparatus to perform symbol synchronization. A sign element obtains signs of samples in a sample vector. A correlation estimator computes a correlation of the sample vector. A synchronization detector detects symbol synchronization. Another embodiment is a method and apparatus to perform frame synchronization. A Fast Fourier Transform (FFT) processing unit computes

a current FFT vector and an accumulated previous FFT vector. The current FFT vector and the accumulated previous FFT vector correspond to sample vectors associated with preamble symbols prior to symbol synchronization detection. A real and imaginary processing unit generates real and imaginary summations using the current FFT vector and the accumulated previous FFT vector. A mode processor generates mode flags representing operational modes using the real and imaginary summations.

BRIEF DESCRIPTION OF THE DRAWINGS

[006] Embodiments may best be understood by referring to the following description and accompanying drawings that are used to illustrate various embodiments. In the drawings.

[007] Figure 1 is a diagram illustrating a data frame structure used for data transmission and for the FCC, ARIB and CENELEC A bands according to one embodiment.

[008] Figure 2 is a diagram illustrating a symbol duration for data symbol according to one embodiment.

[009] Figure 3 is a diagram illustrating a data frame structure for data transmission for CENELECs B, C and BC according to one embodiment.

[0010] Figure 4 is a diagram illustrating a symbol duration for data symbol for CENELC B and C according to one embodiment.

[0011] Figure 5 is a diagram illustrating ACK signal for FCC, ARIB and CENELEC A according to one embodiment.

[0012] Figure 6 is a diagram illustrating ACK signal for CENELEC B, C, and BC according to one embodiment.

[0013] Figure 7 is a diagram illustrating a base-band transmitter according to one embodiment.

[0014] Figure 8 is a diagram illustrating the FEC encoding unit according to one embodiment.

[0015] Figure 9A is a diagram illustrating the data scrambler according to one embodiment.

[0016] Figure 9B is a diagram illustrating the convolutional encoder according to one embodiment.

[0017] Figure 10 is a diagram illustrating the modulator according to one embodiment.

[0018] Figure 11A is a diagram illustrating the DBPSK modulator according to one embodiment.

[0019] Figure 11B is a diagram illustrating the carrier index numbers according to one embodiment.

[0020] Figure 11C is a diagram illustrating the input/output configuration according to one embodiment.

[0021] Figure 12 is a diagram illustrating the PSD shaping module according to one embodiment.

[0022] Figure 13A is a diagram illustrating a raised cosine function according to one embodiment.

[0023] Figure 13B is a diagram illustrating a overlapping and add operation according to one embodiment.

[0024] Figure 14 is a diagram illustrating a preamble signal according to one embodiment.

[0025] Figure 15 is a diagram illustrating the pre-emphasis filter according to one embodiment.

[0026] Figure 16 is a diagram illustrating the pre-emphasis filter according to one embodiment.

[0027] Figure 17A is a diagram illustrating a data scaler on the transmitter data path according to one embodiment.

[0028] Figure 17B is a diagram illustrating a P and M scaler on the transmitter data path according to one embodiment.

[0029] Figure 17C is a diagram illustrating a scaler for frequency-domain P and M signals according to one embodiment.

[0030] Figure 18 is a diagram illustrating a receiver according to one embodiment.

[0031] Figure 19 is a diagram illustrating the demodulator according to one embodiment.

[0032] Figure 20 is a diagram illustrating the FEC decoding unit according to one embodiment.

[0033] Figure 21 is a diagram illustrating timings associated with events in the receiver according to one embodiment.

[0034] Figure 22 is a diagram illustrating the DC blocker according to one embodiment.

[0035] Figure 23 is a diagram illustrating the FFT according to one embodiment.

[0036] Figure 24 is a diagram illustrating the DBPSK demodulator according to one embodiment.

[0037] Figure 25 is a diagram illustrating the ROBO combiner/decoder according to one embodiment.

[0038] Figure 26 is a diagram illustrating the RS decoder according to one embodiment.

[0039] Figure 27A is a diagram illustrating the signal waveforms according to one embodiment.

[0040] Figure 27B is a diagram illustrating the symbol synchronizer according to one embodiment.

[0041] Figure 28 is a diagram illustrating the moving average filter according to one embodiment.

[0042] Figure 29 is a diagram illustrating the preamble waveform and the autocorrelation function according to one embodiment.

[0043] Figure 30 is a diagram illustrating the correlation estimator according to one embodiment.

[0044] Figure 31 is a diagram illustrating the synchronization detector according to one embodiment.

[0045] Figure 32 is a diagram illustrating the various waveforms in the symbol synchronizer according to one embodiment.

[0046] Figure 33 is a diagram illustrating the frame synchronizer according to one embodiment.

[0047] Figure 34 is a diagram illustrating the FFT processing unit according to one embodiment.

[0048] Figure 35 is a diagram illustrating the real and imaginary processing unit according to one embodiment.

[0049] Figure 36 is a diagram illustrating the constellation associated with symbols P and M in various operational modes according to one embodiment.

[0050] Figure 37 is a diagram illustrating the output when P is followed by noise according to one embodiment.

[0051] Figure 38 is a diagram illustrating the threshold effect on detection criteria according to one embodiment.

[0052] Figure 39 is a diagram illustrating the mode processor according to one embodiment.

DETAILED DESCRIPTION

[0053] One disclosed feature of the embodiments is a method and apparatus to perform symbol synchronization. A sign element obtains signs of samples in a sample vector. A correlation estimator computes a correlation of the sample vector. A synchronization detector detects symbol synchronization. Another embodiment is a method and apparatus to perform frame synchronization. A Fast Fourier Transform (FFT) processing unit computes

a current FFT vector and an accumulated previous FFT vector. The current FFT vector and the accumulated previous FFT vector correspond to sample vectors associated with preamble symbols prior to symbol synchronization detection. A real and imaginary processing unit generates real and imaginary summations using the current FFT vector and the accumulated previous FFT vector. A mode processor generates mode flags representing operational modes using the real and imaginary summations.

[0054] One disclosed feature of the embodiments may be described as a process which is usually depicted as a flowchart, a flow diagram, a structure diagram, or a block diagram. Although a flowchart may describe the operations as a sequential process, many of the operations can be performed in parallel or concurrently. In addition, the order of the operations may be re-arranged. A process is terminated when its operations are completed. A process may correspond to a method, a program, a procedure, a method of manufacturing or fabrication, etc. One embodiment may be described by a schematic drawing depicting a physical structure. It is understood that the schematic drawing illustrates the basic concept and may not be scaled or depict the structure in exact proportions.

[0055] One disclosed feature of the embodiments is the implementation of a data communication modem for Federal Communication Commission (FCC), Association of Radio Industries and Businesses (ARIB), and European Committee of electrotechnical standardization or Comité Européen de Normalisation Electrotecnique (CENELEC) bands over the power line medium. The system may include a highly integrated PHY (physical layer) and MAC (Media Access Control) digital transceiver and analog front end processing circuits. The system is based on Orthogonal Frequency Division Multiplexing (OFDM). OFDM has been chosen as the modulation technique because of its inherent adaptability in the presence of frequency selective channels, its resilience to jammer signals, and its robustness to impulsive noise.

[0056] The OFDM system may place $N_{carrier}$ evenly spaced carriers into a specified frequency band such as from DC to 1.2MHz. In one embodiment, $N_{carrier}=128$. In the following description, the value $N_{carrier}=128$ will be used for illustrative purposes. It is contemplated that $N_{carrier}$ may be any suitable number. Depending on the band selection, number of carriers participating in transporting data varies. Every carrier may be modulated with Differential Binary Phase Shift Keying (DBPSK). The system may support two modes of operation namely Normal and ROBO (Robust OFDM). The ROBO modulation is robust in the sense that it may provide four times extra redundancy parity bits by mean of

repetition code and therefore the system may reliably deliver data under severe channel conditions.

[0057] The system parameters include signal frequency bands, modulation schemes, sampling frequency and physical layer frame structure, etc. The system supports various CELENEC, FCC and ARIB bands. The frequency band associated with each standard is given in Table 1.

	F Low (KHz)	F High (KHz)
FCC	10	480
ARIB	10	450
CELENEC A	9	95
CELENEC B	95	125
CELENEC C	125	140
CELENEC B, C	95	140

Table 1: FCC, ARIB and CENELEC Bands

[0058] An OFDM with DBPSK modulation scheme per carrier may be selected. The OFDM modulation technique is very robust against channel fading, narrowband interference and spike noise. The DBPSK modulation for each carrier may make the receiver design significantly simple since no tracking circuitry is required at the receiver for coherently detecting the phase of each carrier. In DBPSK demodulator, the phases of carriers in adjacent symbol may be taken as the reference for detecting the phases of the carriers in the current symbol. The phases of corresponding carriers in adjacent OFDM symbols may be expected to be stationary since the channel and the clock frequency variations in time are very slow as compared to the duration of two adjacent symbols.

[0059] Assuming the maximum spectral content of the signal is 480KHz, the sampling frequency at the transmitter and receiver may be selected to be 1.2MHz that is about 240KHz above the Nyquist rate to provide a sufficient margin for signal filtering in the transmitter (for PSD shaping to remove the signal images) and at the receiver (for band selection and signal enhancement).

[0060] The number of frequency bins (FFT points) may be any suitable number. In one embodiment, this number is selected to be N = 256. This results in a frequency resolution for the OFDM carriers equal to 4.6875KHz (Fs / N). Note that imperfection such as sampling clock frequency variation may cause Inter Carrier Interference (ICI). In practice, the ICI caused by a typical sampling frequency variation about 2% of frequency resolution is negligible. In other word, considering ±20ppm sampling frequency in transmitter and receiver clocks, the drift of the carriers may be approximately equal to 48Hz that is approximately 1.26% of the selected frequency resolution. Considering these selections, the number of usable carriers for each standard may be obtained as given in Table 2.

	Number of Carriers (Ncarr) (KHz)	First Carrier (KHz)	Last Carrier (KHz)
FCC	100	14.063	478.125
ARIB	93	14.0625	445.3125
CELENEC A	19	9.375	93.75
CELENEC B	6	98.4375	121.875
CELENEC C	3	126.5625	135.9375
CELENEC B, C	9	98.4375	135.9375

Table 2: Number of carriers for various bands

[0061] The system may work in two different modes namely Normal and ROBO modes. In Normal mode, the FEC may include a Reed Solomon encoder and a convolutional encoder. The system may also support Reed Solomon code with parity of 8 and 16 Bytes.

[0062] In ROBO mode (robust modulation scheme) the FEC may include Reed Solomon and convolutional encoders followed by a Repetition Code (RC). The RC code may repeat each bit four times making system more robust to channel impairments. This of course may reduce the throughput by a factor of 4. The FEC parameters for RS parity of 8 may be given in Table 3.

	Normal Mode	ROBO Mode
FCC	½ convolutional Code + Reed Solomon (241/249)	½ convolutional Code + Reed Solomon (53/61)+RC (4)
ARIB	½ convolutional Code + Reed Solomon (223/231)	½ convolutional Code + Reed Solomon (49/57) + RC (4)
CENELEC A	½ convolutional Code + Reed Solomon (181/189)	½ convolutional Code + Reed Solomon (38/46) + RC (4)
CENELEC BC	½ convolutional Code + Reed Solomon (171/179)	½ convolutional Code + Reed Solomon (36/44)+RC (4)
CENELEC B	½ convolutional Code + Reed Solomon (111/119)	½ convolutional Code + Reed Solomon (21/29)+RC (4)
CENELEC C	½ convolutional Code + Reed Solomon (111/119)	½ convolutional Code + Reed Solomon (21/29)+RC (4)

Table 3: FEC Parameters

[0063] The number of symbols in each PHY (Physical Layer) frame may be selected based on two parameters, the required data rate and the acceptable delay. Since high bandwidth standard (FCC, ARIB) may be utilized for some delay sensitive applications such as voice transmission, therefore the number of symbols in PHY frame may be selected less that that of low bandwidth standard (CENELEC). The number of symbols and data rate associated with each band may be tabulated in Table 4. To calculate the data rate, the packets may be assumed to be continuously transmitted with no inter frame time gap.

	Data Rate (DBPSK)(kbps)	Data Rate (ROBO)(kbps)	No. of symbols per PHY Frame **(Ns)**
FCC	170	37	40
ARIB	157	34	40
CELENEC A	37	7.7	160
CELENEC B	9.71	1.84	320
CELENEC C	4.9	0.93	640
CELENEC B, C	14.95	3.15	320

Table 4: Data rate for various standards

[0064] The data rate may be calculated based on the number of symbols per PHY frame (NS), number of carrier per symbol (Ncarr) and number of parity bits added by FEC blocks. As an example, consider the system in the FCC band working in ROBO mode. Total number of bits carried by the whole PHY frame may be equal to:

Total_No_Bits = NS x Ncarr = 40 x 100 = 4000bits

[0065] The number of bits required at the input of ROBO encoder may be given by:

No_Bits_ROBO = 4000 x ROBORate = 4000 x ¼ = 1000bits

[0066] Considering the fact that convolutional encoder may have a rate equal to ½ (CCRrate = ½) and also consider adding CCZerotail = 6bits of zeros to terminate the states of the encoder to all zero states then the maximum number of symbols at the output of Reed Solomon encoder (MAXRSbytes) may be equal to:

MAXRSbytes = floor((No_Bits_ROBO x CCRate x CCZeroTail)/8) = floor((1000 x ½ - 6)/8) = 61

[0067] Symbols: Removing 8 symbols associated with the parity bits, we may obtain:

DataLength = (61 - ParityLength) x 8 = 424 bits

[0068] These 424 bits may be carried within the duration of a PHY frame. The duration of a PHY frame may be calculated by the following formula:

T_Frame = ((NS x (N_CP + N - NO) + (Npre x N)))/Fs

where Npre, N, NO and N_CP are the number of samples in the preamble, FFT length, the number of samples overlapped at each side of one symbol and the number of samples in the cyclic prefix, respectively. The Fs is the sampling frequency. Typical values for all these parameters for various frequency bands may be given in Table 5.

Number of FFT points	N =256
Number of overlapped samples	NO = 8
Number of cyclic Prefix (CENELEC B and C) Number of cyclic Prefix (FCC, ARIB, CENELEC A)	N_CP = 89 N_CP = 30
Sampling frequency	Fs =1.2MHz

Table 5: Parameters for various frequency bands

[0069] Replacing the above numbers in the equation, T-Frame (PHY frame duration) may be obtained as follows:

T_Frame = (40 x (256 + 22) + (9.5 x 256))/1200000 = 0.0112 sec.

[0070] Therefore the data rate may be calculated by:

Data rate = 424 /0.0112 ~ 37kbps

[0071] Signal Types: There are 2 transmission commands to the physical layer as described below.

[0072] Figure 1 is a diagram illustrating a data frame structure 100 used for data transmission and for the FCC, ARIB and CENELEC A bands according to one embodiment. The data frame 100 includes a preamble portion 110 and a data symbol portion 120.

[0073] The preamble 110 may include 8 identical P symbols and 1½ identical M symbols. Each symbol may be 256 samples and may be pre-stored in the transmitter and may be transmitted right before the data symbols. The symbols P may be used for AGC adaptation, symbol synchronization, channel estimation and initial phase reference estimation. For M symbols, two types of symbol may be used. One is the M1 in which all the carriers may be π phase shifted and the other one is M2 in which all the carriers may be $\pi/2$ phase shifted. M1 is used in ROBO mode and M2 may be used in Normal mode. At the receiver, the phase distance between symbol P and symbol M waveforms may be used for frame synchronization purpose. And the distance between the phases of two possible M symbols may be used to detect whether the PHY frame is sent in Normal mode or in ROBO mode.

[0074] Figure 2 is a diagram illustrating a symbol duration for data symbol according to one embodiment. Each symbol may have 8 samples overlapped with adjacent symbols. The last 8 samples (tail) of preamble may also be overlapped with the 8 samples of the first data symbol (head) as shown in the Figure 2. The overlap may be included to smooth the transition between symbols thus reducing the out of band spectral growth.

[0075] Figure 3 is a diagram illustrating a data frame structure 300 for data transmission for CENELECs B, C and BC according to one embodiment. The data frame 300 includes a preamble portion 310 and a data symbol portion 320.

[0076] The preamble 310 for CENELECs B, C & BC bands may include for special symbols labeled as F1F2 symbols, followed by four identical P symbols and 1½ identical M symbols. For CENELEC C, each F1F2 symbol may include three sinewaves whose phases may switch by 180^{o} after 256 samples. Hence, we generate 256 samples of each of the three tones and sum them together, then we add 180^{o} phase shift to each of the three tones and generate another 257 samples, so that the total length of an F1F2 symbol may be 513 samples. For CENELECs B & BC, six tones may be used instead of three, but the length of the F1F2 symbols remains unchanged. The F1F2 symbols may be used for synchronization. Each preamble symbol may contain 513 samples. The reason that we have used a different technique for synchronization is that the allocated bandwidth in CENELECs C, B and BC may be too small, which makes the autocorrelation property of the P symbols not good enough for robust synchronization. As a result, F1F2 symbols may be used. They have much better autocorrelation property. As for the P symbols for narrowband, they may still be used for channel estimation and initial phase reference estimation, same as was the case for wideband. The symbols M1 or M2 proposed for FCC, ARIB and CENELEC standards are also used for narrowband for the same purposes (frame synchronization and mode detection).

[0077] Figure 4 is a diagram illustrating a symbol duration for data symbol for CENELC B and C according to one embodiment. Again, the same approach is used for PHY frame in ROBO mode that is the P and M symbol are exchanged.

ACK/NACK signal

[0078] Figure 5 is a diagram illustrating ACK signal for FCC, ARIB and CENELEC A according to one embodiment. This signal may be used when an acknowledgement is required to confirm whether the data is correctly received (ACK) or it is erroneous (NACK). The same waveform used in preamble with modified M symbol may be used as an ACK

signal. The P with 90 degrees shift (M=jP) and P with 180 degrees shift (M=-P) may already reserved for normal mode and ROBO mode respectively. The P with 270 degrees shift (M=-jP) may be proposed to be used for ACK signaling.

[0079] This may simplifies the system, as only one waveform need to be stored in the transmitter and same detection circuit in the receiver as used for preamble detection, is used for ACK signal detection as well. If no signal is received during the specified period, it is interpreted as a NACK signal.

[0080] Figure 6 is a diagram illustrating ACK signal for CENELEC B, C, and BC according to one embodiment. Again the same symbols as used for the preamble for the purpose of synchronization, may also be used for the ACK signal. During the time period that a device is waiting for an acknowledgement, the reception of this signal may be an indication that the data may have been delivered with no error. If the time expires and the ACK signal has not been received, it may be an indication that the data may have been lost or delivered in errors.

[0081] Figure 7 is a diagram illustrating a base-band transmitter 700 according to one embodiment. The base-band transmitter 700 includes a Forward Error Correction (FEC) encoding unit 710, a modulator 720, a power spectral shaping (PSD) module 730, a switch 740, an output formatter 750, and a switch 760.

[0082] The base-band transmitter 700 may receive its input bits in one packet from the Media Access (MAC) Layer. The FEC encoding unit 710 may include a number of FEC encoders. Each FEC encoder may add parity bits to the data and the packet grows as it goes through various blocks in FEC encoding unit 710. At the end of the FEC encoding unit 710, the final packet may be broken down into small packet so that each small packet may be fitted into one OFDM symbol. The size of one small packet depends on the number of carriers used in each OFDM symbol. For example, in FCC band, the packet size becomes equal to 100 bits. In order to understand the size of data as well as signal dimensions at each various points in the transmitter baseband, the calculation method may be described in the following.

[0083] Packet size calculation:

[0084] The total number of bits carried by a PHY frame may be obtained by:

$$N_F = N_G = \text{Ncarr X Ns}$$

[0085] The N_F and N_G may represent the size of packet (signal) at nodes (F) and (G), respectively. Where Ncarr is the number of carriers in each OFDM symbol and Ns is the

number of symbols per PHY frame. Note that the Interleaver does not change the size of packet. The number of bits at point (E) may be given by:

$$N_E = N_F \times R$$

[0086] The value R may be one for Normal mode and ¼ for ROBO Mode. In order to find M, the number of zeros may need to be padded at the output of convolutional encoder; first we need to calculate the maximum number of RS bytes. The maximum number of RS bytes (MaxRSbytes) at the output of RS encoder may be obtained by the following equation:

$$MaxRSbytes = floor((N_E \times CCRate - CCZeroTail)/8)$$

[0087] Where CCRate and CCZeroTail are the convolutional code rate (½) and the number of zeros to be added to the input of convolutional encoder (to terminate the states to zero state), respectively. And "8" refers to the length of each RS word that is one byte. Therefore, the value of M may be obtained by:

$$M = N_E - ((MaxRSbytes \times 8) + 6) \times 2$$

[0088] Table 6 shows the number of zeroes padded after convolutional encoder for various bands.

	ROBO (Bits)	Normal (bits)
FCC	M=12	M=4
ARIB	M=6	M=12
CELENEC A	M=12	M=4
CELENEC B	M=4	M=4
CELENEC C	M=4	M=4
CELENEC B, C	M=4	M=4

Table 6: Number of zeroes padded after convolutional encoder

[0089] The number of bits at point (D), (C) and (B) now may be calculated by:

$$N_D = N_E - M,\ N_C = N_D / 2,\ N_B = N_C - 6$$

[0090] Finally, considering the fact the number of parity bytes in RS code may be equal to 8, the packet size delivered by MAC to the physical layer may be given by:

$$N_A = (N_B/8 - 8) \times 8$$

[0091] Table 7 summarizes the input packet to the physical layer for various band and both normal and ROBO modes. It should be noted that CENELEC B and CENELEC C ROBO

may not be able to have long header format (48-bit addressing) and RS parity of 16 Bytes at the same time because of the size of the packet limitations.

	ROBO (bits)	Normal (bits)
FCC	424	1928
ARIB	392	1784
CELENEC A	304	1448
CELENEC B	168	888
CELENEC C	168	888
CELENEC B, C	288	1368

Table 7: Packet size delivered by MAC layer to PHY layer

[0092] The packet size at various nodes in the FEC encoding unit 710 for each band (CENELEC (A,B,BC)/FCC/ARIB) may be calculated and summarized in Tables 8A, 8B, 8C, 8D, 8E, and 8F. The nodes A, B, C, D, E, and F are shown in Figure 8.

FEC Node	**Normal Mode**	**ROBO Mode**
A	1928	424
B	1992	428
C	1998	494
D	3996	988
E	4000	1000
F	4000	4000

Table 8A: Packet Size at various node of FEC encoder for FCC band

FEC Node	**Normal Mode**	**ROBO Mode**
A	1784	392
B	1848	456
C	1854	462
D	3708	924
E	3720	930
F	3720	3720

Table 8B: Packet Size at various node of FEC encoder for ARIB band

FEC Node	Normal Mode	ROBO Mode
A	1448	304
B	1512	368
C	1518	374
D	3036	748
E	3040	760
F	3040	3040

Table 8C: Packet Size at various nodes of FEC encoder for CENELEC A band

FEC Node	Normal Mode	ROBO Mode
A	888	168
B	952	232
C	958	238
D	1916	476
E	1920	480
F	1920	1920

Table 8D: Packet Size at various node of FEC encoder for CENELEC B band

FEC Node	Normal Mode	ROBO Mode
A	888	168
B	952	232
C	958	238
D	1916	476
E	1920	480
F	1920	1920

Table 8E: Packet Size at various node of FEC encoder for CENELEC C band

FEC Node	Normal Mode	ROBO Mode
A	1368	288
B	1432	352
C	1438	358
D	2876	716
E	2880	720
F	2880	2880

Table 8F: Packet Size at various nodes of FEC encoder for CENELEC BC band

[0093] Figure 8 is a diagram illustrating the FEC encoding unit 710 according to one embodiment. The FEC encoding unit 710 includes a data scrambler 810, a Reed-Solomon (RS) encoder 820, a zero padding 830, a convolutional encoder 840, a zero padding 850, a ROBO encoder 860, a switch 870, an interleaver 880, and an un-buffer 890. It is noted that the FEC encoding unit 710 may include more or less than the above elements. In addition, any one of the above elements may be implemented by hardware, software, firmware, or any combination of hardware, software, and firmware.

[0094] The FEC encoders may include Reed Solomon encoder 820 followed by convolutional encoder 840. In ROBO mode, an extra encoder namely Repetition Code (RC) or ROBO encoder 860 may be used after the convolutional encoder 840 that repeats the bits at the output of convolutional encoder 840 four times

[0095] The data scrambler 810 may help give the data a random distribution. Figure 9A is a diagram illustrating the data scrambler 810 according to one embodiment. The data stream may be XOR-ed with a repeating pseudo random number (PN) sequence using the following generator polynomial: $S(x) = x^7 + x^4 + 1$. The bits in the scrambler are initialized to all ones at the start of processing each PHY frame.

[0096] The RS encoder 820 encodes data from the scrambler 810. The RS encoder 820 may be may be created by shortening RS (255,247, t = 4) and (255,239, t = 8) code. The "RS symbol word length" (i.e., the size of the data words used in the Reed-Solomon block) may be fixed at 8 bits. The value of t (number of word errors that can be corrected) may be either 4 or 8 for different standards. For CENELEC B and C ROBO the RS parity of 8 Bytes (corresponding to t=4) should be used. The number of parity words in a RS-block is thus 2t words. The number of non-parity data words (bytes) in Reed-Solomon encoder 820

may be provided in Table 3. The first bit in time from the data scrambler 810 may become the most significant bit of that symbol. Each RS encoder input block (consisting of 247 symbols) is conceptually formed by one or more fill symbols ("00000000") followed by the message symbols. Output of the RS encoder (with fill symbols discarded) may proceed in time from first message symbol to last message symbol followed by parity symbols, with each symbol shifted out most significant bit first.

Code Generator Polynomial $g(x) = (x - \alpha^1)(x - \alpha^2)(x - \alpha^3)... (x - \alpha^8)$

Field Generator Polynomial: $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ (435 octal)

	Normal Mode N_A/N_B (Bytes)	ROBO Mode N_A/N_B (Bytes)
FCC	241/249	53/61
ARIB	223/231	49/57
CENELEC A	181/189	38/46
CENELEC BC	171/179	36/44
CENELEC B	111/119	21/29
CENELEC C	111/119	21/29

Table 9: RS encoder input/output packet size

[0097] The representation of α^0 is "00000001", where the left most bit of this RS symbol is the MSB and is first in time from the scrambler 810 and is the first in time out of the RS encoder 820. The packet size (in Bytes) at the input and output of RS encoder 820 (N_A and N_B) may be given in Table 9.

[0098] The zero padding 830 may pad six zeroes after the RS encoder 820.

[0099] Figure 9B is a diagram illustrating the convolutional encoder 840 according to one embodiment. The convolutional encoder 840 may encode the bit stream at the output of the Reed-Solomon encoder 820 with a standard rate = ½, K=7. The tap connections are defined as x = 0b1111001 and y = 0b1011011, as shown in Figure 9B.

[00100] When the last bit of data to the convolutional encoder 840 may have been received, the convolutional encoder 840 may insert six tail bits, which may be required to return the convolutional encoder 840 to the "zero state". This may improve the error probability of the convolutional decoder, which relies on future bits when decoding. The tail

bits may be defined as six zeros. The number of bits at the input and the output of convolutional encoder may be given in Table 10.

	Normal Mode N_A/N_B (bits)	ROBO Mode N_A/N_B (bits)
FCC	1998/3996	494/988
ARIB	1854/3708	462/924
CENELEC A	1518/3036	374/748
CENELEC BC	1438/2876	358/716
CENELEC B	958/1916	238/476
CENELEC C	958/1916	238/476

Table 10: Convolutional encoder input/output packet sizes

[00101] The zero padding 850 may pad M zeroes after the convolutional encoder 840. M is given in Table 6.

[00102] The ROBO encoder 860 repeats the resulting packet after adding M number of zeros to the packet four times in ROBO mode. The ROBO encoder 860 may be only activated in ROBO mode. The repeat code may be implemented inside the interleaver 880. The switch 870 selects whether to bypass the ROBO encoder 860 or uses the output of the ROBO encoder 860 in ROBO mode.

[00103] The interleaver 880 interleaves the data packet selected from the switch 870. It may be used for both normal mode and ROBO mode. The interleaver 880 may use a linear block interleaver and may achieve the same performance of a random interleaver using a simpler architecture with less computation.

[00104] The un-buffer 890 breaks down the final packet into small packet so that each small packet may be fitted into one OFDM symbol, as described earlier.

[00105] Figure 10 is a diagram illustrating the modulator 720 according to one embodiment. The modulator 720 includes a DBPSK modulator 1010, a zero padding 1020, an inverse Fast Fourier Transform (IFFT) 1030, and a cyclic prefix (CP) extension 1040. It is noted that the modulator 720 may include more or less than the above elements. In

addition, any one of the above elements may be implemented by hardware, software, firmware, or any combination of hardware, software, and firmware.

[00106] Figure 11A is a diagram illustrating the DBPSK modulator 1010 according to one embodiment. The DBPSK modulator 1010 includes a mapper 1110 and a differential modulator 1120.

[00107] The mapper 1110 map data bits for differential modulation. Each phase vector may use its predecessor (same carrier, previous symbol) as phase reference. The mapping function for DBPSK may be given in Table 11.

Input Bit	**Output Phase**
0	Ψ_k
1	$\Psi_k + \pi$

Table 11: DBPSK Encoding Table of Kth Sub carrier

[00108] The initial phase for the first symbol are the carrier phases of the preamble symbol and are provided in Table 12. Each value in Table 12 may be a multiple integer of $\pi/8$ and may be quantized by 4 bits. The preamble phase reference index in Table 12 may start from 1 and it may refer to the first carrier in the corresponding band as given in Table 13. Note that the carrier index may be numbered from 0 to 127. This may be been illustrated in Figure 11B.

c	FCC ϕ_c	ARIB ϕ_c	CNLCCA ϕ_c	CNLCCB ϕ_c	CNLCCC ϕ_c	CNLCCBC ϕ_c	c	FCC ϕ_c	ARIB ϕ_c	c	FCC ϕ_c	ARIB ϕ_c	c	FCC ϕ_c	ARIB ϕ_c
1	14	14	14	X	X	X	**26**	8	4	**51**	11	12	**76**	9	8
2	14	14	13	X	X	X	**27**	3	15	**52**	3	3	**77**	12	11
3	13	13	11	X	X	X	**28**	15	10	**53**	10	10	**78**	15	13
4	13	12	9	X		X	**29**	10	5	**54**	1	1	**79**	3	0
5	12	12	6	X		X	**30**	5	0	**55**	9	7	**80**	6	2
6	11	11	2	X		X	**31**	0	11	**56**	15	14	**81**	9	4
7	10	10	12			X	**31**	11	5	**57**	7	4	**82**	12	6
8	9	8	6			X	**33**	6	0	**58**	13	10	**83**	14	7
9	8	7	14			X	**34**	0	10	**59**	4	0	**84**	0	8
10	6	6	6				**35**	11	4	**60**	10	6	**85**	3	10
11	5	4	12				**36**	5	14	**61**	0	11	**86**	5	10
12	3	2	2				**37**	15	7	**62**	6	0	**87**	7	11
13	1	0	7				**38**	9	1	**63**	12	5	**88**	9	12
14	14	13	10				**39**	3	10	**64**	1	10	**89**	10	13
15	12	11	13				**40**	12	3	**65**	7	15	**90**	11	13
16	9	8	15				**41**	6	12	**66**	12	4	**91**	12	14
17	7	5	0				**42**	15	5	**67**	1	8	**92**	13	14
18	4	2	1				**43**	8	14	**68**	7	12	**93**	14	14
19	1	15	1				**44**	1	6	**69**	11	0	**94**	15	
20	14	11					**45**	10	14	**70**	0	4	**95**	0	
21	10	8					**46**	3	7	**71**	5	8	**96**	0	
22	7	4					**47**	11	14	**72**	9	11	**97**	1	
23	4	0					**48**	3	6	**73**	13	15	**98**	1	
24	0	12					**49**	11	14	**74**	1	2	**99**	1	
25	11	8					**50**	3	5	**75**	5	5	**100**	2	

Table 12: Preamble Phase Vector Definition

	C_{n1}	C_{n2}
FCC	3	102
ARIB	3	95
CELENEC-A	2	20
CELENEC-B	21	26
CELENEC-C	27	29
CELENEC-BC	21	29

Table 13: First and Last Carrier Indexes for each band

[00109] Figure 11B is a diagram illustrating the carrier index numbers according to one embodiment.

[00110] The IFFT 1030 may take the 256-point IFFT of the input vector and may generate the main 256 time domain OFDM words preceded by N_CP words of cyclic prefix. In other words, the last N_CP samples at the output of the IFFT 1030 may be taken and placed in front of symbol. The useful output may be the real part of the IFFT coefficients. Figure 11C is a diagram illustrating the input/output configuration according to one embodiment. The first carrier C_{n1} and the last carrier index C_{n2} associated with each band may be given in Table 13.

[00111] Figure 12 is a diagram illustrating the PSD shaping module 730 according to one embodiment. The PSD shaping module 730 includes a raised cosine shaping 1210, an overlapping 1220, and a pre-emphasis filter 1230.

[00112] Figure 13A is a diagram illustrating a raised cosine function according to one embodiment. Figure 13B is a diagram illustrating a overlapping and add operation according to one embodiment.

[00113] In order to reduce the out of band emission and to reduce the spectral side lobe, a window function may be applied. In one embodiment, the Raised Cosine shaping 1210 may be applied to all the data symbols. Then the tails and heads of successive symbols may be overlapped and added together by the overlapping 1220. This process is described below. Each side of a symbol may be first shaped by the raised cosine function as shown in Figure 13A.

[00114] The windowing function at each 8-sample boundary may be a Raised Cosine function and its values are given in Table 14. The window function may have a value equal to one at other samples. Then the 8 tail and 8 head shaped samples of the symbol from each side of symbol may be overlapped with the tail and head samples of adjacent symbols as shown in Figure 13B. In other words, In order to construct the n_{th} symbol, firstly its 8 head samples may be overlapped with the 8 tail samples of the $(n-1)_{th}$ symbol and its 8 tail samples may be overlapped with the 8 head samples of the $(n+1)_{th}$ symbol. Finally, the corresponding overlapped parts may be added together. Note that the head of the first symbol is overlapped with the tail of preamble. And the tail of last symbol may be sent out with no overlapping applied.

	Head samples	**Tail samples**
1	0	0.9619
2	0.0381	0.8536
3	0.1464	0.6913
4	0.3087	0.5000
5	0.5000	0.3087
6	0.6913	0.1464
7	0.8536	0.0381
8	0.9619	0

Table 14: The Raised Cosine Samples

[00115] Figure 14 is a diagram illustrating a preamble signal according to one embodiment.

[00116] Memory locations may need to be allocated in the baseband to store the preamble samples. The preamble samples may be prepared in advance and download into the baseband memory during initialization period by the processor that implements the MAC layer. Each sample of preamble symbol may have an 8-bit length. The preamble signal that may be added to the beginning of each PHY frame may be shown in Figure 14. It may include 8 symbols of type P and 1.5 symbols of type M. The total number of samples may be equal to 2432 samples. The first and the last 8 samples may be shaped according to

Raised Cosine window. Note that the last 8 samples may be overlapped by the first 8 samples of the first data symbol. In practice we only need to store 256 sample of symbol P, 256 samples of symbol M, the first and the last 8 samples. Note that the symbol M may be different in Normal mode from that in the ROBO mode. In the ROBO mode, symbol M may be signed reversed of the P symbol, so there may be no extra storage required to store another symbol M for ROBO mode. In normal mode, the M symbol may have 90^{o} phase shift from the P symbol.

[00117] Figure 15 is a diagram illustrating the pre-emphasis filter 1230 according to one embodiment.

Time-Domain Pre-emphasis Filter:

[00118] A time-domain pre-emphasis filter 1230 may be a linear equalization method where the transmit signal spectrum may be shaped to compensate for amplitude distortion. The purpose of this filter may be to provide frequency shaping to the transmit signal in order to compensate for attenuation introduced to the signal as it goes through the power line.

[00119] The pre-emphasis filter 1230 may be a first order recursive filter with transfer function of H(z) = 0.5*[(Gamma + Beta*z^-1) / (1 - R*z^-1)]. It may be specified with below deference equation:

y(n) = 0.5*[Gamma*x(n) + Beta*x(n-1) + R*y(n-1)]

[00120] As shown, the pre-emphasis filter 1230 may have one zero and one pole. In this implementation Gamma, Beta, and R may be programmable and may be assigned 16-bit registers. The pre-emphasis filter 1230 may be the last block in the transmit path right before the output formatter 750. The pre-emphasis filter may have the following register requirements: an enable/disable bit to enable/bypass the pre-emphasis filter, a Gamma register (signed 16 bits): parameter to control the shape of the pre-emphasis filter, a Beta register (signed 16 bits): parameter to control the shape of the pre-emphasis filter, and an R register (signed 16 bits): parameter to control the shape of the pre-emphasis filter.

Frequency Domain Pre-emphasis Filter:

[00121] Figure 16 is a diagram illustrating the pre-emphasis filter according to one embodiment. The purpose of this block may be to provide frequency shaping to the transmit signal in order to compensate for attenuation introduced to the signal as it goes through the power line.

[00122] The frequency-domain pre-emphasis filter may include of a multiplier that may multiply the complex frequency domain samples of an OFDM symbol with 128 real filter coefficients, then do four right shifts at the output. The filter coefficients may be 5 bits representing unsigned values from 0h to 10h. The filter coefficients may not be allowed to have values larger than 10h. The filter may multiply the first 128 frequency-domain complex samples of an OFDM symbol with the 128 real coefficients of the filter. The rest of the 128 frequency-domain samples of the OFDM symbol may be usually set to zero and may not be multiplied by the filter coefficients. As the block diagram below shows, the input complex samples may be 8 bits each while the filter coefficients may be 5 unsigned bits each. Since the maximum allowed value of any filter coefficients may be 10h, the output of the multiplication may be 12 bits (not 13 bits). The output may then be right shifted by 4 to get a final output of 8 bits that may be used as input to the IFFT.

[00123] The filter coefficient values may vary from 0 to 16, and since we do 4 right shifts at the output, it follows that the filter may provide the following attenuation for any of the 128 carriers:

Scaling factor	attenuation in dB
16/16	0 dB
15/16	-0.53 dB
14/16	-1.16 dB
13/16	-1.8 dB
12/16	-2.5 dB
11/16	-3.25 dB
10/16	-4 dB
9/16	-5 dB
8/16	-6 dB
7/16	-7.2 dB
6/16	-8.5 dB
5/16	-10.1 dB
4/16	-12 dB
3/16	-14.5 dB
2/16	-18 dB
1/16	-24 dB
0/16	-infinite

[00124] The following registers may be needed to control the frequency-domain pre-emphasis filter: Enable/Disable bit: Allows for enabling/disabling the filter.

Transmitter (TX) P and D Scaler:

[00125] In order to control the transmitted power level of P and M in relation to power level of data two scalers may be implemented in the transmitter: a Data scaler to scale the data, and a P/M scaler to control the levels of the P & M that we are now generating from the frequency domain using the IFFT. Both scalers may be described below. On the receiver path, no scaling may be needed.

[00126] Figure 17A is a diagram illustrating a data scaler on the transmitter data path according to one embodiment. Figure 17B is a diagram illustrating a P and M scaler on the transmitter data path according to one embodiment. Figure 17C is a diagram illustrating a scaler for frequency-domain P and M signals according to one embodiment. Figures 17A and 17B are provided to show how the 4-bit sync reference may be generated and scaled. The P/M scaler is used to scale IFFT output of frequency-domain P and M so that their levels may be as close as possible to original time-domain P & M. Once that is accomplished, the data scaler is used to achieve the desired P/Data RMS ratio. In what follows, the P/M scaler is described first, followed by the data scaler, which may have an identical architecture.

[00127] Figure 17C shows how the IFFT output of frequency-domain P and M may be scaled so that their levels may be as close as possible to original time-domain P & M. This block may be called the "P/M Scaler". The table lists the values for P_scale_factor and P_shift_factor registers for the different wideband and narrowband standards.

[00128] The data scaler may have identical block to the P/M scaler except that P_scale_factor is renamed to Data_scale_factor, and P_shift_factor is renamed to Data_shift_factor, where both may remain 8 bits each. The table shows the default values for the data scalers for the different standards.

[00129] Figure 18 is a diagram illustrating a receiver 1800 according to one embodiment. The receiver 1800 includes a data formatter 1810, a direct current (DC) blocker 1815, an analog automatic gain control (AGC) processor 1820, a processing unit 1830, a digital AGC processor 1840, a demodulator 1850, a symbol synchronizer 1852, a frame synchronizer 1854, a preamble FFT coefficient buffer 1860, a mode detector 1870, and a FEC decoding unit 1880. It is noted that the receiver 1880 may include more or less

than the above elements. In addition, any one of the above elements may be implemented by hardware, software, firmware, or any combination of hardware, software, and firmware.

[00130] Figure 19 is a diagram illustrating the demodulator 1850 according to one embodiment. The demodulator 1850 includes a cyclic prefix (CP) remover 1910, a FFT processor 1920, and a DBPSK demodulator 1930.

[00131] Figure 20 is a diagram illustrating the FEC decoding unit 1880 according to one embodiment. The FEC decoding unit 1880 includes a buffer 2010, a de-interleaver 2020, a ROBO combiner 2030, a zero remover 2040, a Viterbi decoder 2050, a RS decoder 2060, and a descrambler 2070. It is noted that the FEC decoding unit 1880 may include more or less than the above elements. In addition, any one of the above elements may be implemented by hardware, software, firmware, or any combination of hardware, software, and firmware.

[00132] On the receiver side, the PHY layer may receive its input samples from the power line and may hand the demodulated data bits over to the MAC layer. The processing unit 1830 may include a first infinite impulse response (IIR) filter 1832, a second IIR filter 1834, a jammer canceller 1836, and a root mean square (RMS) module 1838. The jammer canceller 1836 removes interference or a jamming signal in the input signal. The symbol synchronizer 1852 and the frame synchronizer 1854 may be used for preamble (ACK signal) detection, symbol and frame synchronization. The frame synchronizer 1854 and the preamble FFT coefficient buffer 1860 may be used to perform the initial preamble phase and the channel estimation, respectively.

[00133] The synchronizers 1852 and 1854 and the jammer canceller 1836 may be ON when the system is in the "Receive" mode. If the jammer or interfering signal may be present in the channel and detected, a switch may be set so that the signal may be taken from the output of the jammer canceller 1836. The jammer detector in the jammer canceller 1836 may do this automatically. The mode detector 1870 detects the mode of operation and sets an ACK flag 1872 or a ROBO flag 1874 as appropriate.

[00134] Two different synchronizer circuits, one for FCC, ARIB and CENELEC bands (Wideband OFDM) and another for CENELEC B, C and BC (Narrow band OFDM), may be used for different bands. The tasks for synchronizers may be the detection of preamble and obtaining the start of preamble symbol (symbol synchronizer) and the start of data symbol (frame synchronizer). As soon as the start of data symbol may be found, a switch may be moved to place the CP remover in the demodulator 1850 (Figure 19) in the

signal path. At the same time a timer 1865 may be enabled to generate the Physical Carrier Sense (PCS) signal. This signal may be high for the entire frame period. It may be at the end of PCS signal that the ACK flag 1872 and the ROBO flag 1874 are reset. Note that same waveforms may be used for ACK signaling and therefore as soon as the preamble is detected the ACK flag 1872 may be set. The value of this flag may be read by the MAC software and may be reset at the end of PCS signal. Note that the frame synchronizer 1854 may also detect if the PHY frame may be in ROBO mode or in Normal mode accordingly set/reset the ROBO flag 1874.

[00135] Once the symbol synchronizer identifies the start of preamble symbols, the initial channel estimator may be activated. At this time a switch may be set since there may be no cyclic prefix extension for preamble symbols. This block may measure the reference phase from the preamble. It may also measure the channel quality at each frequency bin. The channel estimator may also estimate the SNR for each carrier.

[00136] The ROBO flag 1874 may select the position of a switch in the FEC decoding unit 1880. Depending on the preamble waveform, the frame synchronizer 1854 may identify if the frame is in ROBO mode or in Normal Mode and the switch in the FEC decoding unit 1880 is set accordingly.

[00137] Figure 21 is a diagram illustrating timings associated with events in the receiver according to one embodiment.

[00138] The data formatter 1810 may take the data bits from the analog-to-digital converter (ADC) and may perform tasks including, scaling, and mapping to convenient signed value representation. The DC blocker 1815 may be used to remove the DC component of incoming data. Since A/D converters and analog front-end circuitry may not be expected to be totally DC free, this filter may remove the DC residual. Figure 22 is a diagram illustrating the DC blocker 1815 according to one embodiment. The DC blocker 1815 may be a first order recursive filter with transfer function of $H(z) = 1\text{-}z^{-1}/1\text{-}Az^{-1}$. It may be specified with the deference equation $y(n) = x(n) - x(n\text{-}1) + A\ y(n\text{-}1)$. DC blocker may have a zero at DC (z=1) and a pole near DC at z=A. In order to have the pole and zero cancel each other A may be selected as close as possible to unit circle. In one embodiment, $A=.995*2^{15}=32604$. The DC blocker 1815 may be the first block in receiver path before jammer canceller 1836. An enable/disable register may be allocated for the DC blocker 1815.

[00139] Figure 23 is a diagram illustrating the FFT 1920 according to one embodiment. The same structure as used for the IFFT in the transmitter is used for FFT as well.

[00140] Figure 24 is a diagram illustrating the DBPSK demodulator 1930 according to one embodiment. The phase difference between carriers over successive symbols may be estimated after the FFT of the current symbol may be multiplied by the conjugate values of the FFT of the previous symbol. The size of the signal at each node in Figure 4.15 may be equal to the number of carriers (Ncarr). The real value of the signal at the output of multiplier may be taken and quantized appropriately by soft detection block. Each bit (carried by each carrier) may be represented by an integer number. The value of this number may depend on the reliability of the bit. The length of this integer number may be provided in fixed-point implementation.

[00141] The bit de-interleaver 2020 may reverse the mappings described in the transmitter section.

[00142] Figure 25 is a diagram illustrating the ROBO combiner/decoder 2030 according to one embodiment. In ROBO mode, the encoded data may be extended 4 times by parity bits. For the repeat code, the soft values for each demodulated carrier are obtained. Then all the four values associated with one data bit may be averaged prior to hard decoding.

[00143] The errors at the output of Viterbi decoder tend to occur in a burst fashion. To correct these burst errors a RS code may be concatenated with convolutional code.

[00144] Figure 26 is a diagram illustrating the RS decoder 2060 according to one embodiment.

[00145] The de-scrambler 2070 may reverse the scrambling action, done on the source information bits in the transmitter.

[00146] The tasks of synchronizer are to identify the beginnings of preamble symbols as well as the beginning of data symbols. The symbol synchronizer circuit may detect the preamble and the beginning of the preamble symbols while the frame synchronizer circuit is used to detect the beginning of the data symbol. The frame synchronizer may also be used to identify the mode of operation of the current PHY frame (Normal or ROBO). Two different synchronization techniques are used for symbol synchronizer. One technique is used for wideband OFDM (ARIB, FCC and CENELEC A) and another circuit for narrowband OFDM (CENELEC B and C).

Symbol synchronizer:

[00147] The symbol synchronizer circuit may search for the preamble symbol P and by detecting it, the beginning of the symbol P may be identified. Once the symbol synchronization may be achieved, the frame synchronizer circuit may become active to detect the symbol M in the preamble. Once the preamble symbol M may be detected, the position of current sample relative to the beginning of the data symbol may be known. The general scheme for symbol and frame synchronization is as follows. During the search for symbol M, the FFT of the preamble symbol P may be taken. These FFT results may be averaged over the number of times that the symbol P is detected and used by the demodulator to detect the first data symbol. Note that in practice, the phase may not be calculated. Instead the phase difference may be calculated by multiplying the FFT of the current symbol by the conjugate of the previous symbol.

[00148] Figure 27A is a diagram illustrating the signal waveforms according to one embodiment. The waveform (A) corresponds to the input signal to the symbol synchronizer and the frame synchronizer. It may correspond to the output of the jammer caller 1836 or the digital AGC processor 1840. The waveform (B) may correspond to the output of the symbol synchronizer 1852. The waveform (C) may correspond to the output of the frame synchronizer 1854. The preamble may include 8 identical symbols P and 1½ symbols M. The symbol synchronizer may point at the start of one of the preamble symbols P (B). The transition of the signal (C) may indicate that the end of preamble symbol M is found from which the start of data symbol is then identified. Also the frame synchronizer may set one bit flag (ROBO Flag) indicating that the current PHY frame is transmitted in ROBO mode. A moving average filter at the beginning of the synchronizer may be used to enhance the Signal to Noise Ratio (SNR).

[00149] Figure 27B is a diagram illustrating the symbol synchronizer 1852 according to one embodiment. The symbol synchronizer 1852 includes a moving average filter 2710, a sign element 2720, a correlation estimator 2730, and a synchronization detector 2740. It is noted that the symbol synchronizer 1852 may include more or less than the above elements. In addition, any one of the above elements may be implemented by hardware, software, firmware, or any combination of hardware, software, and firmware.

[00150] The moving average filter 2710 may be coupled to the processing circuit 1830 or the digital AGC processor 1840 to generate the samples in the sample vector by averaging L input symbols that are separated from each other by M input samples, L being

less than M. In one embodiment, L = 3 and M = 256. It may be placed right after the jammer canceller 1836. During the transmission of symbol P, the samples may be repeated every M samples. Therefore, these corresponding samples may be averaged in a sliding fashion to improve the SNR. Increasing the length of moving average filter may increase the enhancement of the SNR, however, on the other hand, the size of memory required to implement the circuit increases. An averaging scheme over L = 3 symbols may be selected as a suitable compromise between the performance and the complexity.

[00151] The sign element 2720 obtains signs of samples in a sample vector. The sign may be positive or negative. Each sample is therefore represented by one bit. The correlation estimator 2730 computes a correlation of the sample vector. The synchronization detector 2740 detects symbol synchronization.

[00152] Figure 28 is a diagram illustrating the moving average filter 2710 according to one embodiment. The moving average filter 2710 may include two adders 2810 and 2820 and two delay lines 2830 and 2840 each with M-sample memory. The delay lines 2830 and 2840 may be implemented by shift registers or first-in-first-out (FIFO) memories. At every sample time the output of the adders may be calculated and then the input may be shifted to the right inside the shift registers. The number of bits for each node may be shown in Figure 28 for illustrative purposes.

[00153] Figure 29 is a diagram illustrating the preamble waveform and the autocorrelation function according to one embodiment. In FCC, ARIB and CENELEC A bands, each symbol of the preamble may be similar to a chirp signal with an impulse like autocorrelation. This special property of the preamble symbol provides a suitable means for timing recovery. The correlation between a replica of the transmitted symbol (reference signal) and the received symbols is used to detect the beginning of the symbol. Figure 29 shows a typical waveform for preamble and its autocorrelation function. This waveform may be transmitted several times as a preamble symbol prior to the transmission of the data symbol. In the receiver the cross correlation between the received and the replica of the transmitted signal is performed. It is the objective of the symbol synchronizer to detect the peak which itself occurs at the boundaries of the preamble symbols.

[00154] Figure 30 is a diagram illustrating the correlation estimator 2730 according to one embodiment. The correlation estimator 2730 includes a serial-to-parallel converter 3010, R multipliers 3020_1 to 3020_R, and an adder 3030.

[00155] The serial-to-parallel converter 3010 serially shifts the signs of the samples and produces a sign vector with a vector length of R representative of the sample vector. In one embodiment, R = 256. The R multipliers 3020_1 to 3020_R multiply the sign vector with a corresponding set of R reference samples to generate R products;. The R reference samples may include a portion of the transmitted preamble symbol and may be quantized to 3 or 4 bits. The adder 3030 adds the R products to produce the correlation. In one embodiment, the correlation may be 11 bits.

[00156] Figure 31 is a diagram illustrating the synchronization detector 2740 according to one embodiment. The synchronization detector 2740 includes an absolute processor 3110, a filter 3120, a buffer 3130, and a peak detector 3140.

[00157] The absolute processor 3110 computes an absolute correlation from a scaled correlation. The scaled correlation is the correlation shifted by a pre-determined number of bits. In one embodiment, the correlation is 11 bits and the scaled correlation is 7 bits. The scaled correlation is generated from the correlation by truncating the least significant 4 bits, or shifting the correlation 4 bits to the right.

[00158] The filter 3120 filters the absolute correlation to produce filtered samples. In one embodiment, the filter is a finite impulse response (FIR) filter having K identical taps. In one embodiment, K = 7. The filter is very effective in enhancing the peaks at the symbol P boundaries particularly in low SNR and when the signal is distorted.

[00159] The buffer 3130 stores the filtered samples in a buffer having a programmable size. The buffer size corresponds to a gating window which may be programmable between 8 and 64.

[00160] The peak detector 3140 detects first and second peaks using a symbol synchronization threshold in a programmable gating window. The detected second peak represents the symbol synchronization. The detection of the peaks is performed by comparing the samples from the buffer 3130 with the symbol synchronization threshold. This value may be programmable and may be a 16-bit positive number. The symbol synchronization threshold may be adaptive based on a root mean square (RMS) value of a received signal associated with the broadband signal. The gating mechanism may be the final stage and may be used to reduce the probability of the false alarm and to increase the accuracy. Once the first symbol associated with the first peak is detected, the flags called Peak1_Flag and PHY_Carrier_Sense are set and the gating mechanism becomes active. The gating mechanism may provide a means to look for the next peak in a short period of

time when it is expected to have the second peak. The duration of the gate searching for the second peak is programmable and may be equal to the gating window. In one embodiment, 8 < gating window < 64. If during the second search, the second peak, i.e., the sample that exceeds the symbol synchronization threshold, is found, then the Peak2_Flag may be set indicating that the symbol synchronization is completed, otherwise, Peak1_Flag, Peak2_Flag and PHY_Carrier_Sense flags may be reset and the symbol synchronization may restart.

[00161] Figure 32 is a diagram illustrating the various waveforms in the symbol synchronizer according to one embodiment. The waveform (a) represents the input samples. The waveform (b) represents the correlation which is the output of the correlation estimator 2730. The waveform (c) represents the output of the filter 3120. The waveform (d) represents the output of the peak detector 3140. The waveform (e) represents the gating mechanism.

Frame Synchronizer:

[00162] For both narrowband and wideband OFDM cases, the same circuit may be used for frame synchronization. This frame synchronizer may be used for the identification of the beginning of the data symbols. The phase differences between the corresponding carriers in the symbol P and symbol M may be used for frame synchronization. Frame synchronization circuit may start its operation after preamble is detected (symbol synchronization is achieved). The frame synchronizer may successively compare the incoming received symbols in frequency domain to locate the start of the data frame. This is done by detecting the symbol M.

[00163] In general, the frame synchronizer operates as follows. N samples, which correspond to one preamble symbol, are collected from the start of symbol P, i.e., starting from the time that is pointed to by the symbol synchronizer, and at the same time the number of symbols is counted. This means that when N samples may be collected, the symbol counter may be incremented by one unit (starting from 0). The FFT of each symbol may be taken and the result may be accumulated and stored. Once the symbol counter is greater than 2 then the current FFT result is multiplied by the conjugate of previously accumulated FFTs (at this time three symbols are already collected, FFT taken and accumulated). The result of multiplications is $f_R(k)+j * f_I(k)$ (where k is the carrier index) and is used to calculate the following two parameters:

$R = \sum f_R(k)$ (over all k, the valid carriers)

$I = \sum f_I(k)$ (over all k, the valid carriers)

[00164] The fact that the phase differences between carriers in symbols M and P in Normal, ROBO and ACK Modes are $\pi/2$, π and $3\pi/2$ respectively, may enable us to identify if the PHY frame corresponds to ROBO, Normal or ACK frame from the values of R and I.

[00165] Figure 33 is a diagram illustrating the frame synchronizer 1854 according to one embodiment. The frame synchronizer 1854 includes a Fast Fourier Transform (FFT) processing unit 3310, a real and imaginary processing unit 3320, and a mode processor 3330. It is noted that the frame synchronizer 1854 may include more or less than the above elements. In addition, any one of the above elements may be implemented by hardware, software, firmware, or any combination of hardware, software, and firmware.

[00166] The FFT processing unit 3310 computes a current FFT vector and an accumulated previous FFT vector from the input samples such as from the moving average filter 2710. The current FFT vector and the accumulated previous FFT vector correspond to sample vectors associated with preamble symbols prior to symbol synchronization detection. The real and imaginary processing unit 3320 generates real and imaginary summations using the current FFT vector and the accumulated previous FFT vector. The mode processor 3330 generates mode flags representing operational modes using the real and imaginary summations. The mode processor 3330 communicates with the mode selector 1870. The mode selector 1870 receives inputs from a MAC layer to select a mode being one of a forced mode and an automatic detection mode. The automatic detection mode is achieved by using the mode as detected by the mode processor 3330. The mode selector 1870 or the mode processor 3340 may generate a frame sync flag that can be used to control the FFT processing unit 3310.

[00167] Figure 34 is a diagram illustrating the FFT processing unit 3310 according to one embodiment. The FFT processing unit 3310 includes a buffer 3410, a FFT processor 3420, and a FFT accumulator 3430.

[00168] The buffer 3410 has a programmable size to store the sample vectors as provided by the moving average filter 2710 or other input samples. The FFT processor 3420 computes the previous FFT vectors and the current FFT vector. The FFT accumulator 3430 accumulates previous FFT vectors to generate an accumulated previous FFT vector when a frame sync flag is negated. The FFT accumulator 3430 may include an adder 3432 and a

FFT storage 3434. The accumulated FFT vectors may be provided to the preamble FFT coefficient buffer 1860 to compute the average of the FFT by dividing the accumulated FFT vectors by the number of symbols used for the averaging. This may be given to the demodulator 1850 to detect the first symbol.

[00169] Figure 35 is a diagram illustrating the real and imaginary processing unit 3320 according to one embodiment. The real and imaginary processing unit 3320 includes a conjugate processor 3510, a multiplier 3520, a real summer 3530, and an imaginary summer 3540.

[00170] The conjugate processor 3510 computes a conjugate vector of the accumulated previous FFT vector as provided by the FFT accumulator 3430. The multiplier 3520 multiplies the current FFT vector as provided by the FFT processor 3420 and the conjugate vector to generate a product vector having a real part and an imaginary part as discussed above.

[00171] The real summer 3530 sums components of the real part R over the carrier indices to generate the real summation R. The imaginary summer 3540 sums components of the imaginary part over the carrier indices to generate the imaginary summation I. The real summation R and the imaginary summation I provide information regarding the phase differences to determine the various operational modes as discussed above.

[00172] Figure 36 is a diagram illustrating the constellation associated with symbols P and M in various operational modes according to one embodiment. Figure 36 shows the constellation associated with the output samples for the cases when both symbols (the current one and the previous one) are identical (both are symbols P) and when the current symbol is the symbol M (for ROBO, Normal and ACK Modes) and the previous symbol is P.

[00173] It is seen in order to detect Symbol M in Normal mode; it may be sufficient to check if the following conditions are satisfied:$\{abs(I) > abs(R)\}$ and $\{ I > 0\}$ where abs( ) is the absolute value. For the case to detect the symbol M in ROBO Mode, the conditions may be: $\{abs(R) > abs(I)\}$ and $\{R < 0\}$. For the case to detect the symbol M in ACK Mode, the conditions may be: $\{abs(I) > abs(R)\}$and $\{I < 0\}$. If the above conditions are not satisfied, it means that the current symbol may be P or equivalently: $\{abs(R) > abs(I)\}$ and $\{R > 0\}$.

[00174] Figure 37 is a diagram illustrating the output when P is followed by noise according to one embodiment. The above decision criteria may be very loose. Particularly,

if P is followed by noise, the correlator output may be anywhere in the complex plane as illustrated in Figure 37. Even though, the system may always transmit a kind of M after P, it is still helpful to add the thresholds to the decision. For instance, if a wrong P detection happens, frame detection may detect another P, normal M, ROBO M or ACK M.

[00175] Knowing the fact that when correlation exists, a large pure real or imaginary value may be expected, using two threshold level values may tighten the decision logic. The new decision criteria incorporating the threshold values may be:

- Normal M:{abs(I) > abs(R)} and { I > 0} and { abs(I) > HT} and {abs(R) < LT, where HT and LT are high threshold and low threshold, respectively.
- ROBO M: {abs(R) > abs(I)} and {R < 0} and { abs(R) > HT} and {abs(I) < LT}
- ACK M: {abs(I) > abs(R)} and {I < 0} and { abs(I) > HT} and {abs(R) < LT}
- P: {abs(R) > abs(I)} and {R > 0} and {abs(R) > HT} and {abs(I) < LT}

[00176] These thresholds may be masked by setting HT = 0 and LT = Maximum Value.

[00177] Figure 38 is a diagram illustrating the threshold effect on detection criteria according to one embodiment. It can be seen that these thresholds can tighten the decision boundary.

[00178] Figure 39 is a diagram illustrating the mode processor 3330 according to one embodiment. The mode processor 3330 includes an absolute processor 3910, a comparator unit 3920, and a mode logic module 3930.

[00179] The absolute processor 3910 computes real and imaginary absolute values of the real and imaginary summations. The comparator unit 3920 generates comparison results using the real and imaginary absolute values and the real and imaginary summations. The mode logic module 3930 generates the mode flags using the comparison results. The mode flags include a frame normal flag, a frame ROBO flag, a frame ACK flag, and a preamble symbol flag.

[00180] The mode logic module 4630 may include a frame normal detector 3932, a frame ROBO detector 3934, a frame ACK detector 3936, and a preamble symbol detector 3938. When in the basic mode detection, the frame normal detector 3932 detects a normal mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is greater than zero. The detected normal mode asserts the frame normal flag and negates the frame ROBO flag, the frame ACK flag, and the phase symbol flag. The frame ROBO detector 3934 detects a ROBO mode when the real absolute value is

greater than the imaginary absolute value and the real summation is less than zero. The detected ROBO mode asserts the frame ROBO flag and negates the frame normal flag, the frame ACK flag, and the preamble symbol flag. The frame ACK detector 3936 detects an ACK mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is less than zero. The detected ACK mode asserts the frame ACK flag and negates the frame normal flag, the frame ROBO flag, and the preamble symbol flag. The preamble symbol detector 3938 detects a phase symbol mode when the real absolute value is greater than the imaginary absolute value and the real summation is greater than zero. The detected symbol mode asserts the preamble symbol flag and negates the frame normal flag, the frame ROBO flag, and the frame ACK flag.

[00181] When in two-threshold mode, the frame normal detector 3932 detects a normal mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is greater than zero and the imaginary absolute value is greater than a high threshold and the real absolute value is less than a low threshold. The detected normal mode asserts the frame normal flag and negates the frame ROBO flag, the frame ACK flag, and the preamble symbol flag. The frame ROBO detector 3934 detects a ROBO mode when the real absolute value is greater than the imaginary absolute value and the real summation is less than zero and the real absolute value is greater than a high threshold and the imaginary absolute value is less than a low threshold. The detected ROBO mode asserts the frame ROBO flag and negates the frame normal flag, the frame ACK flag, and the preamble symbol flag. The frame ACK detector 3936 detects an ACK mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is less than zero and the imaginary absolute value is greater than a high threshold and the real absolute value is less than a low threshold. The detected ACK mode asserts the frame ACK flag and negates the frame normal flag, the frame ROBO flag, and the preamble symbol flag. The preamble symbol detector 3938 detects a preamble symbol mode when the real absolute value is greater than the imaginary absolute value and the real summation is greater than zero and the real absolute value is greater than a high threshold and the imaginary absolute value is less than a low threshold. The detected symbol mode asserts the preamble symbol flag and negates the frame normal flag, the frame ROBO flag, and the frame ACK flag.

[00182] Elements of one embodiment may be implemented by hardware, firmware, software or any combination thereof. The term hardware generally refers to an element

having a physical structure such as electronic, electromagnetic, optical, electro-optical, mechanical, electro-mechanical parts, etc. A hardware implementation may include analog or digital circuits, devices, processors, applications specific integrated circuits (ASICs), programmable logic devices (PLDs), field programmable gate arrays (FPGAs), or any electronic devices. The term software generally refers to a logical structure, a method, a procedure, a program, a routine, a process, an algorithm, a formula, a function, an expression, etc. The term firmware generally refers to a logical structure, a method, a procedure, a program, a routine, a process, an algorithm, a formula, a function, an expression, etc., that is implemented or embodied in a hardware structure (e.g., flash memory, ROM, EPROM). Examples of firmware may include microcode, writable control store, micro-programmed structure. When implemented in software or firmware, the elements of an embodiment are essentially the code segments to perform the necessary tasks. The software/firmware may include the actual code to carry out the operations described in one embodiment, or code that emulates or simulates the operations.

[00183] The program or code segments can be stored in a processor or machine accessible medium. The "processor readable or accessible medium" or "machine readable or accessible medium" may include any medium that may store, transmit, receive, or transfer information. Examples of the processor readable or machine accessible medium that may store include a storage medium, an electronic circuit, a semiconductor memory device, a read only memory (ROM), a flash memory, an erasable programmable ROM (EPROM), a floppy diskette, a compact disk (CD) ROM, an optical disk, a hard disk, etc. The machine accessible medium may be embodied in an article of manufacture. The machine accessible medium may include information or data that, when accessed by a machine, cause the machine to perform the operations or actions described above. The machine accessible medium may also include program code, instruction or instructions embedded therein. The program code may include machine readable code, instruction or instructions to perform the operations or actions described above. The term "information" or "data" here refers to any type of information that is encoded for machine-readable purposes. Therefore, it may include program, code, data, file, etc.

[00184] All or part of an embodiment may be implemented by various means depending on applications according to particular features, functions. These means may include hardware, software, or firmware, or any combination thereof. A hardware, software, or firmware element may have several modules coupled to one another. A

hardware module is coupled to another module by mechanical, electrical, optical, electromagnetic or any physical connections. A software module is coupled to another module by a function, procedure, method, subprogram, or subroutine call, a jump, a link, a parameter, variable, and argument passing, a function return, etc. A software module is coupled to another module to receive variables, parameters, arguments, pointers, etc. and/or to generate or pass results, updated variables, pointers, etc. A firmware module is coupled to another module by any combination of hardware and software coupling methods above. A hardware, software, or firmware module may be coupled to any one of another hardware, software, or firmware module. A module may also be a software driver or interface to interact with the operating system running on the platform. A module may also be a hardware driver to configure, set up, initialize, send and receive data to and from a hardware device. An apparatus may include any combination of hardware, software, and firmware modules.

[00185] It will be appreciated that various of the above-disclosed and other features and functions, or alternatives thereof, may be desirably combined into many other different systems or applications. Various presently unforeseen or unanticipated alternatives, modifications, variations, or improvements therein may be subsequently made by those skilled in the art which are also intended to be encompassed by the following claims.

CLAIMS

What is claimed is:

1. An apparatus comprising:

a sign element to obtain signs of samples in a sample vector;

a correlation estimator coupled to the sign element to compute a correlation of the sample vector, and

a synchronization detector coupled to the correlation estimator to detect symbol synchronization.

2. The apparatus of claim 1 wherein the correlation estimator comprises:

a serial-to-parallel converter to serially shift the signs of the samples and produce a sign vector with a vector length of R representative of the sample vector;

R multipliers coupled to the serial-to-parallel converter to multiply the sign vector with a corresponding set of R reference samples to generate R products; and

an adder coupled to the plurality of the multipliers to add the R products to produce the correlation.

3. The apparatus of claim 2 wherein the synchronization detector comprises:

an absolute processor coupled to the adder to compute an absolute correlation from a scaled correlation, the scaled correlation being the correlation shifted by a pre-determined number of bits;

a filter coupled to the absolute processor to filter the absolute correlation to produce filtered samples;

a buffer coupled to the filter to store the filtered samples in a buffer having a programmable size; and

a peak detector coupled to the buffer to detect first and second peaks using a symbol synchronization threshold in a programmable gating window, the detected second peak representing the symbol synchronization.

4. The apparatus of claim 3 wherein the filter comprises:

a finite impulse response (FIR) filter having K taps.

5. The apparatus of claim 1 further comprising:

a moving average filter coupled to a processing circuit to generate the samples in the sample vector by averaging L input symbols that are separated from each other by M input samples, L being less than M.

6. The apparatus of claim 2 wherein R = 256.

7. The apparatus of claim 5 wherein L = 3 and M = 256.

8. An apparatus comprising:

a Fast Fourier Transform (FFT) processing unit to compute a current FFT vector and an accumulated previous FFT vector, the current FFT vector and the accumulated previous FFT vector corresponding to sample vectors associated with preamble symbols prior to symbol synchronization detection;

a real and imaginary processing unit coupled to the FFT processing unit to generate real and imaginary summations using the current FFT vector and the accumulated previous FFT vector; and

a mode processor coupled to the real and imaginary processing unit and a mode selection unit to generate mode flags representing operational modes using the real and imaginary summations.

9. The apparatus of claim 8 wherein the FFT processing unit comprises:

a buffer having a programmable size to store the sample vectors;

a FFT processor coupled to the buffer to compute previous FFT vectors and the current FFT vector; and

an FFT accumulator coupled to the FFT processor to accumulate the previous FFT vectors to generate an accumulated previous FFT vector when a frame sync flag is negated.

10. The apparatus of claim 9 wherein the real and imaginary processing unit comprises:

a conjugate processor coupled to the FFT accumulator to compute a conjugate vector of the accumulated previous FFT vector;

a multiplier coupled to the conjugate processor and the FFT processor to multiply the current FFT vector and the conjugate vector to generate a product vector having a real part and an imaginary part;

a real summer coupled to the multiplier to sum components of the real part over the carrier indices to generate the real summation; and

an imaginary summer coupled to the multiplier to sum components of the imaginary part over the carrier indices to generate the imaginary summation.

11. The apparatus of claim 8 wherein the mode processor comprises:

an absolute processor to compute real and imaginary absolute values of the real and imaginary summations;

a comparator unit coupled to the real and imaginary processing unit and the absolute processor to generate comparison results using the real and imaginary absolute values and the real and imaginary summations; and

a mode logic module to generate the mode flags using the comparison results, the mode flags including a frame normal flag, a frame ROBO flag, a frame ACK flag, and a preamble symbol flag.

12. The apparatus of claim 11 wherein the mode logic module comprises:

a frame normal detector to detect a normal mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is greater than zero, the detected normal mode asserting the frame normal flag and negating the frame ROBO flag, the frame ACK flag, and the preamble symbol flag;

a frame ROBO detector to detect a ROBO mode when the real absolute value is greater than the imaginary absolute value and the real summation is less than zero, the detected ROBO mode asserting the frame ROBO flag and negating the frame normal flag, the frame ACK flag, and the preamble symbol flag;

a frame ACK detector to detect an ACK mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is less than zero, the detected ACK mode asserting the frame ACK flag and negating the frame normal flag, the frame ROBO flag, and the preamble symbol flag; and

a preamble symbol detector to detect a preamble symbol mode when the real absolute value is greater than the imaginary absolute value and the real summation is greater than zero, the detected symbol mode asserting the preamble symbol flag and negating the frame normal flag, the frame ROBO flag, and the frame ACK flag.

13. The apparatus of claim 11 wherein the mode logic module comprises:

a normal frame detector to detect a normal mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is greater than zero and the imaginary absolute value is greater than a high threshold and the real absolute value is less than a low threshold, the detected normal mode asserting the frame normal flag and negating the frame ROBO flag, the frame ACK flag, and the preamble symbol flag;

a ROBO frame detector to detect a ROBO mode when the real absolute value is greater than the imaginary absolute value and the real summation is less than zero and the real absolute value is greater than a high threshold and the imaginary absolute value is less than a low threshold, the detected ROBO mode asserting the frame ROBO flag and negating the frame normal flag, the frame ACK flag, and the preamble symbol flag;

an ACK frame detector to detect an ACK mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is less than zero and the imaginary absolute value is greater than a high threshold and the real absolute value is less than a low threshold, the detected ACK mode asserting the frame ACK flag and negating the frame normal flag, the frame ROBO flag, and the preamble symbol flag; and

a preamble symbol detector to detect a symbol mode when the real absolute value is greater than the imaginary absolute value and the real summation is greater than zero and the real absolute value is greater than a high threshold and the imaginary absolute value is less than a low threshold, the detected symbol mode asserting the preamble symbol flag and negating the frame normal flag, the frame ROBO flag, and the frame ACK flag.

14. The apparatus of claim 8 wherein the mode selection unit receive inputs from a MAC layer to select a mode being one of a forced mode and an automatic detection mode.

15. A method comprising:

obtaining signs of samples in a sample vector;

computing a correlation of the sample vector, and

detecting symbol synchronization.

16. The method of claim 15 wherein computing the correlation comprises:

serially shifting the signs of the samples and producing a sign vector with a vector length of R representative of the sample vector;

multiplying the sign vector with a corresponding set of R reference samples to generate R products; and

adding the R products to produce the correlation.

17. The method of claim 16 wherein detecting the symbol synchronization comprises:

computing an absolute correlation from a scaled correlation, the scaled correlation being the correlation shifted by a pre-determined number of bits;

filtering the absolute correlation to produce filtered samples;

storing the filtered samples in a buffer having a programmable size; and

detecting first and second peaks using a symbol synchronization threshold in a programmable gating window, the detected second peak representing the symbol synchronization.

18. The method of claim 17 wherein filtering comprises:

filtering using a finite impulse response (FIR) filter having K taps.

19. The method of claim 15 further comprising:

generating the samples in the sample vector by averaging L input symbols that are separated from each other by M input samples, L being less than M.

20. The method of claim 16 wherein N = 256.

21. The method of claim 19 wherein L = 3 and M = 256.

22. A method comprising:

computing a current FFT vector and an accumulated previous FFT vector, the current FFT vector and the accumulated previous FFT vector corresponding to sample vectors associated with preamble symbols prior to symbol synchronization detection;

generating real and imaginary summations using the current FFT vector and the accumulated previous FFT vector; and

generating mode flags representing operational modes using the real and imaginary summations.

23. The method of claim 22 wherein computing the current FFT vector and the accumulated previous FFT vector comprises:

storing the sample vectors;

computing previous FFT vectors and the current FFT vector; and

accumulating the previous FFT vectors to generate an accumulated previous FFT vector when a frame sync flag is negated.

24. The method of claim 23 wherein generating the real and imaginary summations comprises:

computing a conjugate vector of the accumulated previous FFT vector;

multiplying the current FFT vector and the conjugate vector to generate a product vector having a real part and an imaginary part;

summing components of the real part over the carrier indices to generate the real summation; and

summing components of the imaginary part over the carrier indices to generate the imaginary summation.

25. The method of claim 22 wherein generating the mode flags comprises:

computing real and imaginary absolute values of the real and imaginary summations;

generating comparison results using the real and imaginary absolute values and the real and imaginary summations; and

generating the mode flags using the comparison results, the mode flags including a frame normal flag, a frame ROBO flag, a frame ACK flag, and a preamble symbol flag.

26. The method of claim 25 wherein generating the mode flags using the comparison results comprises:

detecting a normal mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is greater than zero, the detected normal mode asserting the frame normal flag and negating the frame ROBO flag, the frame ACK flag, and the preamble symbol flag;

detecting a ROBO mode when the real absolute value is greater than the imaginary absolute value and the real summation is less than zero, the detected ROBO mode asserting the frame ROBO flag and negating the frame normal flag, the frame ACK flag, and the preamble symbol flag;

detecting an ACK mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is less than zero, the detected ACK mode asserting the frame ACK flag and negating the frame normal flag, the frame ROBO flag, and the preamble symbol flag; and

detecting a preamble symbol mode when the real absolute value is greater than the imaginary absolute value and the real summation is greater than zero, the detected symbol mode asserting the preamble symbol flag and negating the frame normal flag, the frame ROBO flag, and the frame ACK flag.

27. The method of claim 25 wherein generating the mode flags using the comparison results comprises:

detecting a normal mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is greater than zero and the imaginary absolute value is greater than a high threshold and the real absolute value is less than a low threshold, the detected normal mode asserting the frame normal flag and negating the frame ROBO flag, the frame ACK flag, and the preamble symbol flag;

detecting a ROBO mode when the real absolute value is greater than the imaginary absolute value and the real summation is less than zero and the real absolute value is greater than a high threshold and the imaginary absolute value is less than a low threshold, the detected ROBO mode asserting the frame ROBO flag and negating the frame normal flag, the frame ACK flag, and the preamble symbol flag;

detecting an ACK mode when the imaginary absolute value is greater than the real absolute value and the imaginary summation is less than zero and the imaginary absolute value is greater than a high threshold and the real absolute value is less than a low threshold, the detected ACK mode asserting the frame ACK flag and negating the frame normal flag, the frame ROBO flag, and the preamble symbol flag; and

detecting a symbol mode when the real absolute value is greater than the imaginary absolute value and the real summation is greater than zero and the real absolute value is greater than a high threshold and the imaginary absolute value is less than a low threshold, the detected symbol mode asserting the preamble symbol flag and negating the frame normal flag, the frame ROBO flag, and the frame ACK flag.

28. The method of claim 22 further comprising receiving inputs from a MAC layer to select a mode being one of a forced mode and an automatic detection mode.

29. A system comprising:

a symbol synchronizer to detect symbol synchronization, the symbol synchronizer comprising:

a sign element to obtain signs of samples in a sample vector,

a correlation estimator coupled to the sign element to compute a correlation of the sample vector, and

a synchronization detector coupled to the correlation estimator to detect the symbol synchronization;

a frame synchronizer coupled to the symbol synchronizer to detect a frame synchronization, the frame synchronizer comprising:

a Fast Fourier Transform (FFT) processing unit to compute a current FFT vector and an accumulated previous FFT vector, the current FFT vector and the accumulated previous FFT vector corresponding to sample vectors associated with preamble symbols prior to symbol synchronization detection,

a real and imaginary processing unit coupled to the FFT processing unit to generate real and imaginary summations using the current FFT vector and the accumulated previous FFT vector, and

a mode processor coupled to the real and imaginary processing unit and a mode selection unit to generate mode flags representing operational modes using the real and imaginary summations;

a preamble FFT coefficient buffer coupled to the frame synchronizer to compute FFT average by dividing the accumulated FFT vectors by the number of symbols; and

a demodulator coupled to preamble FFT coefficient buffer to detect data symbols using the FFT average.

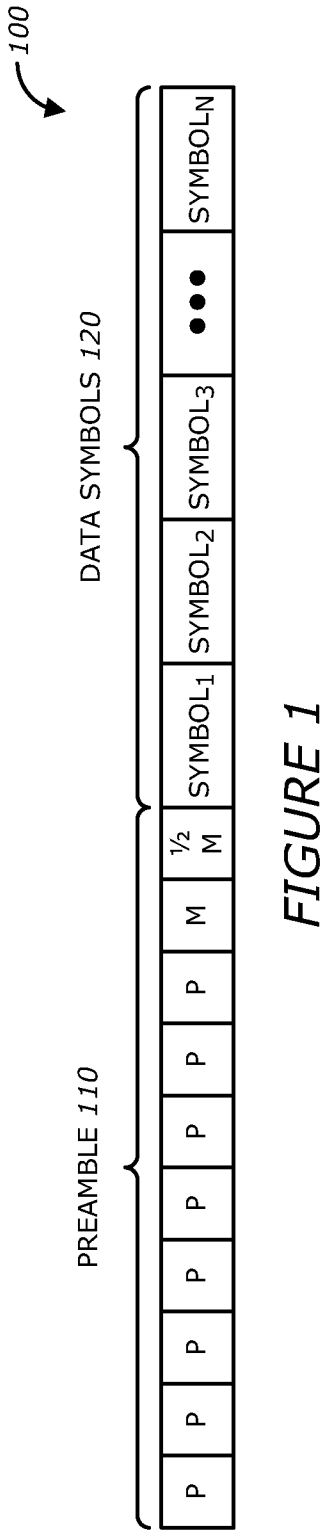

FIGURE 1

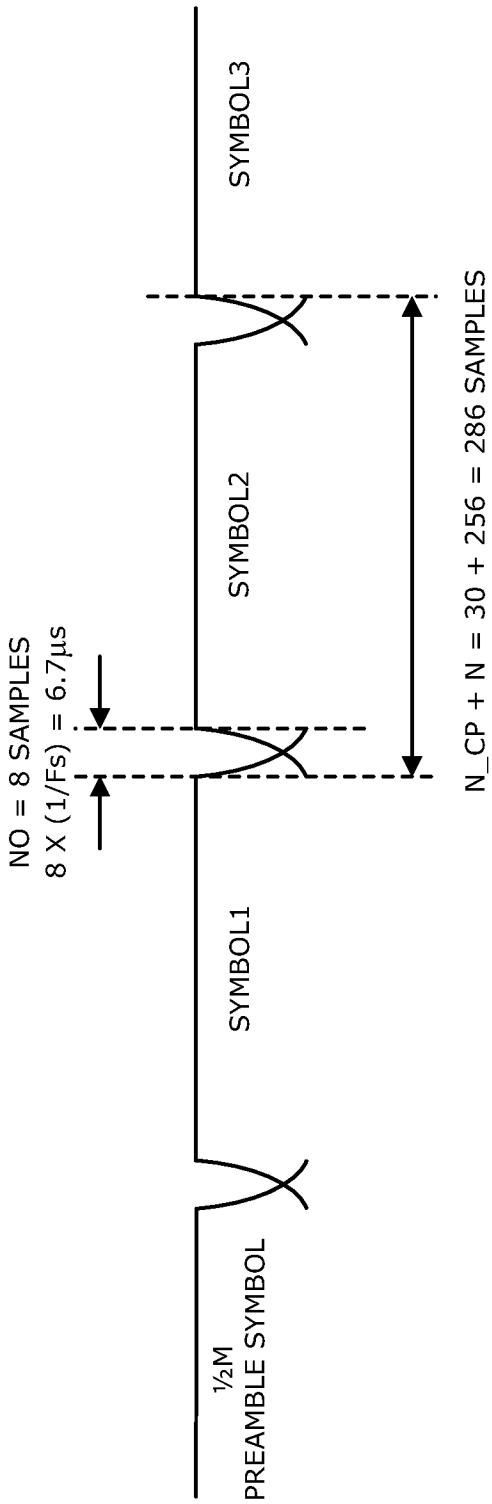

FIGURE 2

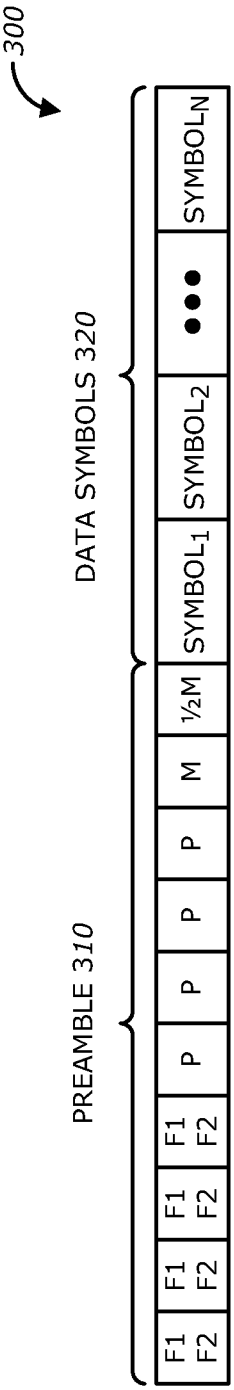

FIGURE 3

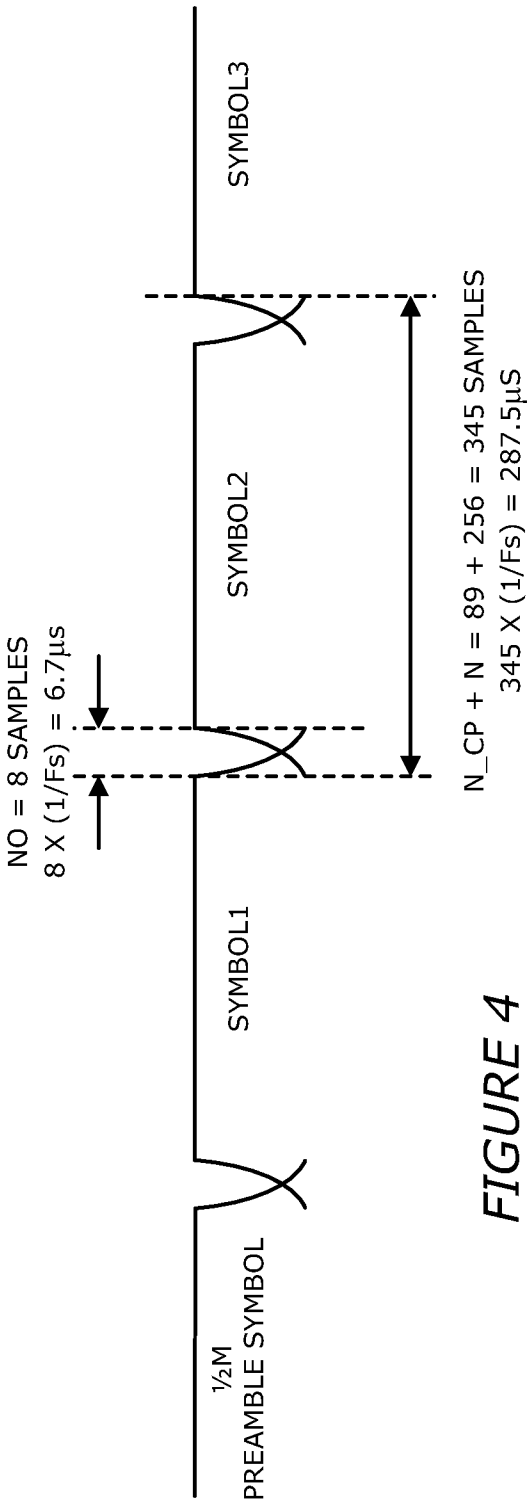

FIGURE 4

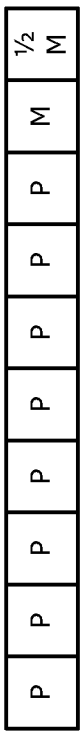

FIGURE 5

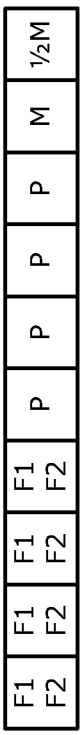

FIGURE 6

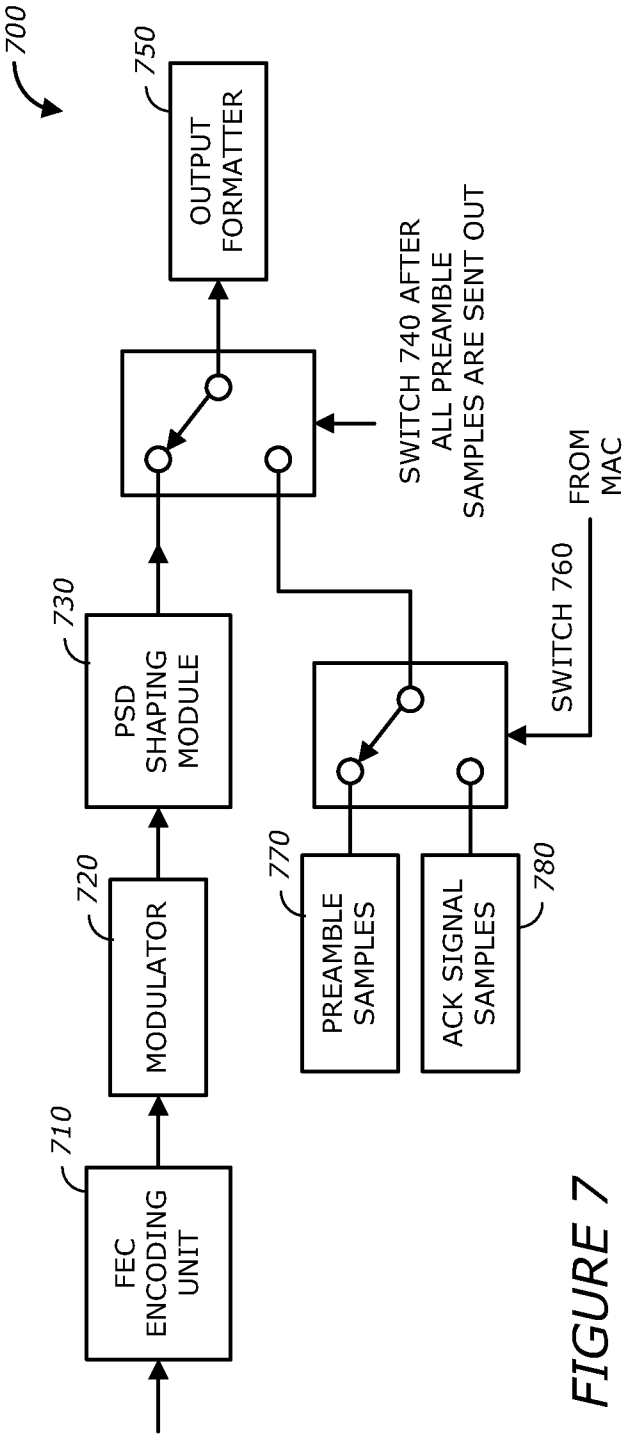

FIGURE 7

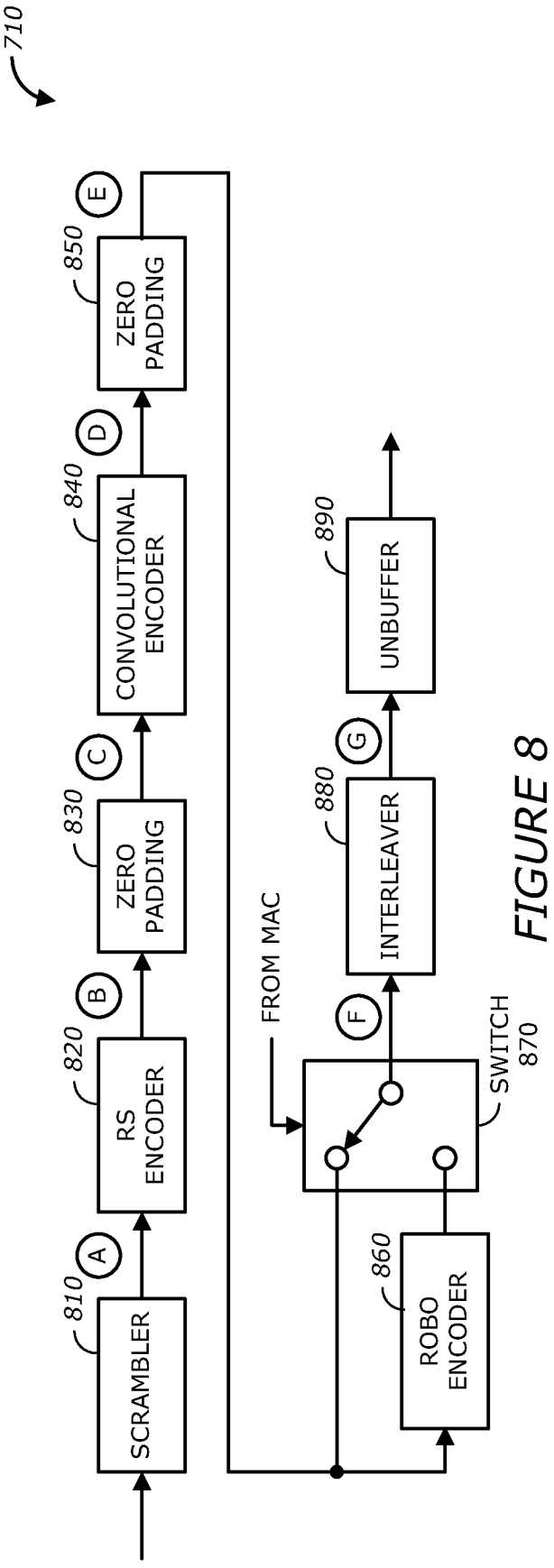

FIGURE 8

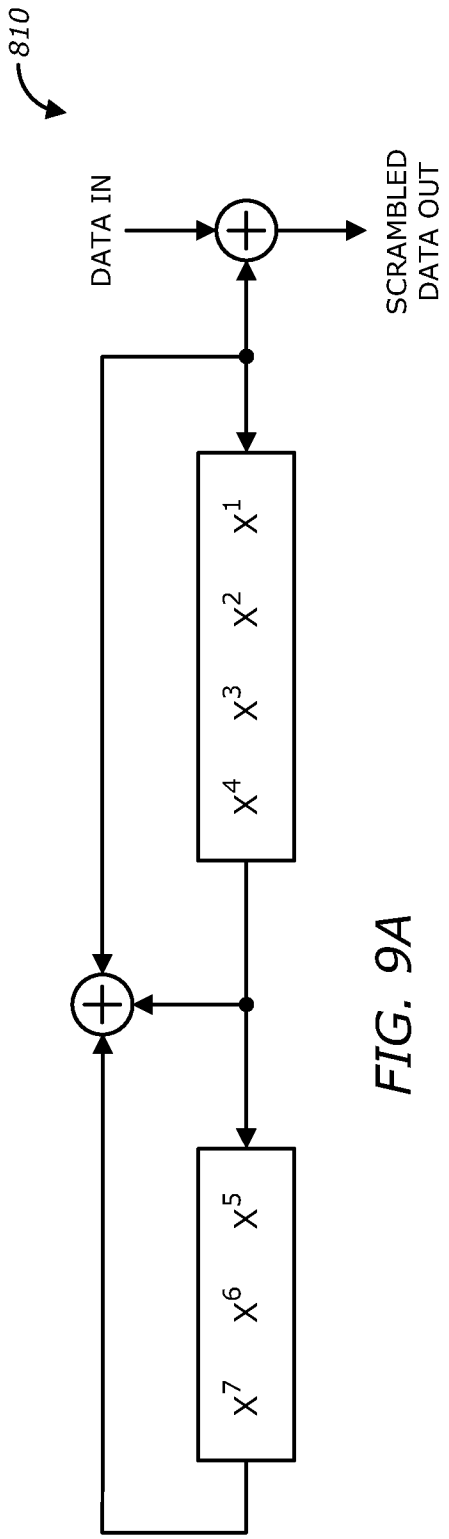
810
DATA IN
SCRAMBLED DATA OUT
X1
X2
X3
X4
X5
X6
X7

FIG. 9A

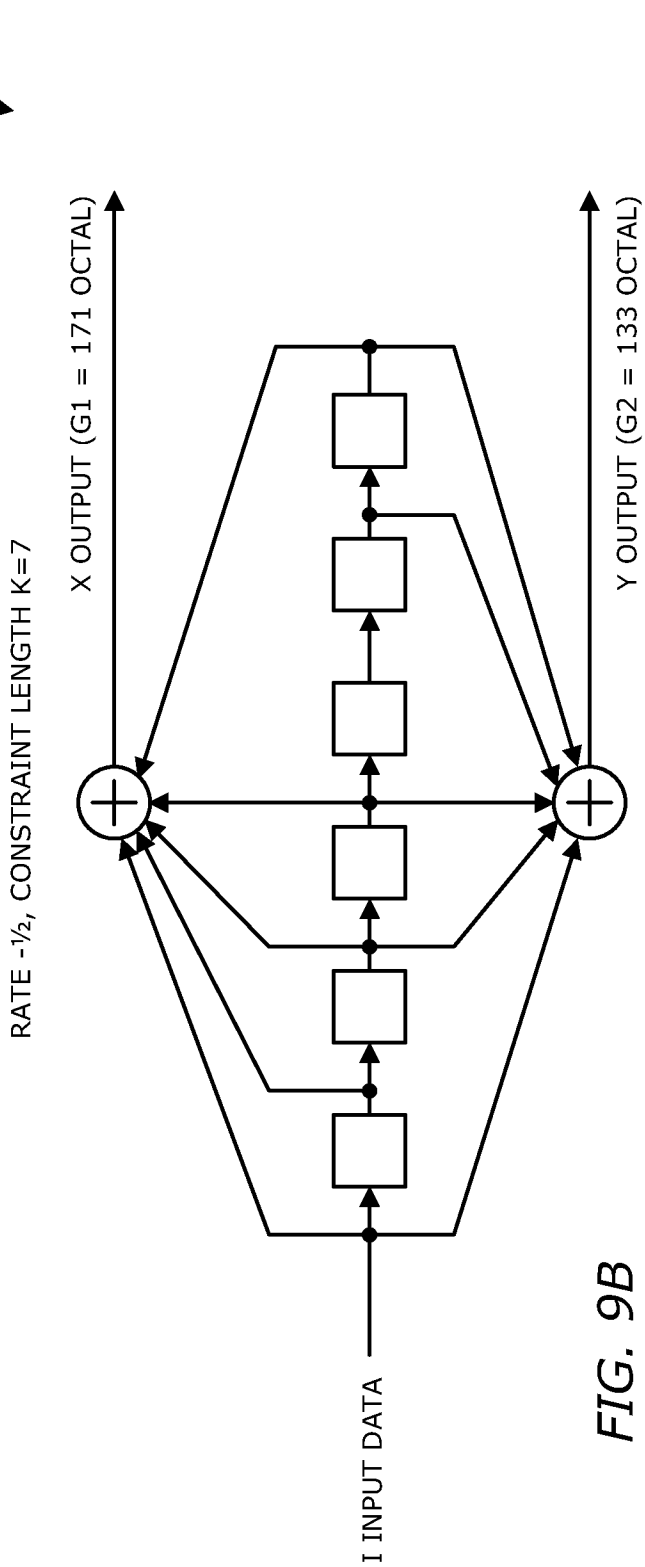
840
RATE -½, CONSTRAINT LENGTH K=7
X OUTPUT (G1 = 171 OCTAL)
Y OUTPUT (G2 = 133 OCTAL)
I INPUT DATA

FIG. 9B

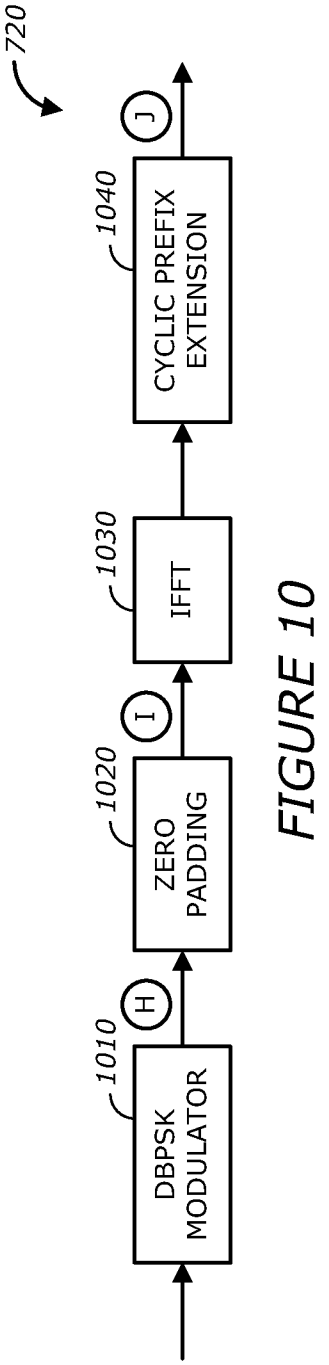

FIGURE 10

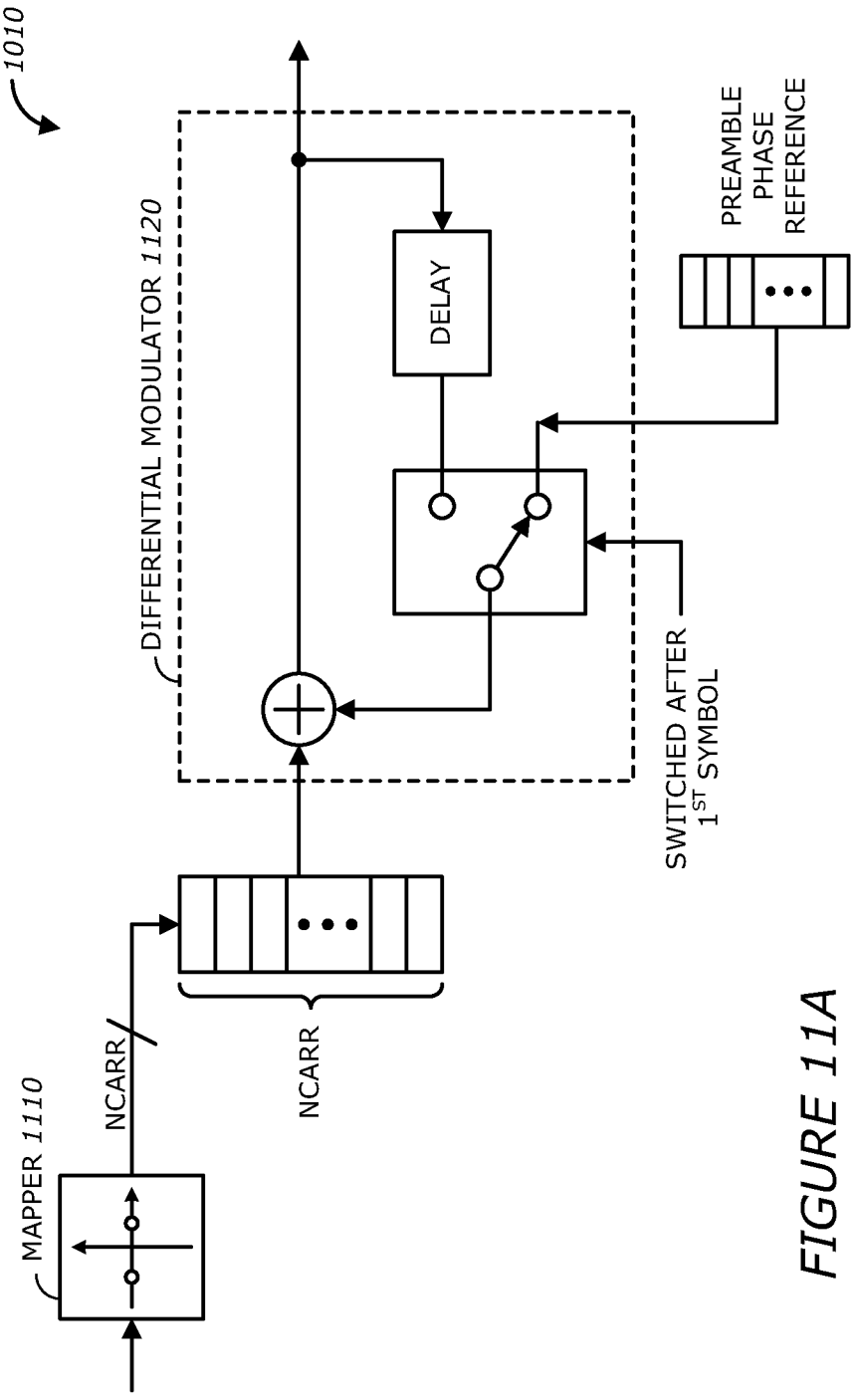

FIGURE 11A

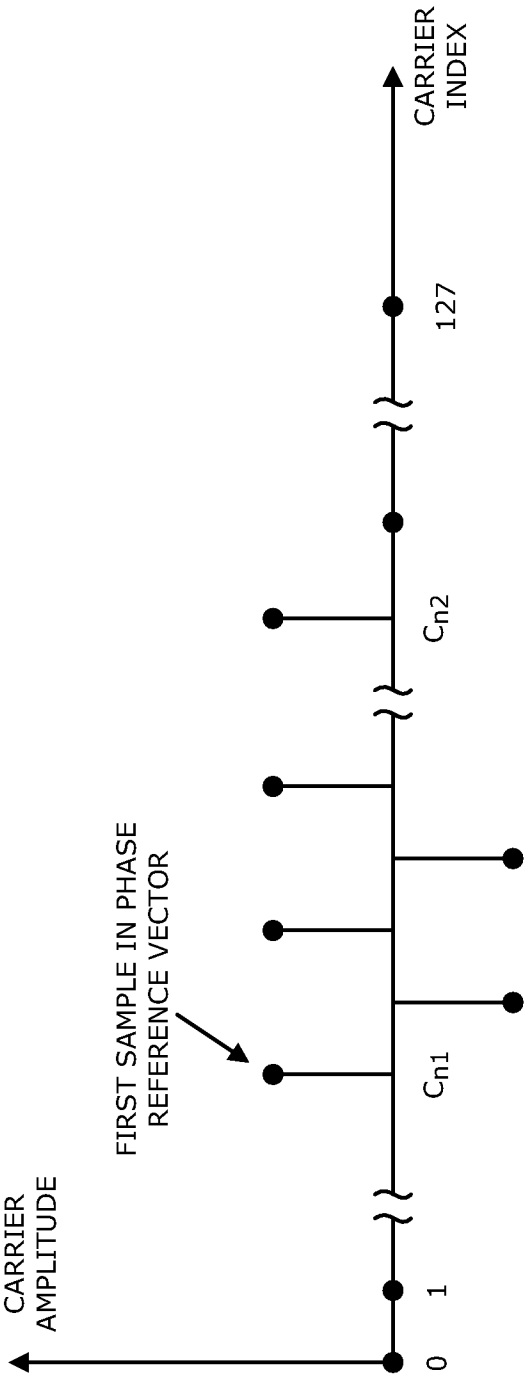

FIGURE 11B

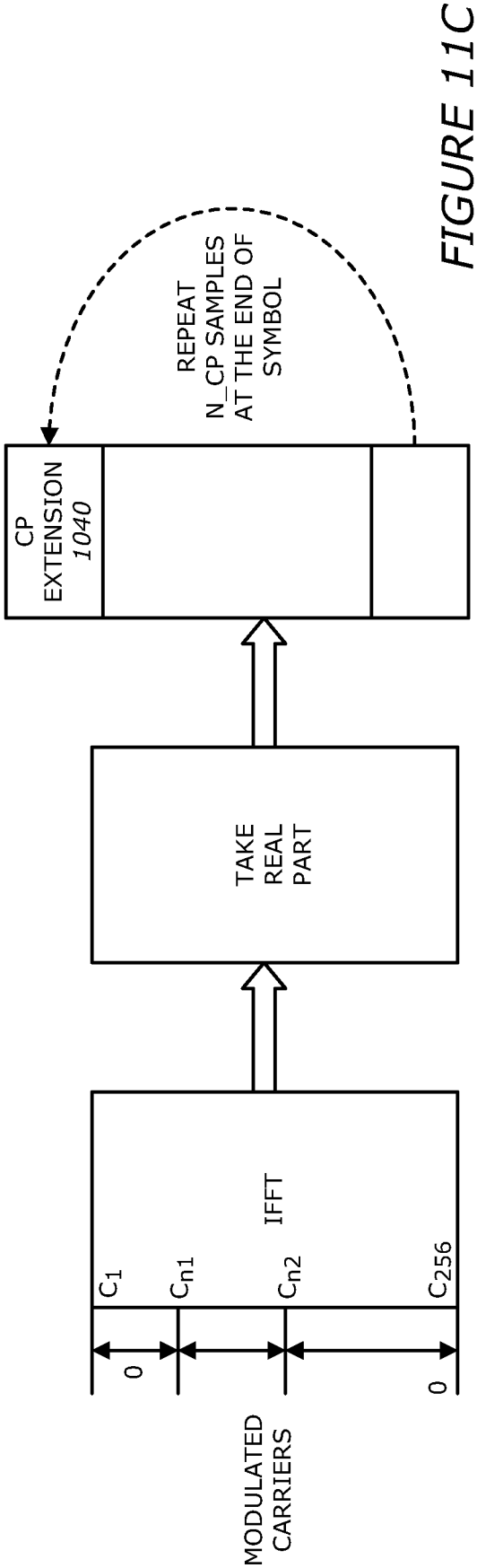

FIGURE 11C

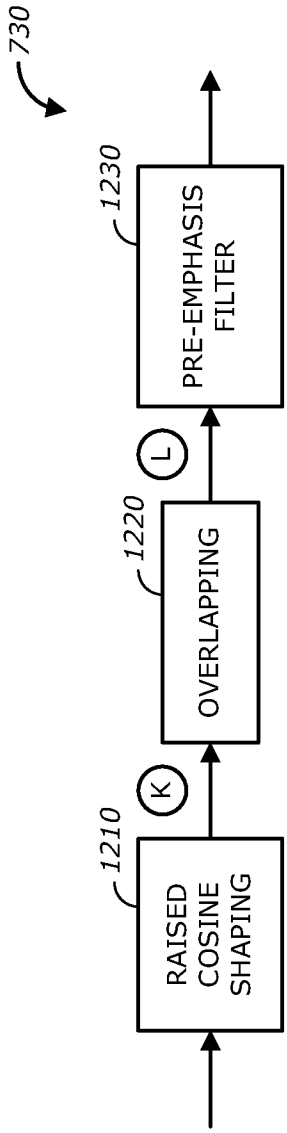

FIGURE 12

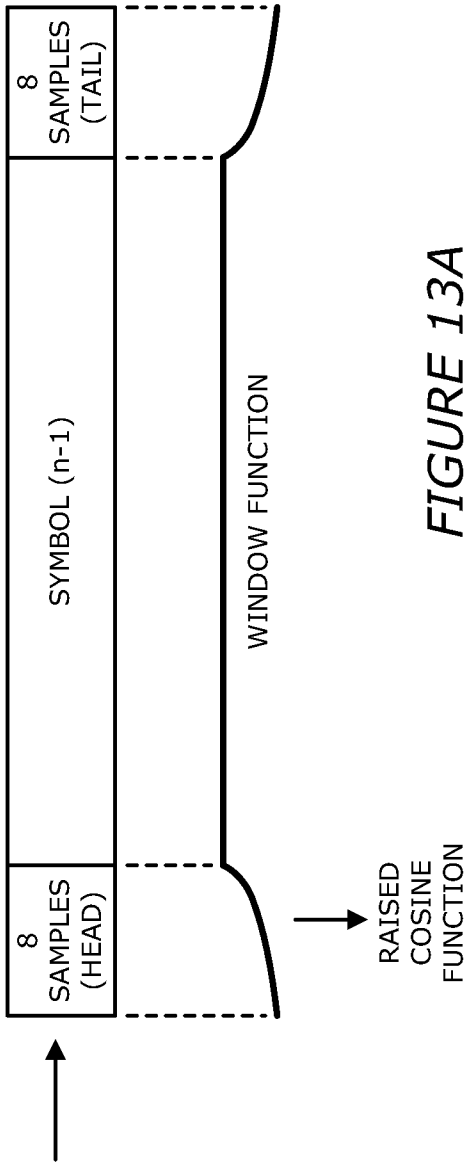

FIGURE 13A

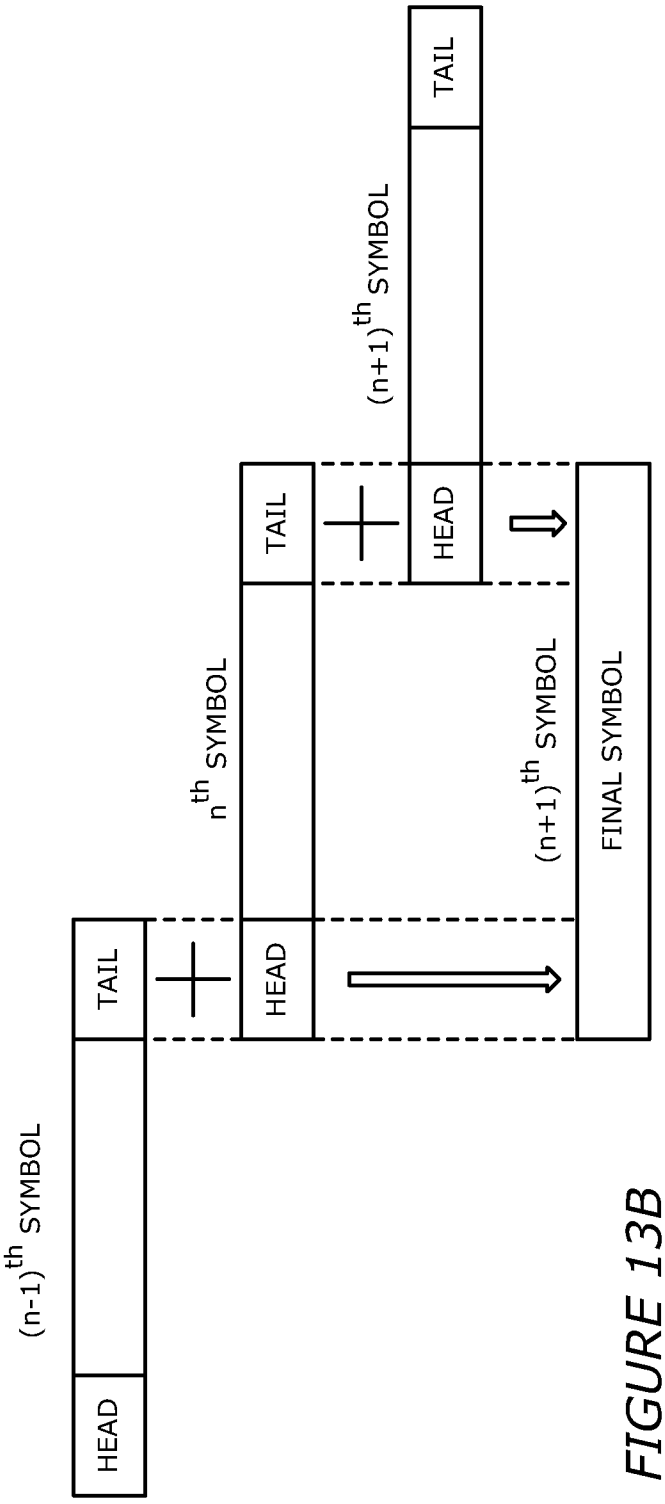

FIGURE 13B

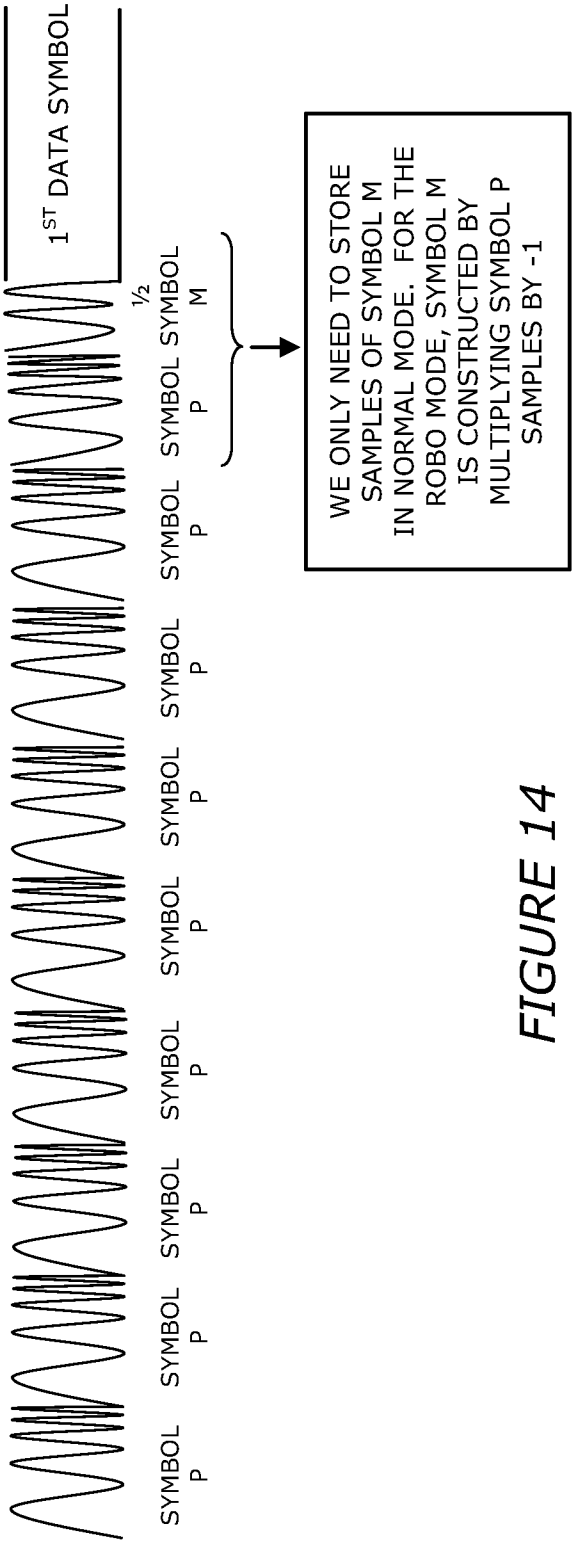

FIGURE 14

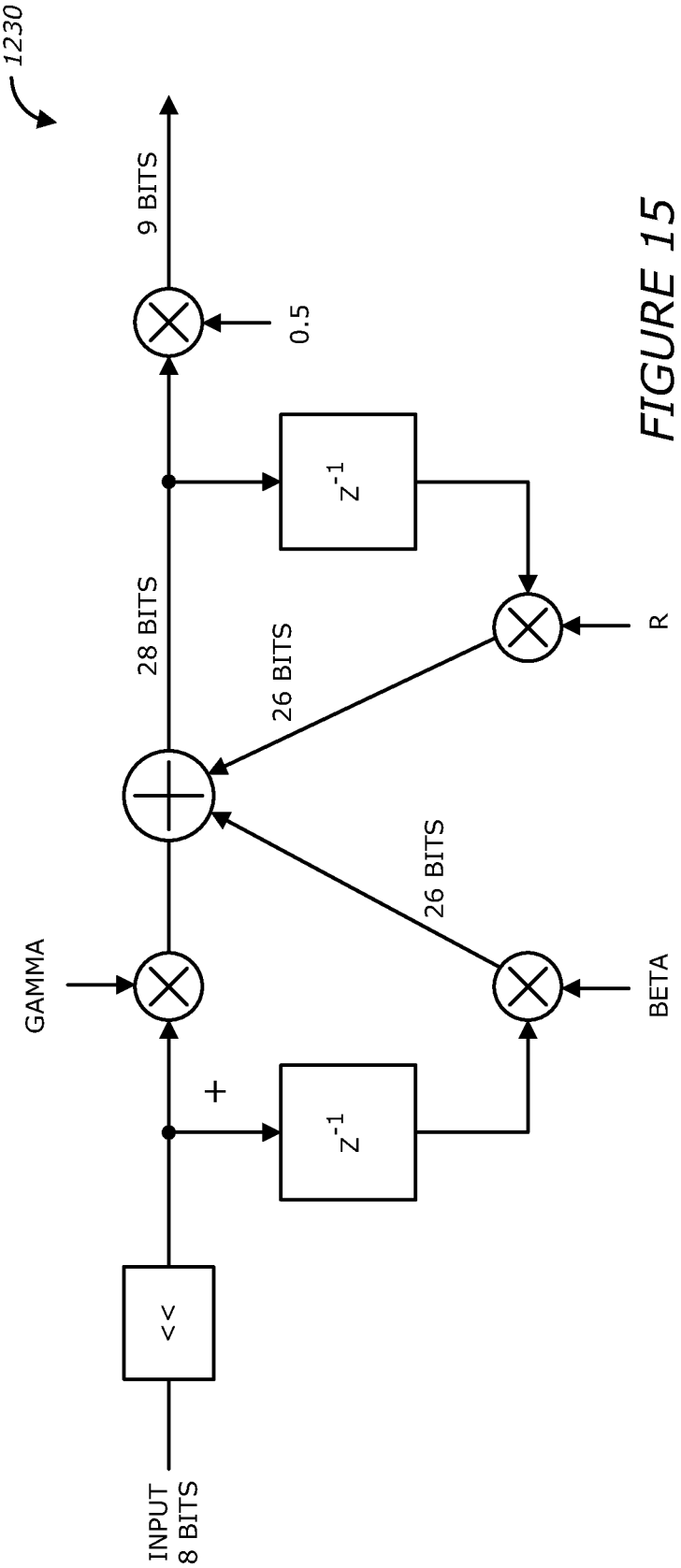

FIGURE 15

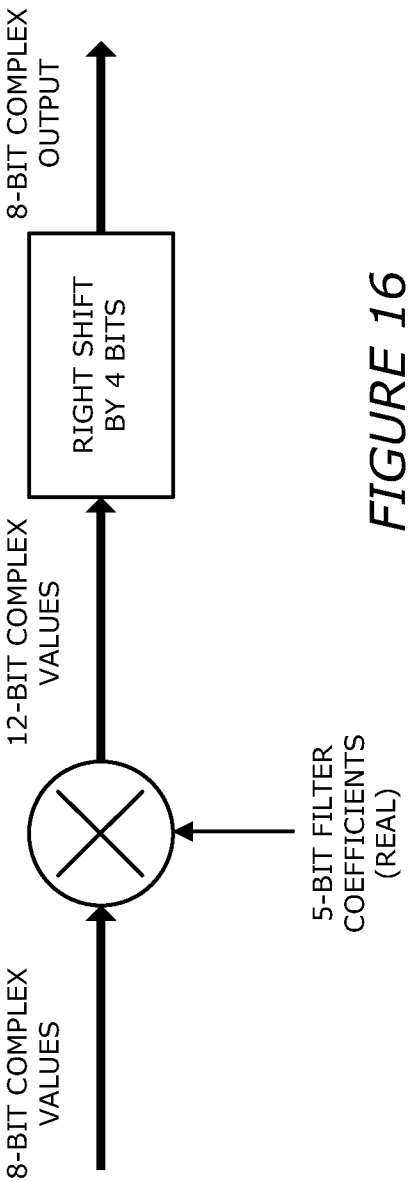

FIGURE 16

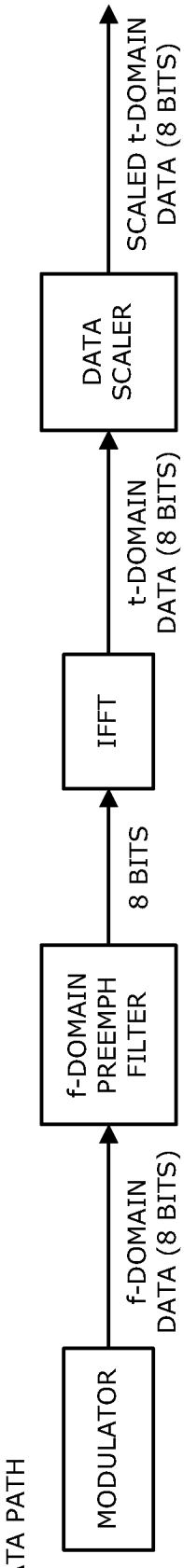

FIGURE 17A

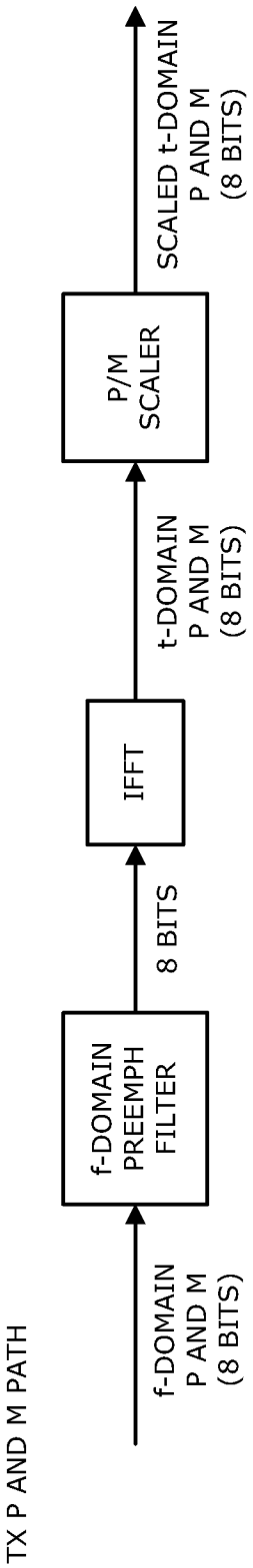

FIGURE 17B

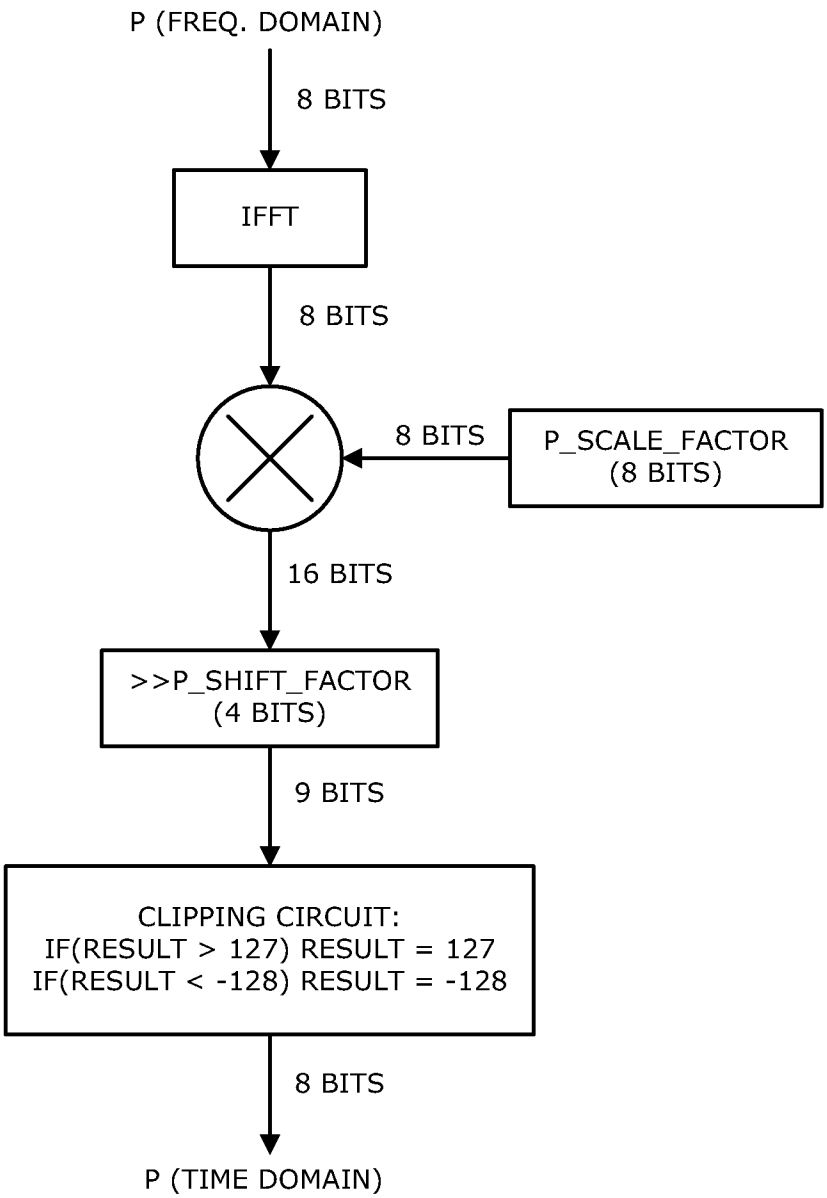
P (FREQ. DOMAIN)
8 BITS
IFFT
8 BITS
8 BITS
P_SCALE_FACTOR
(8 BITS)
16 BITS
>>P_SHIFT_FACTOR
(4 BITS)
9 BITS
CLIPPING CIRCUIT:
IF(RESULT > 127) RESULT = 127
IF(RESULT < -128) RESULT = -128
8 BITS
P (TIME DOMAIN)

FIGURE 17C

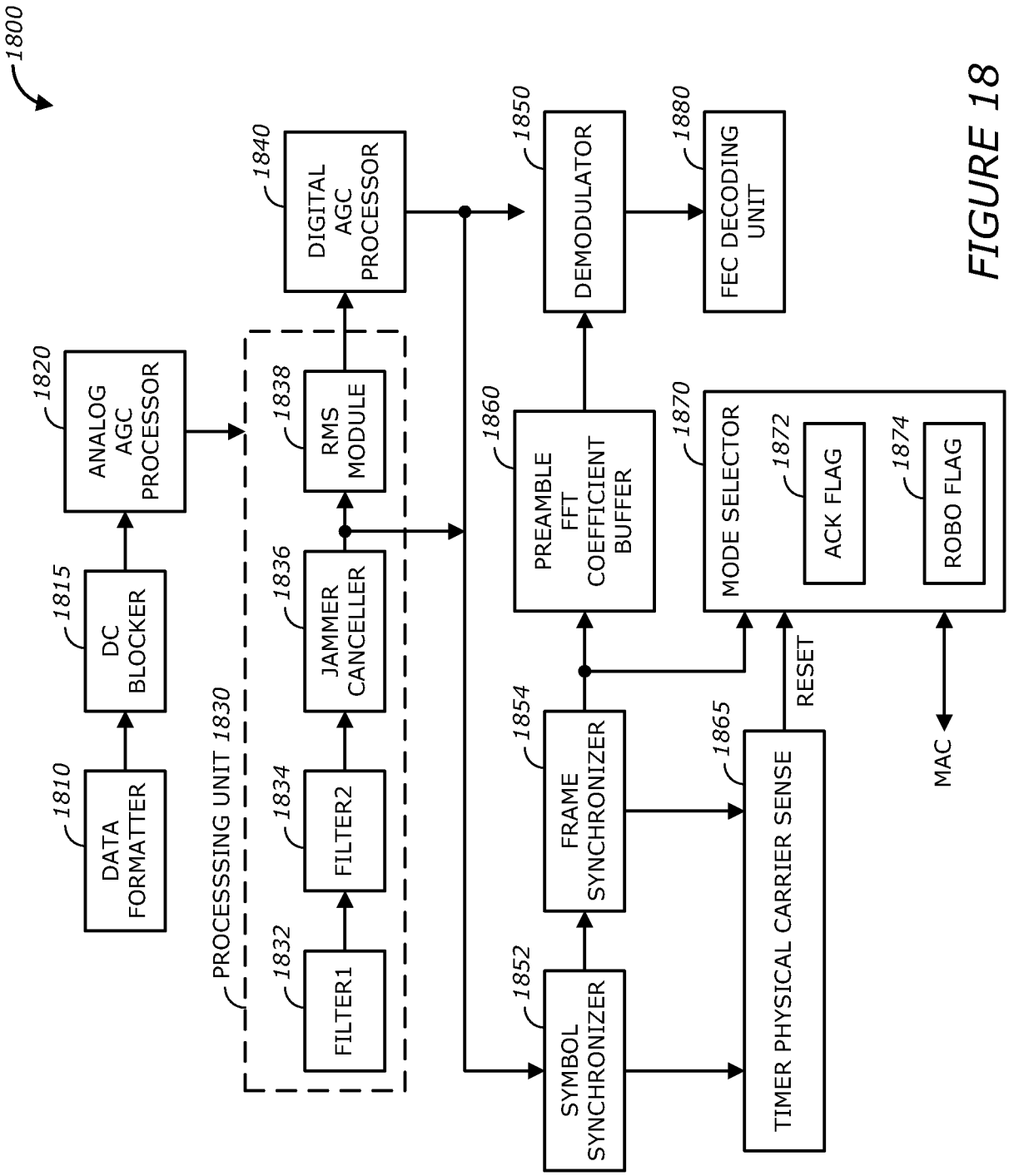
1800
DATA FORMATTER
1810
DC BLOCKER
1815
ANALOG AGC PROCESSOR
1820
PROCESSING UNIT 1830
FILTER1
1832
FILTER2
1834
JAMMER CANCELLER
1836
RMS MODULE
1838
DIGITAL AGC PROCESSOR
1840
SYMBOL SYNCHRONIZER
1852
FRAME SYNCHRONIZER
1854
PREAMBLE FFT COEFFICIENT BUFFER
1860
DEMODULATOR
1850
FEC DECODING UNIT
1880
TIMER PHYSICAL CARRIER SENSE
1865
MODE SELECTOR
1870
ACK FLAG
1872
ROBO FLAG
1874
RESET
MAC

FIGURE 18

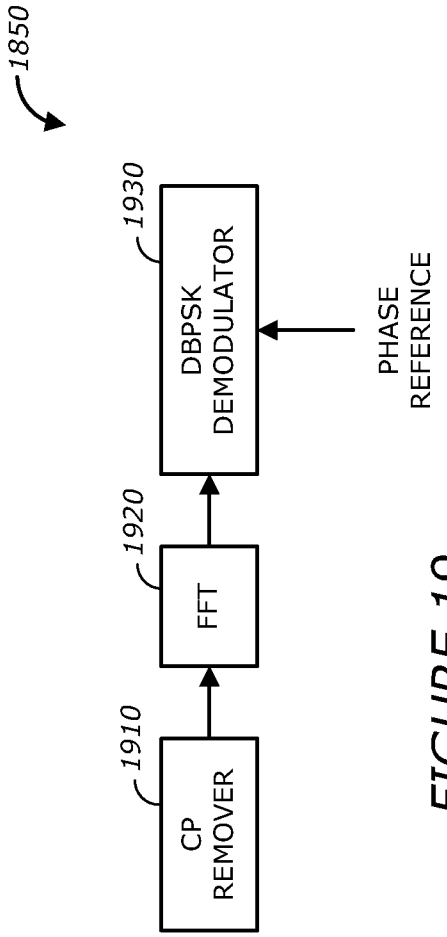

FIGURE 19

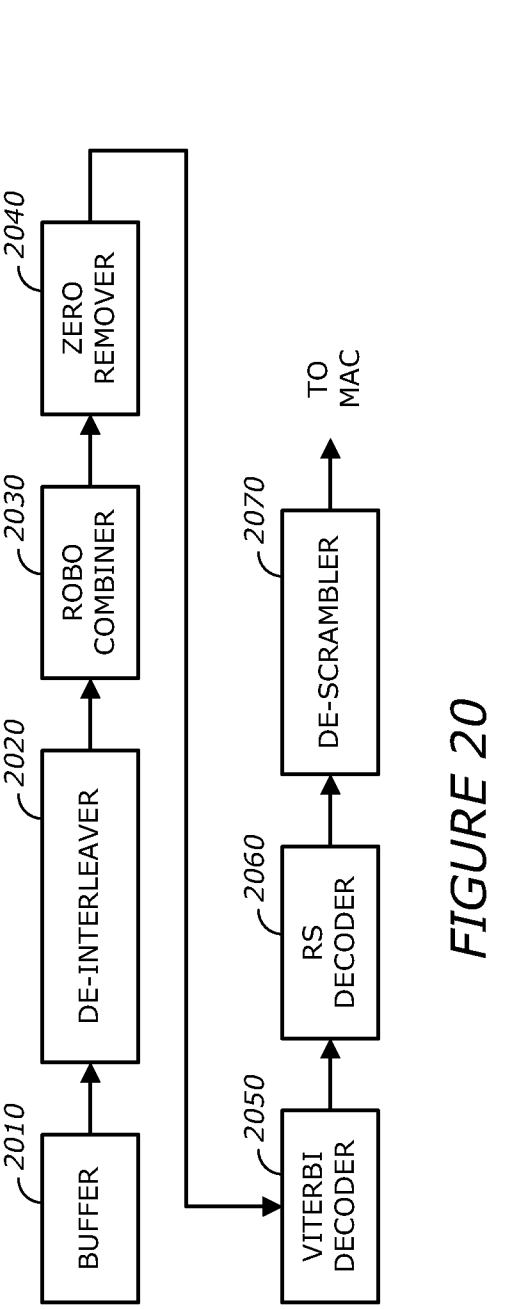

FIGURE 20

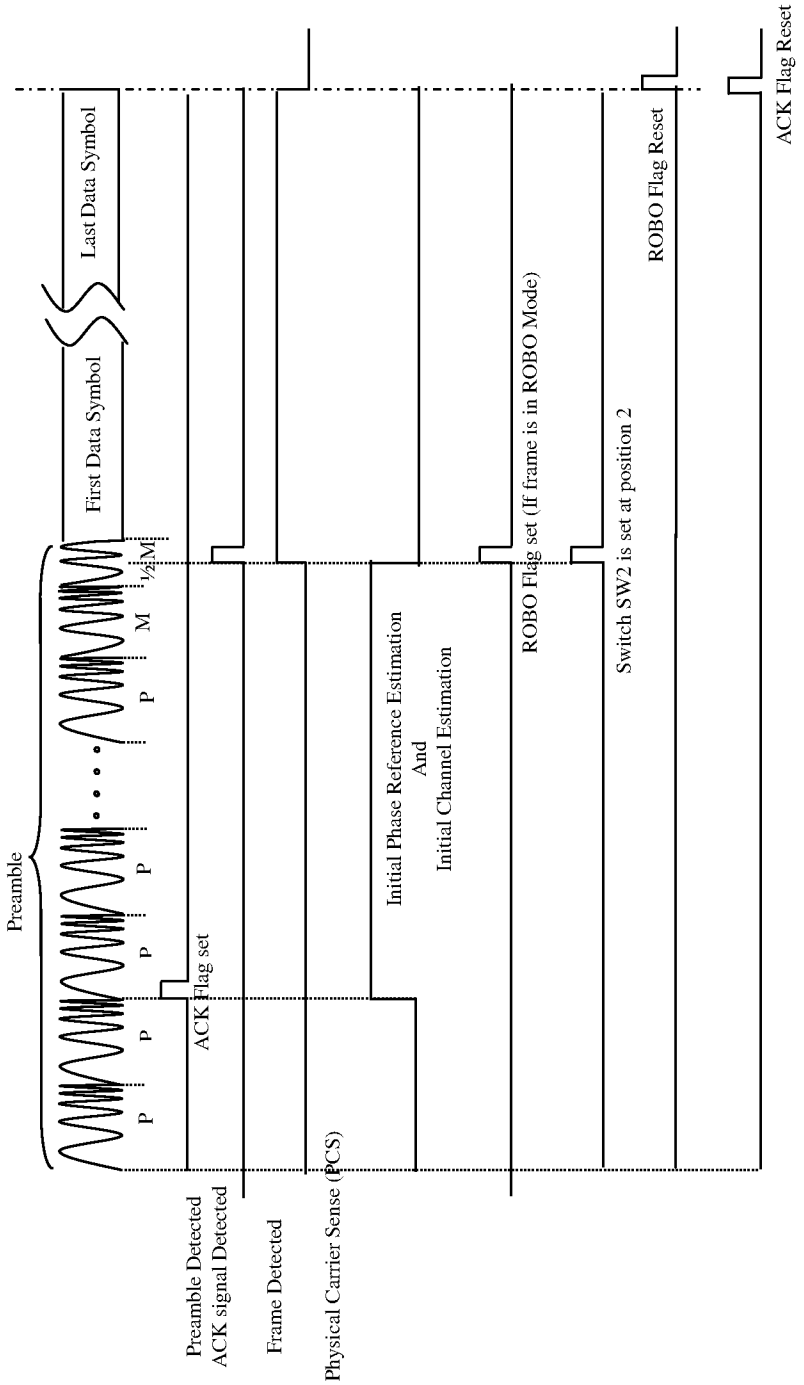
Preamble
P
P
P
P
P
M
½M
First Data Symbol
Last Data Symbol
ACK Flag set
Initial Phase Reference Estimation
And
Initial Channel Estimation
ROBO Flag set (If frame is in ROBO Mode)
Switch SW2 is set at position 2
ROBO Flag Reset
ACK Flag Reset
Preamble Detected
ACK signal Detected
Frame Detected
Physical Carrier Sense (PCS)

FIGURE 21

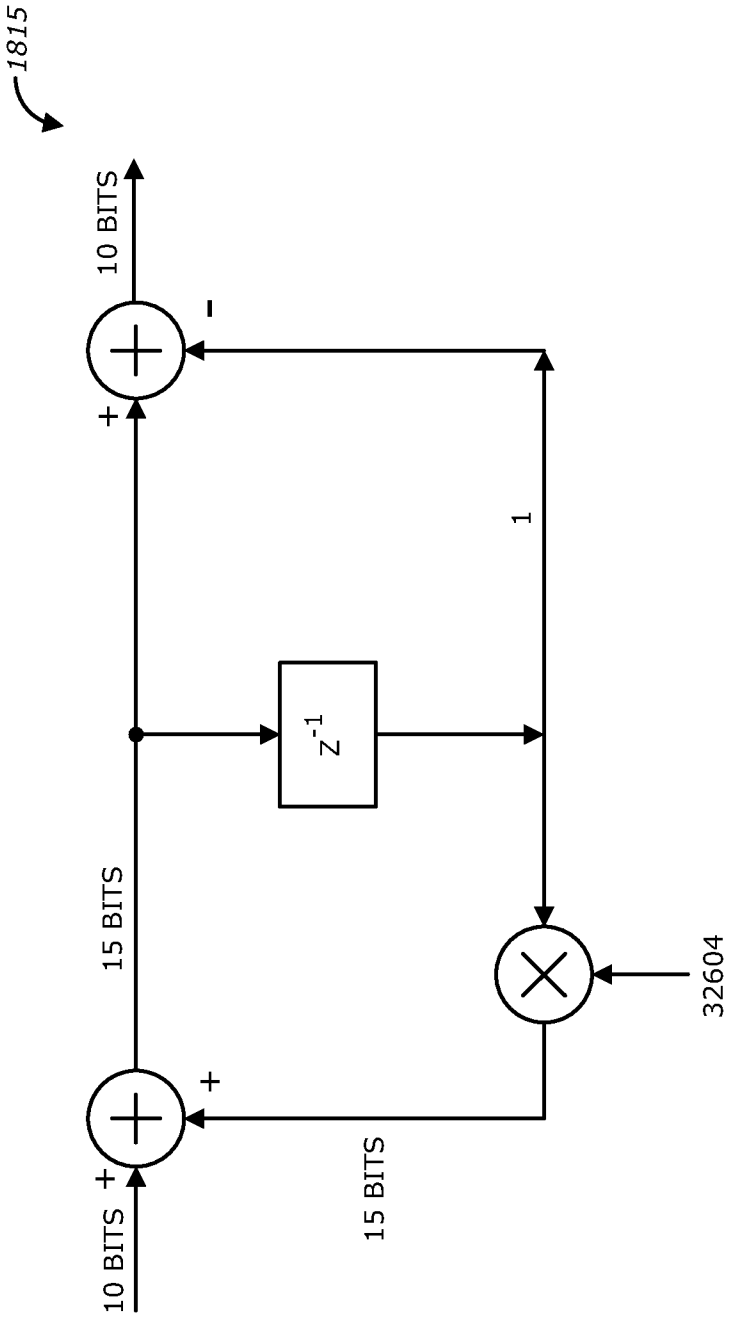

FIGURE 22

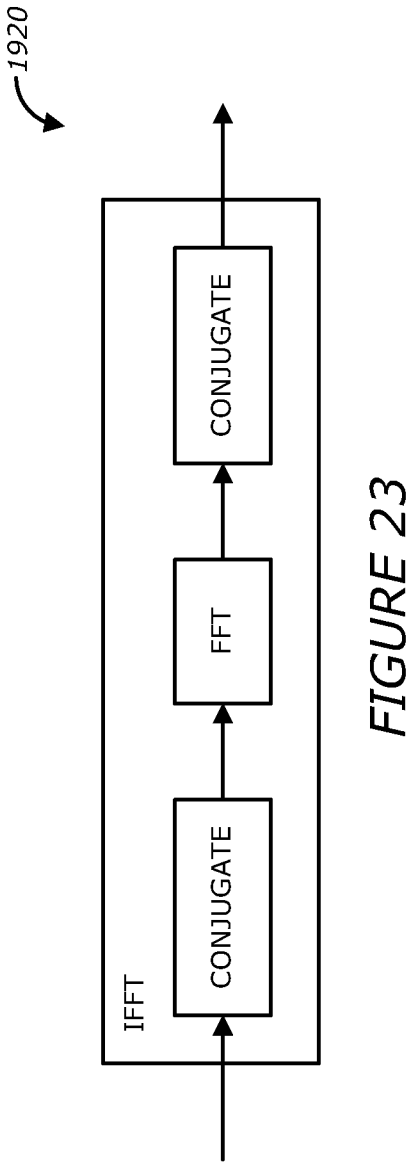

FIGURE 23

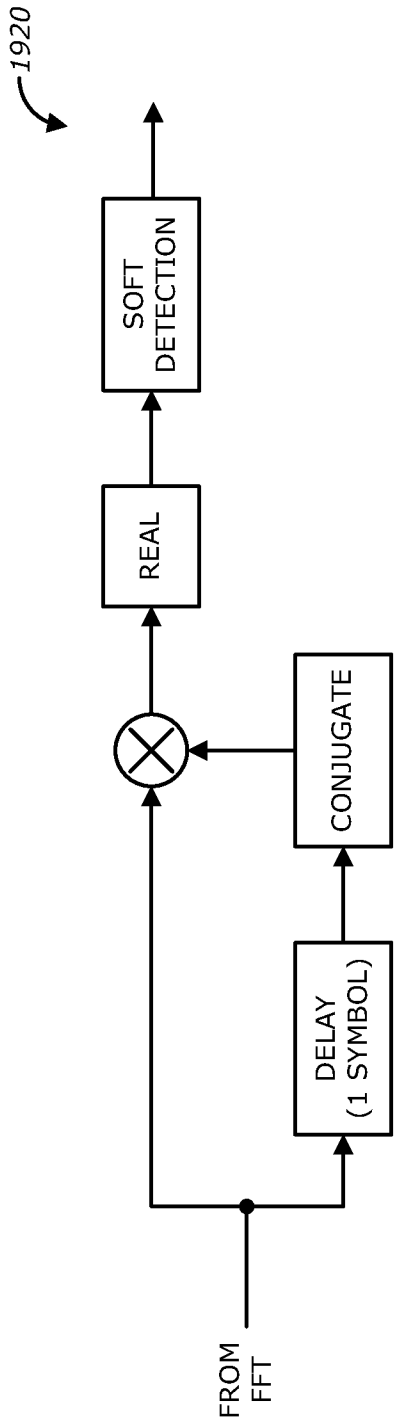

FIGURE 24

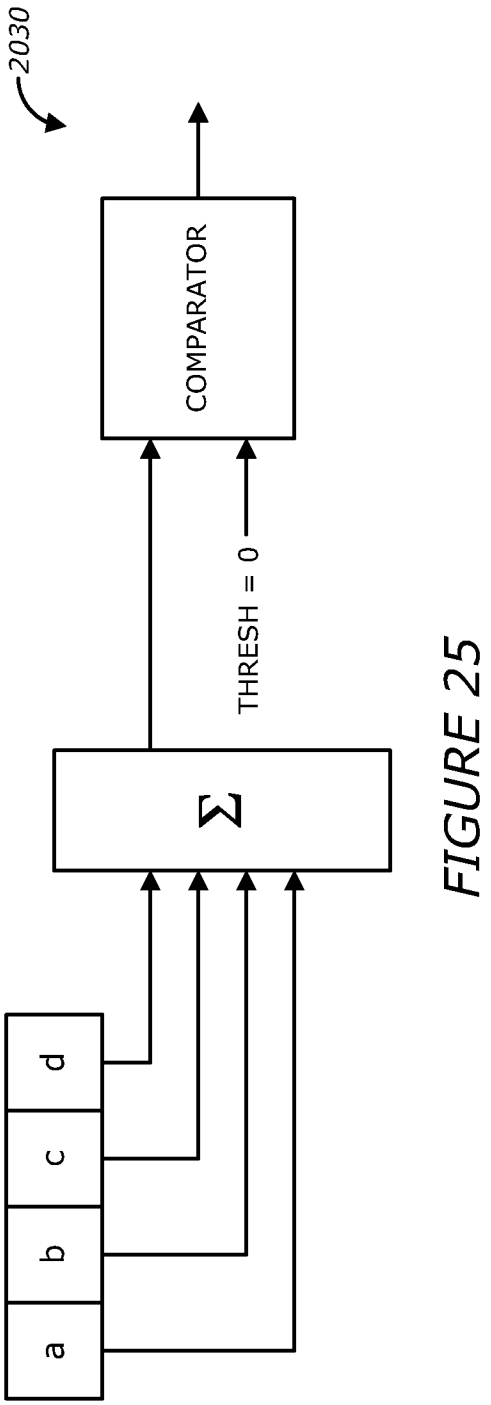

FIGURE 25

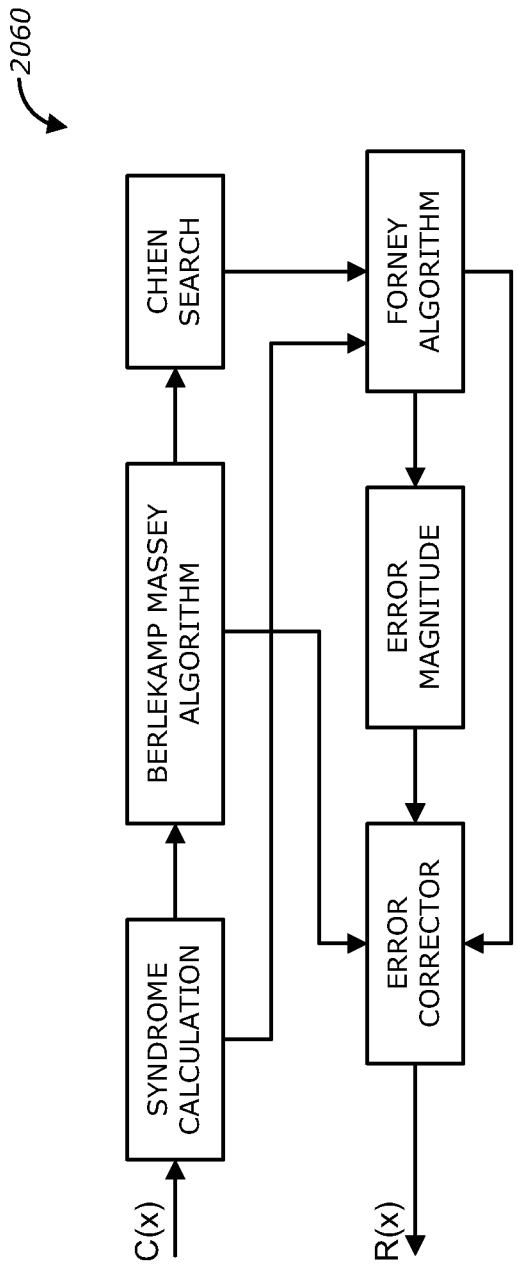
2060
CHIEN SEARCH
FORNEY ALGORITHM
BERLEKAMP MASSEY ALGORITHM
ERROR MAGNITUDE
SYNDROME CALCULATION
ERROR CORRECTOR
C(x)
R(x)

FIGURE 26

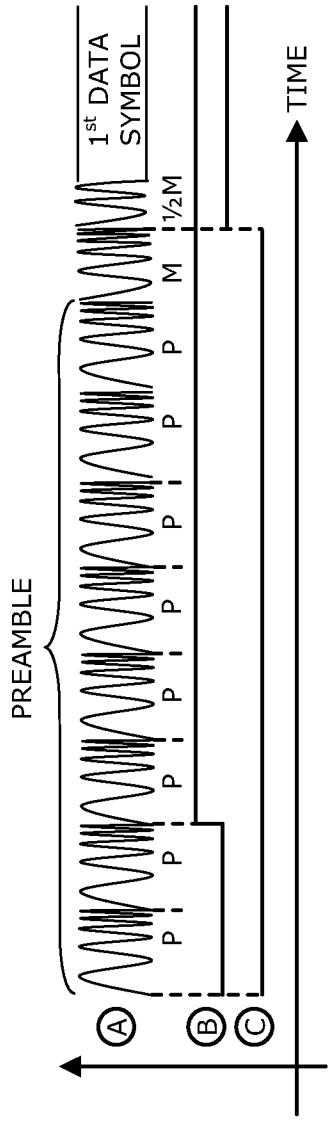

FIGURE 27A

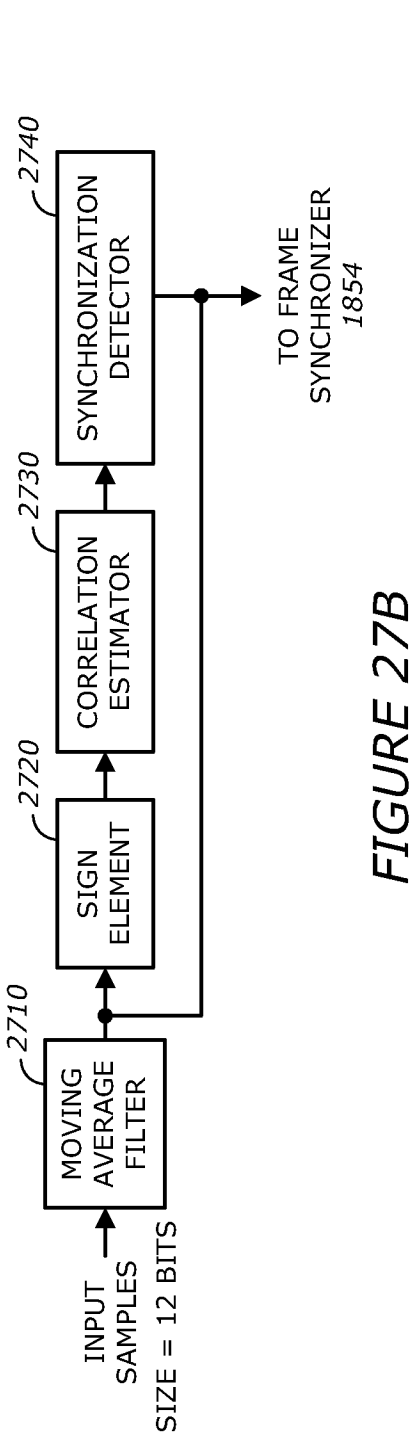

FIGURE 27B

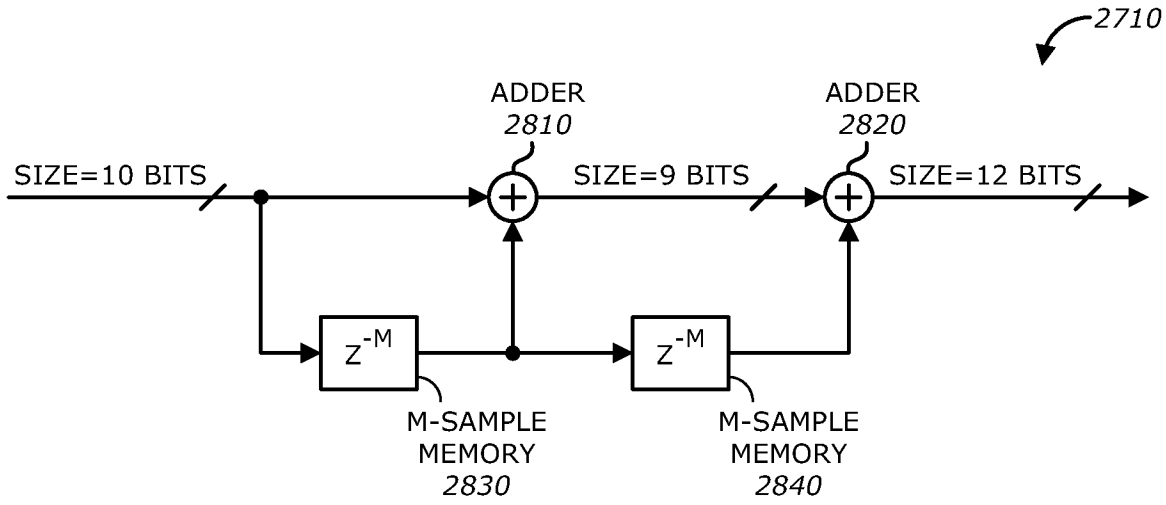

FIGURE 28

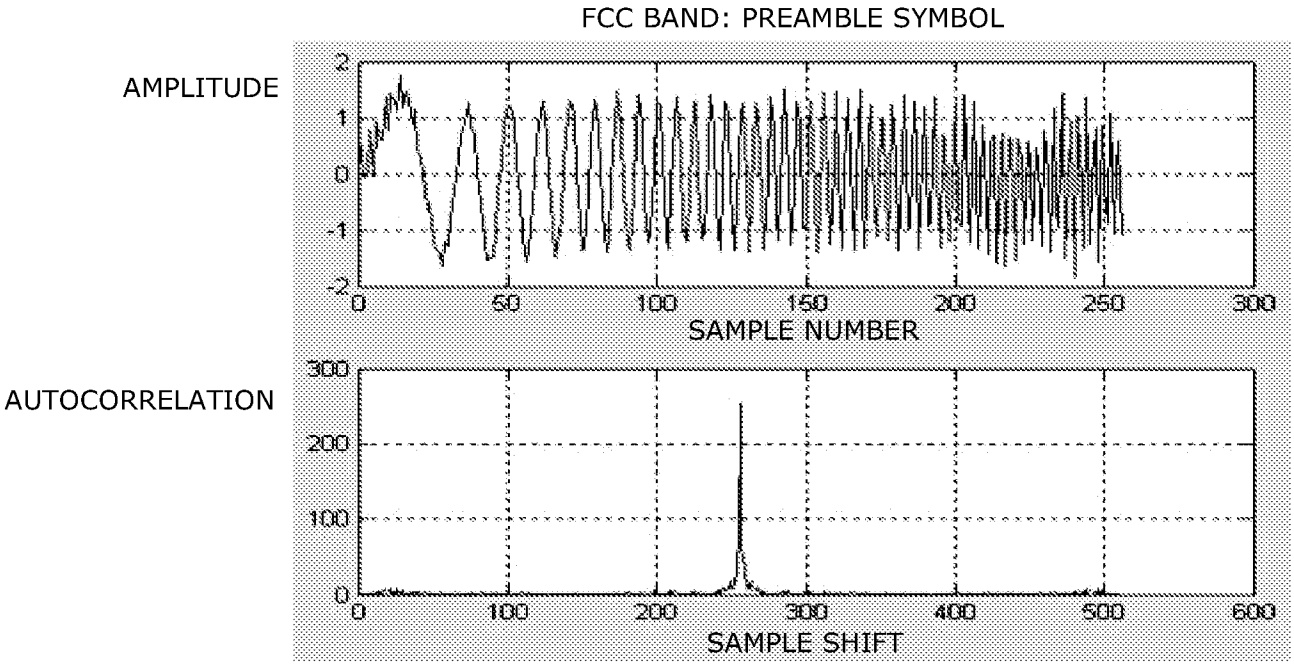

FIGURE 29

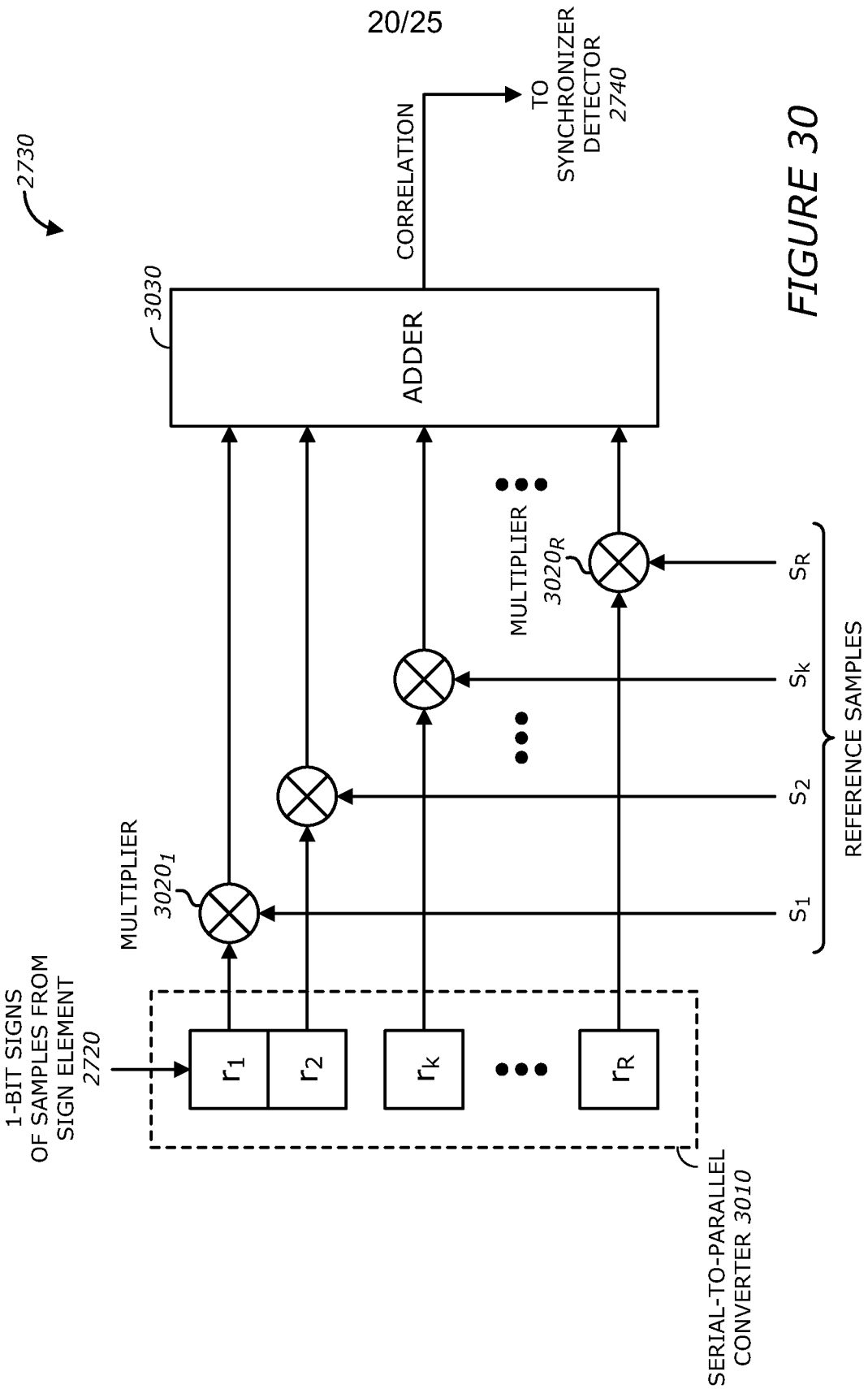
2730
3030
ADDER
CORRELATION
TO SYNCHRONIZER DETECTOR 2740
MULTIPLIER 3020R
MULTIPLIER 3020₁
1-BIT SIGNS OF SAMPLES FROM SIGN ELEMENT 2720
r1
r2
rk
rR
S1
S2
Sk
SR
REFERENCE SAMPLES
SERIAL-TO-PARALLEL CONVERTER 3010

FIGURE 30

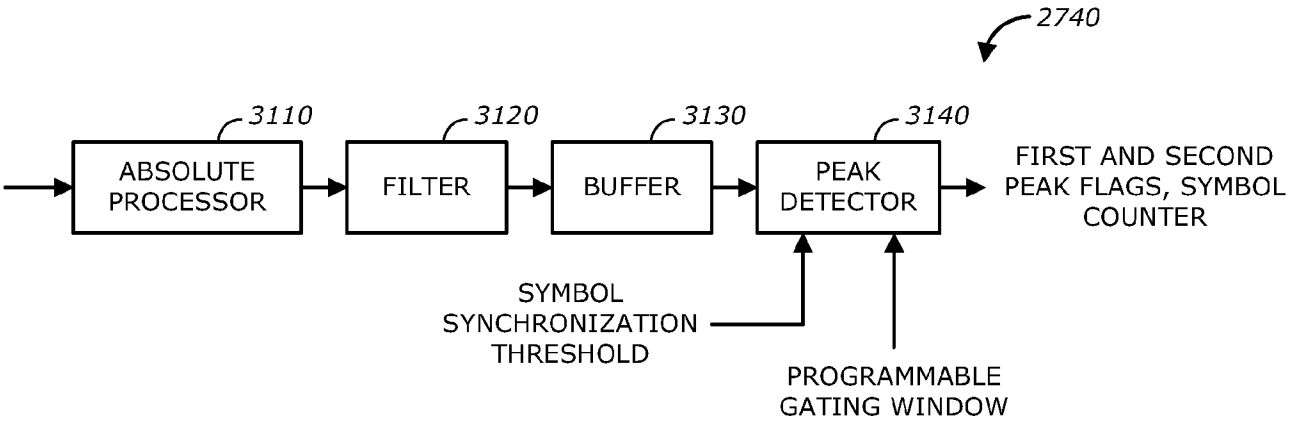

FIGURE 31

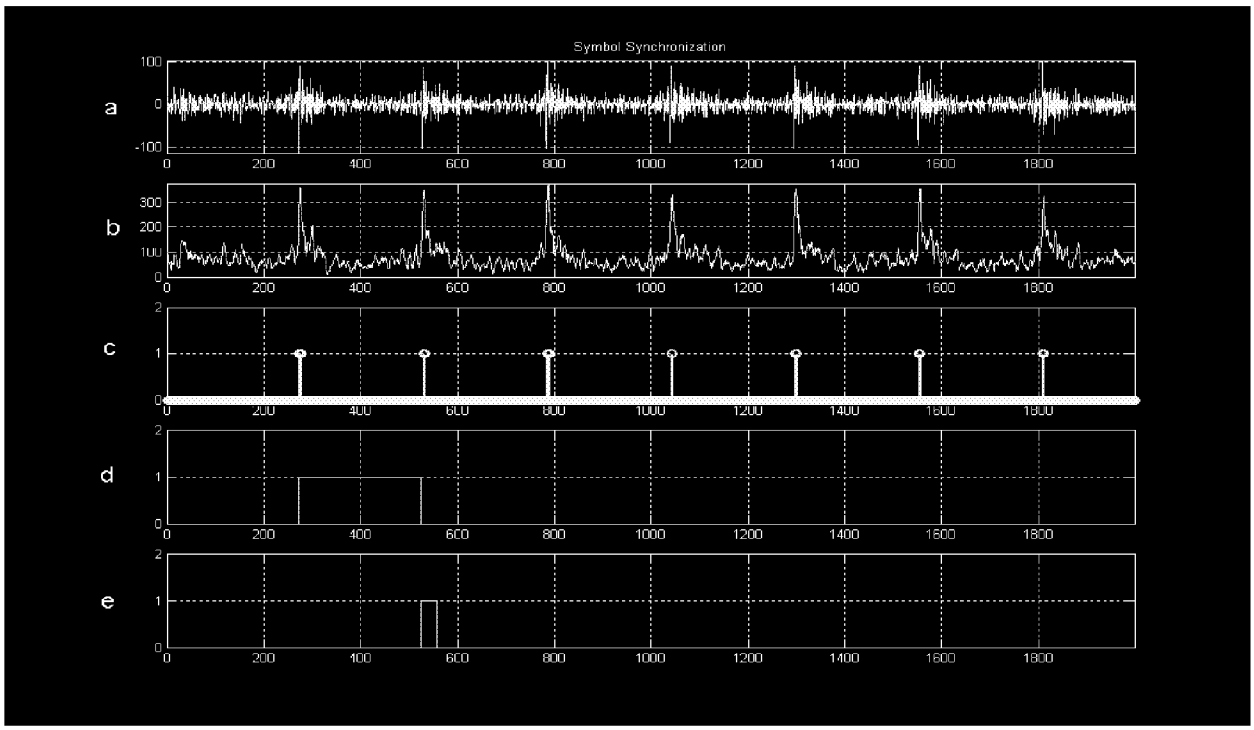

FIGURE 32

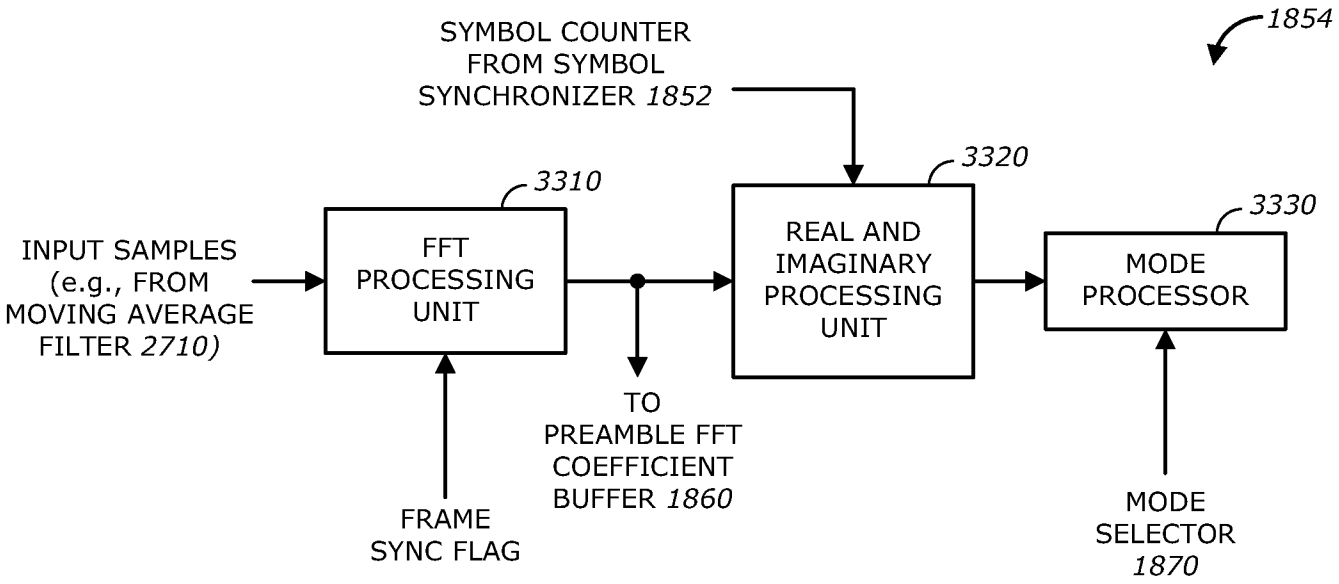

FIGURE 33

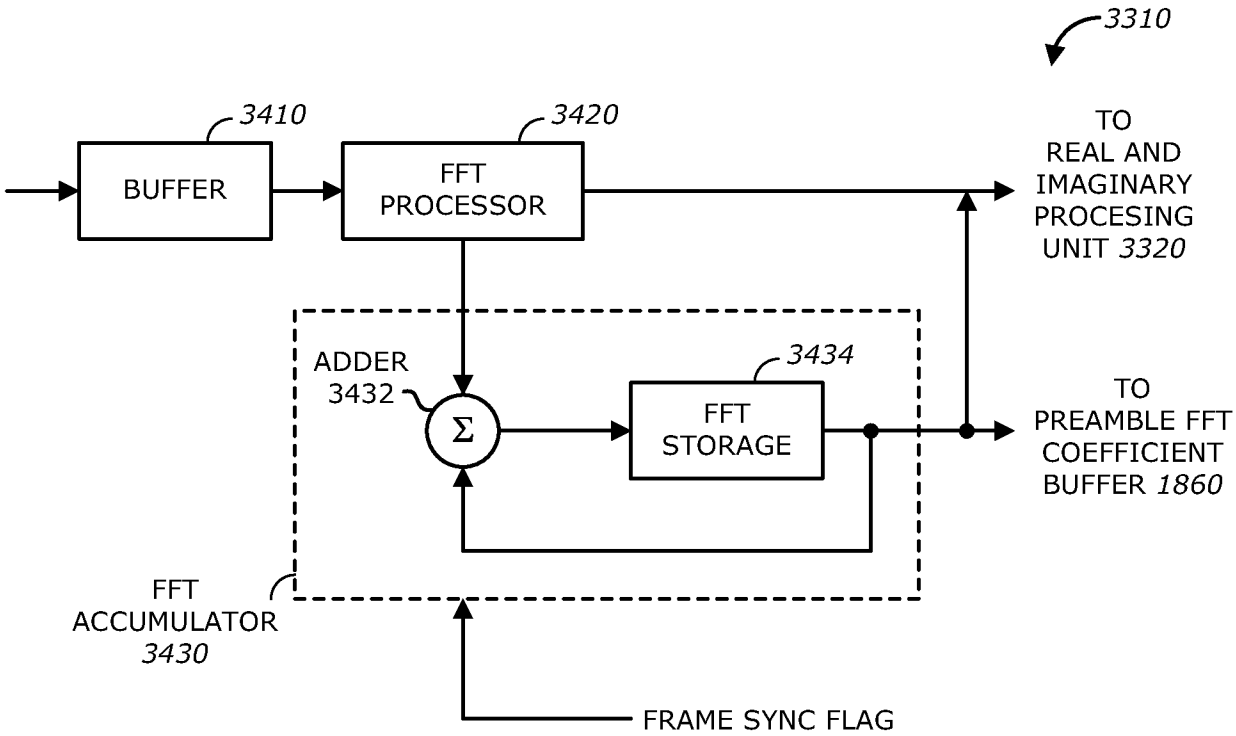

FIGURE 34

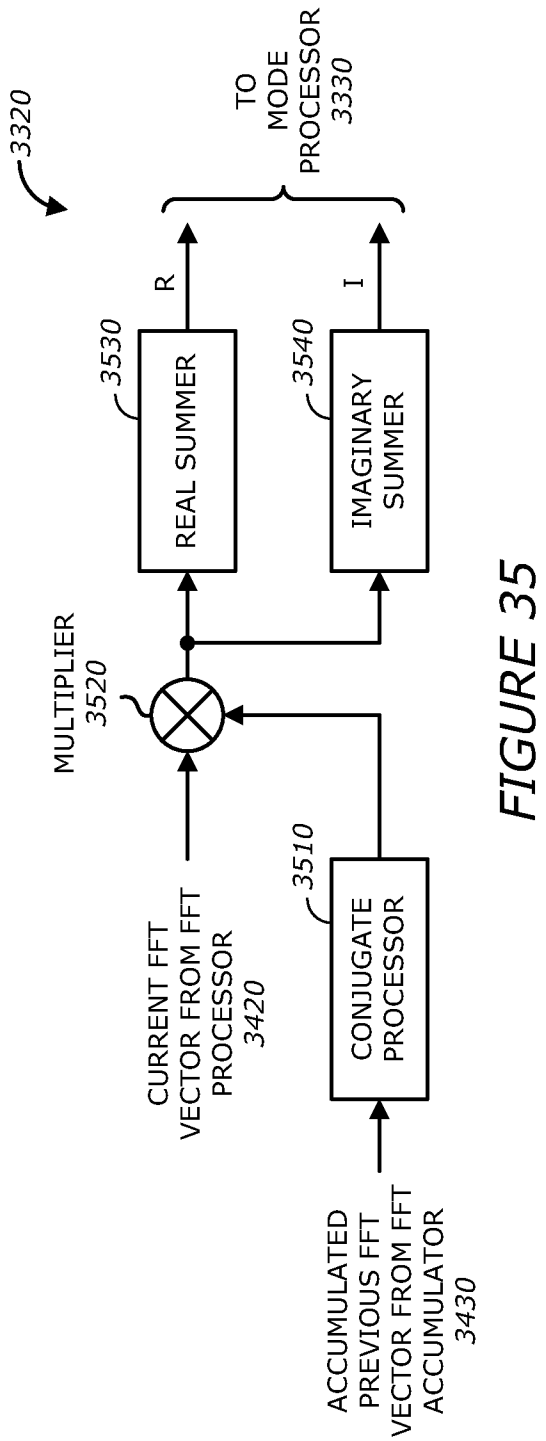
3320
TO MODE PROCESSOR 3330
R
I
3530
REAL SUMMER
3540
IMAGINARY SUMMER
MULTIPLIER 3520
CURRENT FFT VECTOR FROM FFT PROCESSOR 3420
3510
CONJUGATE PROCESSOR
ACCUMULATED PREVIOUS FFT VECTOR FROM FFT ACCUMULATOR 3430

FIGURE 35

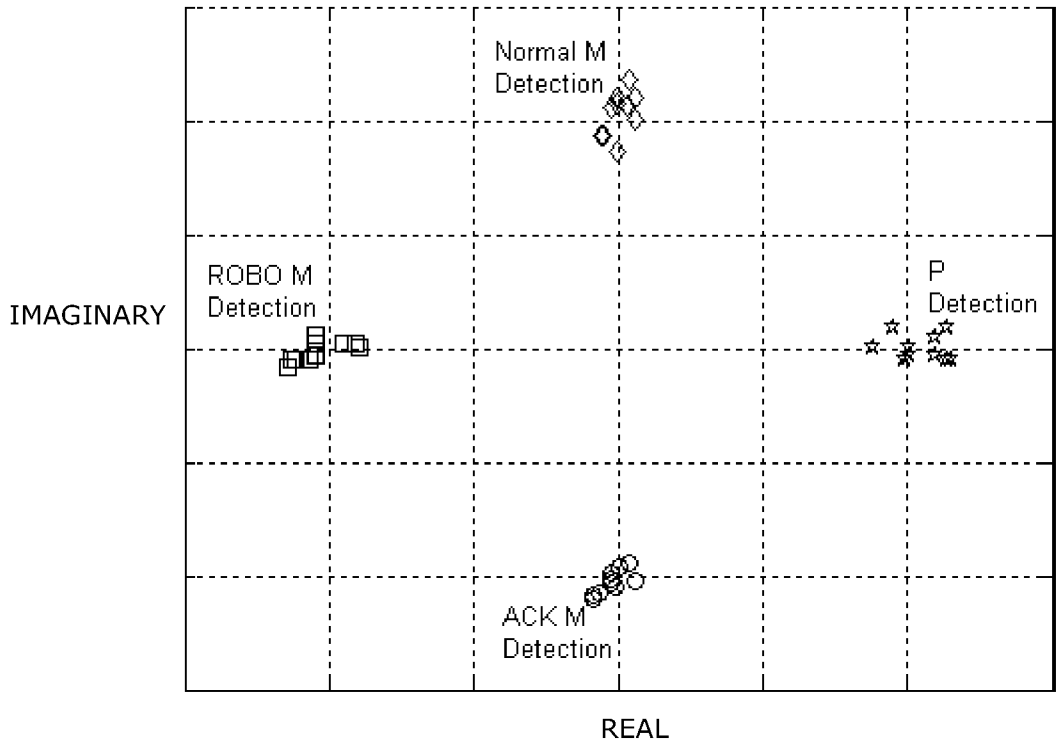

FIGURE 36

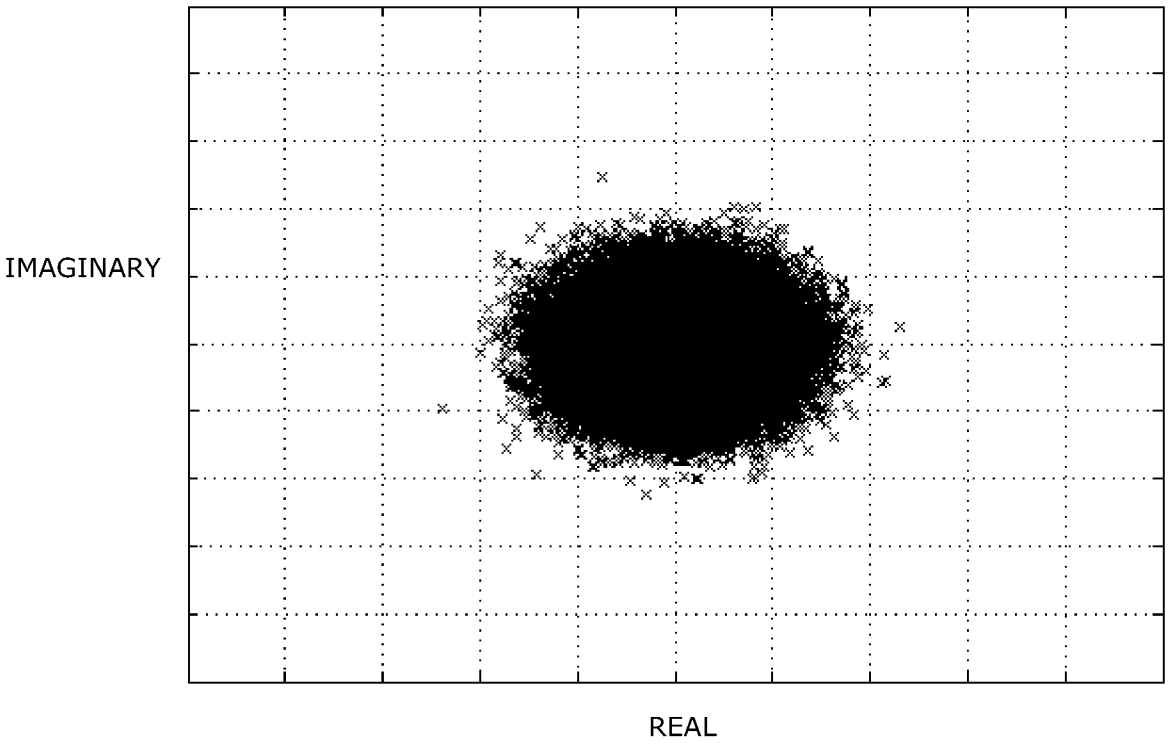

FIGURE 37

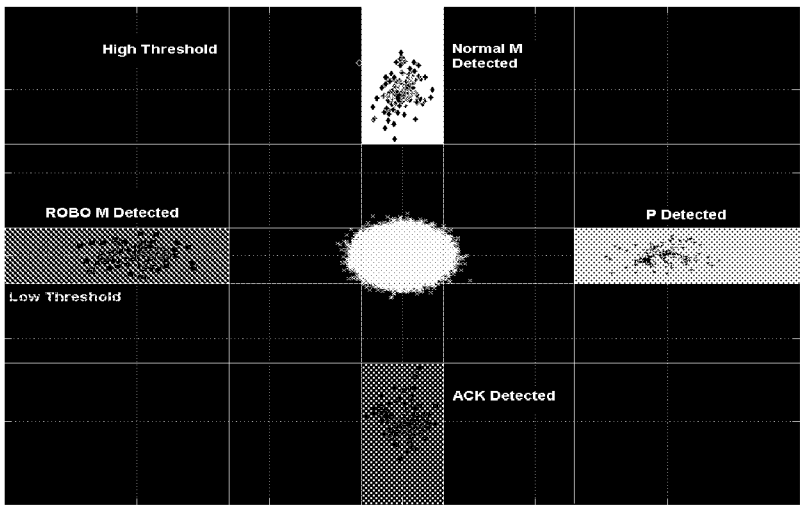

FIGURE 38

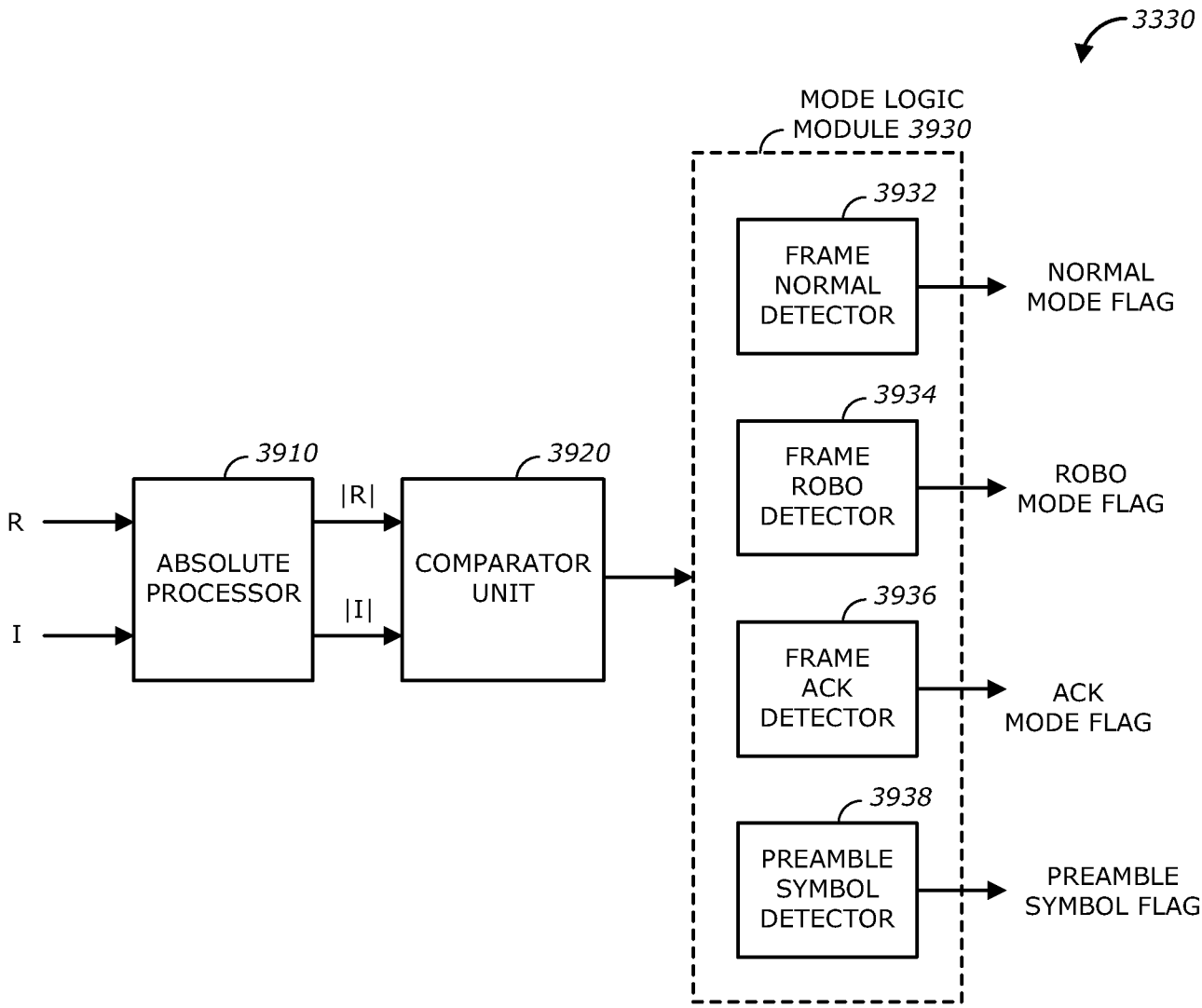

FIGURE 39