module telemetry;

reg [31:0] ASM;

reg [31:0] FH;

reg [31:0] FH1;

reg [31:0] FH2;

reg [31:0] FH3;

reg [31:0] FH4;

reg [31:0] FH5;

reg [31:0] FH6;

reg [31:0] FH7;

reg [31:0] FH8;

initial begin

$display("\nTELEMETRY\n");

// Telemetry Sync Marker ASM = 0x1ACFFC1D 32 bits

ASM = 32'b00110101011001111111111111111111;

// Minor Frame Headers (FFEEDD00 to FFEEDD07) 32 bits

FH = 32'b111111111101111011011110;

FH1 = 32'b11111111110111101101111000000000;

FH2 = 32'b11111111110111101101111000000001;

FH3 = 32'b11111111110111101101111000000010;

FH4 = 32'b11111111110111101101111000000011;

FH5 = 32'b11111111110111101101111000000100;

FH6 = 32'b11111111110111101101111000000101;

FH7 = 32'b11111111110111101101111000000110;

FH8 = 32'b11111111110111101101111000000111;

reg [5:0] x [0:1151];

reg [5:0] txCRC [0:1167];

reg [1167:0] crcCodeword [0:5];

reg [1151:0] rxMajorFrame [0:5];

reg [5:0] y [0:1151];

reg [5:0] rxCRC [0:1167];

reg [1167:0] rxCrcCodeword [0:5];

reg [5:0] crcCompare [0:0];

reg [5:0] match [0:0];

reg [5:0] mismatch [0:0];

reg [31:0] totalMatch;

reg [31:0] totalMismatch;

reg [31:0] cycleBER;

reg [5:0] xGyroRx [0:5][0:127];

reg [5:0] yGyroRx [0:5][0:127];

reg [5:0] zGyroRx [0:5][0:127];

reg [5:0] q1Rx [0:5][0:63];

reg [5:0] q2Rx [0:5][0:63];

reg [5:0] q3Rx [0:5][0:63];

reg [5:0] q4Rx [0:5][0:63];

reg [5:0] lastCmdRx [0:5][0:255];

integer tmNoOfErrors;

integer iiloop;

integer [0:5] xGyro [0:127];

integer [0:5] yGyro [0:127];

integer [0:5] zGyro [0:127];

integer [0:5] q1 [0:63];

integer [0:5] q2 [0:63];

integer [0:5] q3 [0:63];

integer [0:5] q4 [0:63];

integer [0:5] lastCmd [0:255];

integer [0:5] tmSuccessfullyReceivedBits [0:iiloop-1];

integer [0:5] tmUnsuccessfullyReceivedBits [0:iiloop-1];

integer [0:5] tmTransmittedbits [0:iiloop-1];

initial begin

tmNoOfErrors = 0;

iiloop = 100;

for (integer ii=0; ii<iiloop; ii=ii+1) begin

// Input measurands (randomly generated binary values)

for (integer j=0; j<128; j=j+1) begin

xGyro[j] = $random;

yGyro[j] = $random;

zGyro[j] = $random;

end

for (integer j=0; j<64; j=j+1) begin

q1[j] = $random;

q2[j] = $random;

q3[j] = $random;

q4[j] = $random;

end

for (integer j=0; j<256; j=j+1) begin

lastCmd[j] = $random;

end

tmSuccessfullyReceivedBits[ii] = 0;

tmUnsuccessfullyReceivedBits[ii] = 0;

tmTransmittedbits[ii] = 0;

end

end

//8 Minor Frames generated by selecting the input measurands

wire [15:0] minorFrame11 [FH1, xGyro[15:0], yGyro[15:0], zGyro[15:0], q1[15:0], q2[15:0], q3[15:0], q4[15:0]];

wire [63:0] minorFrame12 [FH2, xGyro[31:0], yGyro[31:0], zGyro[31:0], lastCmd[63:0]];

wire [31:0] minorFrame13 [FH3, xGyro[47:16], yGyro[47:16], zGyro[47:16], q1[31:16], q2[31:16], q3[31:16], q4[31:16]];

wire [79:0] minorFrame14 [FH4, xGyro[63:0], yGyro[63:0], zGyro[63:0], lastCmd[127:0]];

wire [47:0] minorFrame15 [FH5, xGyro[79:48], yGyro[79:48], zGyro[79:48], q1[47:32], q2[47:32], q3[47:32], q4[47:32]];

wire [95:0] minorFrame16 [FH6, xGyro[95:80], yGyro[95:80], zGyro[95:80], lastCmd[191:128]];

wire [63:0] minorFrame17 [FH7, xGyro[111:96], yGyro[111:96], zGyro[111:96], q1[63:48], q2[63:48], q3[63:48], q4[63:48]];

wire [143:0] minorFrame18 [FH8, xGyro[127:112], yGyro[127:112], zGyro[127:112], lastCmd[255:192]];

wire [15:0] FH1;

wire [15:0] FH2;

wire [15:0] FH3;

wire [15:0] FH4;

wire [15:0] FH5;

wire [15:0] FH6;

wire [15:0] FH7;

wire [15:0] FH8;

wire [15:0] xGyro [1:2];

wire [15:0] yGyro [1:2];

wire [15:0] zGyro [1:2];

wire [15:0] q1 [1:2];

wire [15:0] q2 [1:2];

wire [15:0] q3 [1:2];

wire [15:0] q4 [1:2];

wire [63:0] lastCmd [1:2];

wire [127:0] minorFrame21;

wire [127:0] minorFrame22;

wire [127:0] minorFrame23;

wire [127:0] minorFrame24;

wire [127:0] minorFrame25;

wire [127:0] minorFrame26;

wire [127:0] minorFrame27;

wire [127:0] minorFrame28;

assign minorFrame21 = {FH1, xGyro[2][15:0], yGyro[2][15:0], zGyro[2][15:0], q1[2][15:0], q2[2][15:0], q3[2][15:0], q4[2][15:0]};

assign minorFrame22 = {FH2, xGyro[2][31:16], yGyro[2][31:16], zGyro[2][31:16], lastCmd[2][63:0]};

assign minorFrame23 = {FH3, xGyro[2][47:32], yGyro[2][47:32], zGyro[2][47:32], q1[2][31:16], q2[2][31:16], q3[2][31:16], q4[2][31:16]};

assign minorFrame24 = {FH4, xGyro[2][63:48], yGyro[2][63:48], zGyro[2][63:48], lastCmd[2][127:64]};

assign minorFrame25 = {FH5, xGyro[2][79:64], yGyro[2][79:64], zGyro[2][79:64], q1[2][47:32], q2[2][47:32], q3[2][47:32], q4[2][47:32]};

assign minorFrame26 = {FH6, xGyro[2][95:80], yGyro[2][95:80], zGyro[2][95:80], lastCmd[2][191:128]};

assign minorFrame27 = {FH7, xGyro[2][111:96], yGyro[2][111:96], zGyro[2][111:96], q1[2][63:48], q2[2][63:48], q3[2][63:48], q4[2][63:48]};

assign minorFrame28 = {FH8, xGyro[2][127:112], yGyro[2][127:112], zGyro[2][127:112], lastCmd[2][255:192]};

wire [15:0] FH1;

wire [15:0] FH2;

wire [15:0] FH3;

wire [15:0] FH4;

wire [15:0] FH5;

wire [15:0] FH6;

wire [15:0] FH7;

wire [15:0] xGyro [3:0][0:127];

wire [15:0] yGyro [3:0][0:127];

wire [15:0] zGyro [3:0][0:127];

wire [15:0] q1 [3:0][0:127];

wire [15:0] q2 [3:0][0:127];

wire [15:0] q3 [3:0][0:127];

wire [15:0] q4 [3:0][0:127];

wire [15:0] lastCmd [3:0][0:255];

wire [127:0] minorFrame31;

wire [127:0] minorFrame32;

wire [127:0] minorFrame33;

wire [127:0] minorFrame34;

wire [127:0] minorFrame35;

wire [127:0] minorFrame36;

wire [127:0] minorFrame37;

assign minorFrame31 = {FH1, xGyro[3][0:15], yGyro[3][0:15], zGyro[3][0:15], q1[3][0:15], q2[3][0:15], q3[3][0:15], q4[3][0:15]};

assign minorFrame32 = {FH2, xGyro[3][16:31], yGyro[3][16:31], zGyro[3][16:31], lastCmd[3][0:63]};

assign minorFrame33 = {FH3, xGyro[3][32:47], yGyro[3][32:47], zGyro[3][32:47], q1[3][16:31], q2[3][16:31], q3[3][16:31], q4[3][16:31]};

assign minorFrame34 = {FH4, xGyro[3][48:63], yGyro[3][48:63], zGyro[3][48:63], lastCmd[3][64:127]};

assign minorFrame35 = {FH5, xGyro[3][64:79], yGyro[3][64:79], zGyro[3][64:79], q1[3][32:47], q2[3][32:47], q3[3][32:47], q4[3][32:47]};

assign minorFrame36 = {FH6, xGyro[3][80:95], yGyro[3][80:95], zGyro[3][80:95], lastCmd[3][128:191]};

assign minorFrame37 = {FH7, xGyro[3][96:111], yGyro[3][96:111], zGyro[3][96:111], q1[3][48:63], q2[3][48:63], q3[3][48:63], q4[3][48:63]};

wire [7:0] minorFrame38 [FH8, xGyro[113:128], yGyro[113:128], zGyro[113:128], lastCmd[193:256]];

wire [7:0] minorFrame41 [FH1, xGyro[1:16], yGyro[1:16], zGyro[1:16], q1[1:16], q2[1:16], q3[1:16], q4[1:16]];

wire [7:0] minorFrame42 [FH2, xGyro[17:32], yGyro[17:32], zGyro[17:32], lastCmd[1:64]];

wire [7:0] minorFrame43 [FH3, xGyro[33:48], yGyro[33:48], zGyro[33:48], q1[17:32], q2[17:32], q3[17:32], q4[17:32]];

wire [7:0] minorFrame44 [FH4, xGyro[49:64], yGyro[49:64], zGyro[49:64], lastCmd[65:128]];

wire [7:0] minorFrame45 [FH5, xGyro[65:80], yGyro[65:80], zGyro[65:80], q1[33:48], q2[33:48], q3[33:48], q4[33:48]];

wire [7:0] minorFrame46 [FH6, xGyro[81:96], yGyro[81:96], zGyro[81:96], lastCmd[129:192]];

wire [7:0] minorFrame47 [FH7, xGyro[97:112], yGyro[97:112], zGyro[97:112], q1[49:64], q2[49:64], q3[49:64], q4[49:64]];

wire [7:0] minorFrame48 [FH8, xGyro[113:128], yGyro[113:128], zGyro[113:128], lastCmd[193:256]];

wire [15:0] FH1;

wire [15:0] FH2;

wire [15:0] FH3;

wire [15:0] FH4;

wire [15:0] FH5;

wire [15:0] FH6;

wire [15:0] FH7;

wire [15:0] FH8;

wire [15:0] xGyro [0:4][0:127];

wire [15:0] yGyro [0:4][0:127];

wire [15:0] zGyro [0:4][0:127];

wire [15:0] q1 [0:4][0:127];

wire [15:0] q2 [0:4][0:127];

wire [15:0] q3 [0:4][0:127];

wire [15:0] q4 [0:4][0:127];

wire [15:0] lastCmd [0:4][0:255];

wire [127:0] minorFrame51;

wire [127:0] minorFrame52;

wire [127:0] minorFrame53;

wire [127:0] minorFrame54;

wire [127:0] minorFrame55;

wire [127:0] minorFrame56;

wire [127:0] minorFrame57;

wire [127:0] minorFrame58;

assign minorFrame51 = {FH1, xGyro[5][0:15], yGyro[5][0:15], zGyro[5][0:15], q1[5][0:15], q2[5][0:15], q3[5][0:15], q4[5][0:15]};

assign minorFrame52 = {FH2, xGyro[5][16:31], yGyro[5][16:31], zGyro[5][16:31], lastCmd[5][0:63]};

assign minorFrame53 = {FH3, xGyro[5][32:47], yGyro[5][32:47], zGyro[5][32:47], q1[5][16:31], q2[5][16:31], q3[5][16:31], q4[5][16:31]};

assign minorFrame54 = {FH4, xGyro[5][48:63], yGyro[5][48:63], zGyro[5][48:63], lastCmd[5][64:127]};

assign minorFrame55 = {FH5, xGyro[5][64:79], yGyro[5][64:79], zGyro[5][64:79], q1[5][32:47], q2[5][32:47], q3[5][32:47], q4[5][32:47]};

assign minorFrame56 = {FH6, xGyro[5][80:95], yGyro[5][80:95], zGyro[5][80:95], lastCmd[5][128:191]};

assign minorFrame57 = {FH7, xGyro[5][96:111], yGyro[5][96:111], zGyro[5][96:111], q1[5][48:63], q2[5][48:63], q3[5][48:63], q4[5][48:63]};

assign minorFrame58 = {FH8, xGyro[5][112:127], yGyro[5][112:127], zGyro[5][112:127], lastCmd[5][192:255]};

wire [15:0] FH1;

wire [15:0] FH2;

wire [15:0] FH3;

wire [15:0] FH4;

wire [15:0] FH5;

wire [15:0] FH6;

wire [15:0] FH7;

wire [15:0] FH8;

wire [15:0] xGyro [0:5][0:127];

wire [15:0] yGyro [0:5][0:127];

wire [15:0] zGyro [0:5][0:127];

wire [15:0] q1 [0:5][0:127];

wire [15:0] q2 [0:5][0:127];

wire [15:0] q3 [0:5][0:127];

wire [15:0] q4 [0:5][0:127];

wire [15:0] lastCmd [0:5][0:255];

wire [255:0] minorFrame61;

wire [255:0] minorFrame62;

wire [255:0] minorFrame63;

wire [255:0] minorFrame64;

wire [255:0] minorFrame65;

wire [255:0] minorFrame66;

wire [255:0] minorFrame67;

wire [255:0] minorFrame68;

assign minorFrame61 = {FH1, xGyro[5][0:15], yGyro[5][0:15], zGyro[5][0:15], q1[5][0:15], q2[5][0:15], q3[5][0:15], q4[5][0:15]};

assign minorFrame62 = {FH2, xGyro[5][16:31], yGyro[5][16:31], zGyro[5][16:31], lastCmd[5][0:63]};

assign minorFrame63 = {FH3, xGyro[5][32:47], yGyro[5][32:47], zGyro[5][32:47], q1[5][16:31], q2[5][16:31], q3[5][16:31], q4[5][16:31]};

assign minorFrame64 = {FH4, xGyro[5][48:63], yGyro[5][48:63], zGyro[5][48:63], lastCmd[5][64:127]};

assign minorFrame65 = {FH5, xGyro[5][64:79], yGyro[5][64:79], zGyro[5][64:79], q1[5][32:47], q2[5][32:47], q3[5][32:47], q4[5][32:47]};

assign minorFrame66 = {FH6, xGyro[5][80:95], yGyro[5][80:95], zGyro[5][80:95], lastCmd[5][128:191]};

assign minorFrame67 = {FH7, xGyro[5][96:111], yGyro[5][96:111], zGyro[5][96:111], q1[5][48:63], q2[5][48:63], q3[5][48:63], q4[5][48:63]};

assign minorFrame68 = {FH8, xGyro[5][112:127], yGyro[5][112:127], zGyro[5][112:127], lastCmd[5][192:255]};

// Major Frame = 8 minor frames

reg [7:0] MajorFrame1;

reg [7:0] MajorFrame2;

reg [7:0] MajorFrame3;

reg [7:0] MajorFrame4;

reg [7:0] MajorFrame5;

reg [7:0] MajorFrame6;

assign MajorFrame1 = {minorFrame11, minorFrame12, minorFrame13, minorFrame14, minorFrame15, minorFrame16, minorFrame17, minorFrame18};

assign MajorFrame2 = {minorFrame21, minorFrame22, minorFrame23, minorFrame24, minorFrame25, minorFrame26, minorFrame27, minorFrame28};

assign MajorFrame3 = {minorFrame31, minorFrame32, minorFrame33, minorFrame34, minorFrame35, minorFrame36, minorFrame37, minorFrame38};

assign MajorFrame4 = {minorFrame41, minorFrame42, minorFrame43, minorFrame44, minorFrame45, minorFrame46, minorFrame47, minorFrame48};

assign MajorFrame5 = {minorFrame51, minorFrame52, minorFrame53, minorFrame54, minorFrame55, minorFrame56, minorFrame57, minorFrame58};

assign MajorFrame6 = {minorFrame61, minorFrame62, minorFrame63, minorFrame64, minorFrame65, minorFrame66, minorFrame67, minorFrame68};

// CRC-16 Checksum generation for MajorFrame

reg [15:0] x [1:6];

reg [15:0] txCRC [1:6];

reg [15:0] crcCodeword [1:6];

x[1] = MajorFrame1; // converting row vector to column vector for the purpose of CRC generation

x[2] = MajorFrame2;

x[3] = MajorFrame3;

x[4] = MajorFrame4;

x[5] = MajorFrame5;

x[6] = MajorFrame6;

reg [15:0] poly = 16'b10001000000100001;

reg [15:0] crcgenerator;

reg [15:0] crcdetector;

always @\* begin

crcgenerator = x[1];

txCRC[1] = crcgenerator;

crcCodeword[1] = txCRC[1];

end

always @\* begin

crcgenerator = x[2];

txCRC[2] = crcgenerator;

crcCodeword[2] = txCRC[2];

end

always @\* begin

crcgenerator = x[3];

txCRC[3] = crcgenerator;

crcCodeword[3] = txCRC[3];

end

always @\* begin

crcgenerator = x[4];

txCRC[4] = crcgenerator;

crcCodeword[4] = txCRC[4];

end

always @\* begin

crcgenerator = x[5];

txCRC[5] = crcgenerator;

crcCodeword[5] = txCRC[5];

end

always @\* begin

crcgenerator = x[6];

txCRC[6] = crcgenerator;

crcCodeword[6] = txCRC[6];

end

//Telemetry Frame

reg [7:0] ASM = 8'b10101010;

reg [7:0] crcCodeword [6:0];

reg [7:0] TF [95:0];

integer sizeTF;

// Convolution Encoding

// CADU obtained by (2,1,7) convolution coding of Telemetry Transfer Frame

reg [7:0] convEncIN [95:0];

reg [7:0] convEncOUT [191:0];

reg [7:0] CADU [191:0];

reg [7:0] txData [191:0];

// QPSK modulation - demodulation through AWGN channel

reg [1:0] txDataBits [383:0];

reg [1:0] rxDataBits [383:0];

reg [1:0] modSigBits [383:0];

reg [1:0] rxSigBits [383:0];

reg [1:0] demodSigBits [383:0];

reg [1:0] errorStats [1:0];

integer tmNoOfErrors;

// Telemetry Frame

assign TF = {ASM, crcCodeword[1], ASM, crcCodeword[2], ASM, crcCodeword[3], ASM, crcCodeword[4], ASM, crcCodeword[5], ASM, crcCodeword[6]};

assign sizeTF = 96;

// Convolution Encoding

// CADU obtained by (2,1,7) convolution coding of Telemetry Transfer Frame

always @\* begin

convEncIN = TF;

end

always @(posedge clk) begin

convEncOUT = convenc(convEncIN, 7'b1111001);

CADU = convEncOUT;

txData = convEncOUT;

end

// QPSK modulation - demodulation through AWGN channel

always @(posedge clk) begin

modSigBits = qpskModulator(txDataBits);

rxSigBits = modSigBits;

demodSigBits = qpskDemodulator(rxSigBits);

errorStats = errorRate(txDataBits, demodSigBits);

rxDataBits = demodSigBits;

tmNoOfErrors = tmNoOfErrors + errorStats[1];

end

//Viterbi Decoding

wire [N-1:0] vitDecIN;

assign vitDecIN = rxData; // converting row vector to column vector for the purpose of viterbi decoding

reg [M-1:0] vitDecOUT;

integer tb = 35; // traceback depth of 35 -> approx. 5 times constraint length

always @(posedge clk) begin

if (reset) begin

vitDecOUT <= '0;

end else begin

vitDecOUT <= vitdec(vitDecIN, trellis, tb, 'trunc', 'hard'); // vit decoding after noisy transmission

// vitDecOUT <= vitdec(txData, trellis, tb, 'trunc', 'hard'); // txData used in place of vitDecIN and hard in place of unquant to override mod-demod and noise channel

end

end

wire [N-1:0] decodedTF;

assign decodedTF = vitDecOUT; // converting column vector back to row vector

wire [31:0] sizeDecodedTF;

assign sizeDecodedTF = decodedTF;

//Synchronizing with ASM

reg [31:0] asmIndex;

reg [31:0] majorFrame1Start;

reg [31:0] majorFrame2Start;

reg [31:0] majorFrame3Start;

reg [31:0] majorFrame4Start;

reg [31:0] majorFrame5Start;

reg [31:0] majorFrame6Start;

asmIndex = strstr(decodedTF, ASM);

// Segregating data

majorFrame1Start = asmIndex + strlen(ASM);

majorFrame2Start = asmIndex + strlen(ASM);

majorFrame3Start = asmIndex + strlen(ASM);

majorFrame4Start = asmIndex + strlen(ASM);

majorFrame5Start = asmIndex + strlen(ASM);

majorFrame6Start = asmIndex + strlen(ASM);

// rxMajorFrame is sans CRC and ASM

rxMajorFrame[0] = decodedTF[majorFrame1Start:(majorFrame2Start-49)];

rxMajorFrame[1] = decodedTF[majorFrame2Start:(majorFrame3Start-49)];

rxMajorFrame[2] = decodedTF[majorFrame3Start:(majorFrame4Start-49)];

rxMajorFrame[3] = decodedTF[majorFrame4Start:(majorFrame5Start-49)];

rxMajorFrame[4] = decodedTF[majorFrame5Start:(majorFrame6Start-49)];

rxMajorFrame[5] = decodedTF[majorFrame6Start:(decodedTF.length-16)];

assign y[0] = rxMajorFrame[0]; // converting row vector to column vector for CRC computation

assign y[1] = rxMajorFrame[1];

assign y[2] = rxMajorFrame[2];

assign y[3] = rxMajorFrame[3];

assign y[4] = rxMajorFrame[4];

assign y[5] = rxMajorFrame[5];

assign rxCRC[0] = crcgenerator(y[0]); // rxCRC is the calculatedCRC appended mainFrame as column vector

assign rxCRC[1] = crcgenerator(y[1]);

assign rxCRC[2] = crcgenerator(y[2]);

assign rxCRC[3] = crcgenerator(y[3]);

assign rxCRC[4] = crcgenerator(y[4]);

assign rxCRC[5] = crcgenerator(y[5]);

rxCrcCodeword[0] = rxCRC[0]; // converting column vector back to row vector. rxCrcCodeword is the calculatedCRC appended mainFrame

rxCrcCodeword[1] = rxCRC[1];

rxCrcCodeword[2] = rxCRC[2];

rxCrcCodeword[3] = rxCRC[3];

rxCrcCodeword[4] = rxCRC[4];

rxCrcCodeword[5] = rxCRC[5];

crcCompare[0] = (rxCrcCodeword[0] == decodedTF[majorFrame1Start:(majorFrame2Start-33)]);

crcCompare[1] = (rxCrcCodeword[1] == decodedTF[majorFrame2Start:(majorFrame3Start-33)]);

crcCompare[2] = (rxCrcCodeword[2] == decodedTF[majorFrame3Start:(majorFrame4Start-33)]);

crcCompare[3] = (rxCrcCodeword[3] == decodedTF[majorFrame4Start:(majorFrame5Start-33)]);

crcCompare[4] = (rxCrcCodeword[4] == decodedTF[majorFrame5Start:(majorFrame6Start-33)]);

crcCompare[5] = (rxCrcCodeword[5] == decodedTF[majorFrame6Start:end]);

//rxMinorFrame = zeros(1,144)

if (crcCompare[0] == 1) begin

match[0] = match[0] + 1;

//fprintf('calculated CRC matches received CRC in Frame1\n') ;

fhIndex1 = strfind(rxMajorFrame[0], FH);

rxMinorFrame11 = rxMajorFrame[0][(fhIndex1[0]:(fhIndex1[1]-1))];

rxMinorFrame12 = rxMajorFrame[0][(fhIndex1[1]:(fhIndex1[2]-1))];

rxMinorFrame13 = rxMajorFrame[0][(fhIndex1[2]:(fhIndex1[3]-1))];

rxMinorFrame14 = rxMajorFrame[0][(fhIndex1[3]:(fhIndex1[4]-1))];

rxMinorFrame15 = rxMajorFrame[0][(fhIndex1[4]:(fhIndex1[5]-1))];

rxMinorFrame16 = rxMajorFrame[0][(fhIndex1[5]:(fhIndex1[6]-1))];

rxMinorFrame17 = rxMajorFrame[0][(fhIndex1[6]:(fhIndex1[7]-1))];

rxMinorFrame18 = rxMajorFrame[0][(fhIndex1[7]):end];

end

else begin

mismatch[0] = mismatch[0] + 1;

$display("CRC mismatch in Frame1");

end

if (crcCompare[1] == 1) begin

match[2] = match[2] + 1;

// $display("calculated CRC matches received CRC in Frame2");

fhIndex2 = rxMajorFrame[2].find(FH);

rxMinorFrame21 = rxMajorFrame[2][fhIndex2[1]:fhIndex2[2]-1];

rxMinorFrame22 = rxMajorFrame[2][fhIndex2[2]:fhIndex2[3]-1];

rxMinorFrame23 = rxMajorFrame[2][fhIndex2[3]:fhIndex2[4]-1];

rxMinorFrame24 = rxMajorFrame[2][fhIndex2[4]:fhIndex2[5]-1];

rxMinorFrame25 = rxMajorFrame[2][fhIndex2[5]:fhIndex2[6]-1];

rxMinorFrame26 = rxMajorFrame[2][fhIndex2[6]:fhIndex2[7]-1];

rxMinorFrame27 = rxMajorFrame[2][fhIndex2[7]:fhIndex2[8]-1];

rxMinorFrame28 = rxMajorFrame[2][fhIndex2[8]:$size(rxMajorFrame[2])-1];

end

else begin

mismatch[2] = mismatch[2] + 1;

$display("CRC mismatch in Frame2");

end

if (crcCompare[0][2] == 1) begin

match[2] = match[2] + 1;

// $display("calculated CRC matches received CRC in Frame3");

fhIndex3 = rxMajorFrame[2].find(FH);

rxMinorFrame31 = rxMajorFrame[2][fhIndex3[0]:fhIndex3[1]-1];

rxMinorFrame32 = rxMajorFrame[2][fhIndex3[1]:fhIndex3[2]-1];

rxMinorFrame33 = rxMajorFrame[2][fhIndex3[2]:fhIndex3[3]-1];

rxMinorFrame34 = rxMajorFrame[2][fhIndex3[3]:fhIndex3[4]-1];

rxMinorFrame35 = rxMajorFrame[2][fhIndex3[4]:fhIndex3[5]-1];

rxMinorFrame36 = rxMajorFrame[2][fhIndex3[5]:fhIndex3[6]-1];

rxMinorFrame37 = rxMajorFrame[2][fhIndex3[6]:fhIndex3[7]-1];

rxMinorFrame38 = rxMajorFrame[2][fhIndex3[7]:$size(rxMajorFrame[2])-1];

end

else begin

mismatch[2] = mismatch[2] + 1;

$display("CRC mismatch in Frame3");

end

if (crcCompare[0][3] == 1) begin

match[3] = match[3] + 1;

// $display("calculated CRC matches received CRC in Frame4");

fhIndex4 = rxMajorFrame[3].find(FH);

rxMinorFrame41 = rxMajorFrame[3][fhIndex4[0]:fhIndex4[1]-1];

rxMinorFrame42 = rxMajorFrame[3][fhIndex4[1]:fhIndex4[2]-1];

rxMinorFrame43 = rxMajorFrame[3][fhIndex4[2]:fhIndex4[3]-1];

rxMinorFrame44 = rxMajorFrame[3][fhIndex4[3]:fhIndex4[4]-1];

rxMinorFrame45 = rxMajorFrame[3][fhIndex4[4]:fhIndex4[5]-1];

rxMinorFrame46 = rxMajorFrame[3][fhIndex4[5]:fhIndex4[6]-1];

rxMinorFrame47 = rxMajorFrame[3][fhIndex4[6]:fhIndex4[7]-1];

rxMinorFrame48 = rxMajorFrame[3][fhIndex4[7]:$size(rxMajorFrame[3])-1];

end

else begin

mismatch[3] = mismatch[3] + 1;

$display("CRC mismatch in Frame4");

end

if (crcCompare[0][4] == 1) begin

match[4] = match[4] + 1;

// $display("calculated CRC matches received CRC in Frame5");

fhIndex5 = rxMajorFrame[4].find(FH);

rxMinorFrame51 = rxMajorFrame[4][fhIndex5[0]:fhIndex5[1]-1];

rxMinorFrame52 = rxMajorFrame[4][fhIndex5[1]:fhIndex5[2]-1];

rxMinorFrame53 = rxMajorFrame[4][fhIndex5[2]:fhIndex5[3]-1];

rxMinorFrame54 = rxMajorFrame[4][fhIndex5[3]:fhIndex5[4]-1];

rxMinorFrame55 = rxMajorFrame[4][fhIndex5[4]:fhIndex5[5]-1];

rxMinorFrame56 = rxMajorFrame[4][fhIndex5[5]:fhIndex5[6]-1];

rxMinorFrame57 = rxMajorFrame[4][fhIndex5[6]:fhIndex5[7]-1];

rxMinorFrame58 = rxMajorFrame[4][fhIndex5[7]:$size(rxMajorFrame[4])-1];

end

else begin

mismatch[4] = mismatch[4] + 1;

$display("CRC mismatch in Frame5");

end

if (crcCompare[0][5] == 1) begin

match[5] = match[5] + 1;

// $display("calculated CRC matches received CRC in Frame6");

fhIndex6 = rxMajorFrame[5].find(FH);

rxMinorFrame61 = rxMajorFrame[5][fhIndex6[0]:fhIndex6[1]-1];

rxMinorFrame62 = rxMajorFrame[5][fhIndex6[1]:fhIndex6[2]-1];

rxMinorFrame63 = rxMajorFrame[5][fhIndex6[2]:fhIndex6[3]-1];

rxMinorFrame64 = rxMajorFrame[5][fhIndex6[3]:fhIndex6[4]-1];

rxMinorFrame65 = rxMajorFrame[5][fhIndex6[4]:fhIndex6[5]-1];

rxMinorFrame66 = rxMajorFrame[5][fhIndex6[5]:fhIndex6[6]-1];

rxMinorFrame67 = rxMajorFrame[5][fhIndex6[6]:fhIndex6[7]-1];

rxMinorFrame68 = rxMajorFrame[5][fhIndex6[7]:$size(rxMajorFrame[5])-1];

end

else begin

mismatch[5] = mismatch[5] + 1;

$display("CRC mismatch in Frame6");

end

assign matchCount = match[0] + match[1] + match[2] + match[3] + match[4] + match[5] + match[6] + match[7];

assign mismatchCount = mismatch[0] + mismatch[1] + mismatch[2] + mismatch[3] + mismatch[4] + mismatch[5] + mismatch[6] + mismatch[7];

assign xGyroRx = {rxMinorFrame11[48:33], rxMinorFrame12[48:33], rxMinorFrame13[48:33], rxMinorFrame14[48:33], rxMinorFrame15[48:33], rxMinorFrame16[48:33], rxMinorFrame17[48:33], rxMinorFrame18[48:33]};

assign yGyroRx = {rxMinorFrame11[64:49], rxMinorFrame12[64:49], rxMinorFrame13[64:49], rxMinorFrame14[64:49], rxMinorFrame15[64:49], rxMinorFrame16[64:49], rxMinorFrame17[64:49], rxMinorFrame18[64:49]};

assign zGyroRx = {rxMinorFrame11[80:65], rxMinorFrame12[80:65], rxMinorFrame13[80:65], rxMinorFrame14[80:65], rxMinorFrame15[80:65], rxMinorFrame16[80:65], rxMinorFrame17[80:65], rxMinorFrame18[80:65]};

assign q1Rx = {rxMinorFrame11[96:81], rxMinorFrame13[96:81], rxMinorFrame15[96:81], rxMinorFrame17[96:81]};

assign q2Rx = {rxMinorFrame11[112:97], rxMinorFrame13[112:97], rxMinorFrame15[112:97], rxMinorFrame17[112:97]};

assign q3Rx = {rxMinorFrame11[128:113], rxMinorFrame13[128:113], rxMinorFrame15[128:113], rxMinorFrame17[128:113]};

assign q4Rx = {rxMinorFrame11[144:129], rxMinorFrame13[144:129], rxMinorFrame15[144:129], rxMinorFrame17[144:129]};

assign lastCmdRx = {rxMinorFrame12[144:81], rxMinorFrame14[144:81], rxMinorFrame16[144:81], rxMinorFrame18[144:81]};

xGyroRx[2] = {rxMinorFrame21[48:33], rxMinorFrame22[48:33], rxMinorFrame23[48:33], rxMinorFrame24[48:33], rxMinorFrame25[48:33], rxMinorFrame26[48:33], rxMinorFrame27[48:33], rxMinorFrame28[48:33]};

yGyroRx[2] = {rxMinorFrame21[64:49], rxMinorFrame22[64:49], rxMinorFrame23[64:49], rxMinorFrame24[64:49], rxMinorFrame25[64:49], rxMinorFrame26[64:49], rxMinorFrame27[64:49], rxMinorFrame28[64:49]};

zGyroRx[2] = {rxMinorFrame21[80:65], rxMinorFrame22[80:65], rxMinorFrame23[80:65], rxMinorFrame24[80:65], rxMinorFrame25[80:65], rxMinorFrame26[80:65], rxMinorFrame27[80:65], rxMinorFrame28[80:65]};

q1Rx[2] = {rxMinorFrame21[96:81], rxMinorFrame23[96:81], rxMinorFrame25[96:81], rxMinorFrame27[96:81]};

q2Rx[2] = {rxMinorFrame21[112:97], rxMinorFrame23[112:97], rxMinorFrame25[112:97], rxMinorFrame27[112:97]};

q3Rx[2] = {rxMinorFrame21[128:113], rxMinorFrame23[128:113], rxMinorFrame25[128:113], rxMinorFrame27[128:113]};

q4Rx[2] = {rxMinorFrame21[144:129], rxMinorFrame23[144:129], rxMinorFrame25[144:129], rxMinorFrame27[144:129]};

lastCmdRx[2] = {rxMinorFrame22[144:81], rxMinorFrame24[144:81], rxMinorFrame26[144:81], rxMinorFrame28[144:81]};

xGyroRx[2:0] = {rxMinorFrame31[48:33], rxMinorFrame32[48:33], rxMinorFrame33[48:33], rxMinorFrame34[48:33], rxMinorFrame35[48:33], rxMinorFrame36[48:33], rxMinorFrame37[48:33], rxMinorFrame38[48:33]};

yGyroRx[2:0] = {rxMinorFrame31[64:49], rxMinorFrame32[64:49], rxMinorFrame33[64:49], rxMinorFrame34[64:49], rxMinorFrame35[64:49], rxMinorFrame36[64:49], rxMinorFrame37[64:49], rxMinorFrame38[64:49]};

zGyroRx[2:0] = {rxMinorFrame31[80:65], rxMinorFrame32[80:65], rxMinorFrame33[80:65], rxMinorFrame34[80:65], rxMinorFrame35[80:65], rxMinorFrame36[80:65], rxMinorFrame37[80:65], rxMinorFrame38[80:65]};

q1Rx[2:0] = {rxMinorFrame31[96:81], rxMinorFrame33[96:81], rxMinorFrame35[96:81], rxMinorFrame37[96:81]};

q2Rx[2:0] = {rxMinorFrame31[112:97], rxMinorFrame33[112:97], rxMinorFrame35[112:97], rxMinorFrame37[112:97]};

q3Rx[2:0] = {rxMinorFrame31[128:113], rxMinorFrame33[128:113], rxMinorFrame35[128:113], rxMinorFrame37[128:113]};

q4Rx[2:0] = {rxMinorFrame31[144:129], rxMinorFrame33[144:129], rxMinorFrame35[144:129], rxMinorFrame37[144:129]};

lastCmdRx[2:0] = {rxMinorFrame32[144:81], rxMinorFrame34[144:81], rxMinorFrame36[144:81], rxMinorFrame38[144:81]};

assign xGyroRx[3:0] = {rxMinorFrame41[48:33], rxMinorFrame42[48:33], rxMinorFrame43[48:33], rxMinorFrame44[48:33], rxMinorFrame45[48:33], rxMinorFrame46[48:33], rxMinorFrame47[48:33], rxMinorFrame48[48:33]};

assign yGyroRx[3:0] = {rxMinorFrame41[64:49], rxMinorFrame42[64:49], rxMinorFrame43[64:49], rxMinorFrame44[64:49], rxMinorFrame45[64:49], rxMinorFrame46[64:49], rxMinorFrame47[64:49], rxMinorFrame48[64:49]};

assign zGyroRx[3:0] = {rxMinorFrame41[80:65], rxMinorFrame42[80:65], rxMinorFrame43[80:65], rxMinorFrame44[80:65], rxMinorFrame45[80:65], rxMinorFrame46[80:65], rxMinorFrame47[80:65], rxMinorFrame48[80:65]};

assign q1Rx[3:0] = {rxMinorFrame41[96:81], rxMinorFrame43[96:81], rxMinorFrame45[96:81], rxMinorFrame47[96:81]};

assign q2Rx[3:0] = {rxMinorFrame41[112:97], rxMinorFrame43[112:97], rxMinorFrame45[112:97], rxMinorFrame47[112:97]};

assign q3Rx[3:0] = {rxMinorFrame41[128:113], rxMinorFrame43[128:113], rxMinorFrame45[128:113], rxMinorFrame47[128:113]};

assign q4Rx[3:0] = {rxMinorFrame41[144:129], rxMinorFrame43[144:129], rxMinorFrame45[144:129], rxMinorFrame47[144:129]};

assign lastCmdRx[3:0] = {rxMinorFrame42[144:81], rxMinorFrame44[144:81], rxMinorFrame46[144:81], rxMinorFrame48[144:81]};

xGyroRx[5] = {rxMinorFrame51[48:33], rxMinorFrame52[48:33], rxMinorFrame53[48:33], rxMinorFrame54[48:33], rxMinorFrame55[48:33], rxMinorFrame56[48:33], rxMinorFrame57[48:33], rxMinorFrame58[48:33]};

yGyroRx[5] = {rxMinorFrame51[64:49], rxMinorFrame52[64:49], rxMinorFrame53[64:49], rxMinorFrame54[64:49], rxMinorFrame55[64:49], rxMinorFrame56[64:49], rxMinorFrame57[64:49], rxMinorFrame58[64:49]};

zGyroRx[5] = {rxMinorFrame51[80:65], rxMinorFrame52[80:65], rxMinorFrame53[80:65], rxMinorFrame54[80:65], rxMinorFrame55[80:65], rxMinorFrame56[80:65], rxMinorFrame57[80:65], rxMinorFrame58[80:65]};

q1Rx[5] = {rxMinorFrame51[96:81], rxMinorFrame53[96:81], rxMinorFrame55[96:81], rxMinorFrame57[96:81]};

q2Rx[5] = {rxMinorFrame51[112:97], rxMinorFrame53[112:97], rxMinorFrame55[112:97], rxMinorFrame57[112:97]};

q3Rx[5] = {rxMinorFrame51[128:113], rxMinorFrame53[128:113], rxMinorFrame55[128:113], rxMinorFrame57[128:113]};

q4Rx[5] = {rxMinorFrame51[144:129], rxMinorFrame53[144:129], rxMinorFrame55[144:129], rxMinorFrame57[144:129]};

lastCmdRx[5] = {rxMinorFrame52[144:81], rxMinorFrame54[144:81], rxMinorFrame56[144:81], rxMinorFrame58[144:81]};

xGyroRx[6] = {rxMinorFrame61[48:33], rxMinorFrame62[48:33], rxMinorFrame63[48:33], rxMinorFrame64[48:33], rxMinorFrame65[48:33], rxMinorFrame66[48:33], rxMinorFrame67[48:33], rxMinorFrame68[48:33]};

yGyroRx[6] = {rxMinorFrame61[64:49], rxMinorFrame62[64:49], rxMinorFrame63[64:49], rxMinorFrame64[64:49], rxMinorFrame65[64:49], rxMinorFrame66[64:49], rxMinorFrame67[64:49], rxMinorFrame68[64:49]};

zGyroRx[6] = {rxMinorFrame61[80:65], rxMinorFrame62[80:65], rxMinorFrame63[80:65], rxMinorFrame64[80:65], rxMinorFrame65[80:65], rxMinorFrame66[80:65], rxMinorFrame67[80:65], rxMinorFrame68[80:65]};

q1Rx[6] = {rxMinorFrame61[96:81], rxMinorFrame63[96:81], rxMinorFrame65[96:81], rxMinorFrame67[96:81]};

q2Rx[6] = {rxMinorFrame61[112:97], rxMinorFrame63[112:97], rxMinorFrame65[112:97], rxMinorFrame67[112:97]};

q3Rx[6] = {rxMinorFrame61[128:113], rxMinorFrame63[128:113], rxMinorFrame65[128:113], rxMinorFrame67[128:113]};

q4Rx[6] = {rxMinorFrame61[144:129], rxMinorFrame63[144:129], rxMinorFrame65[144:129], rxMinorFrame67[144:129]};

lastCmdRx[6] = {rxMinorFrame62[144:81], rxMinorFrame64[144:81], rxMinorFrame66[144:81], rxMinorFrame68[144:81]};

totalMatch = matchCount;

totalMismatch = mismatchCount;

tmSuccessfullyReceivedBits[ii] = totalMatch\*150;

tmUnsuccessfullyReceivedBits[ii] = totalMismatch\*150;

tmTransmittedbits[ii] = 900\*ii; // 900bytes or 7200bits

cycleBER = cycleBER + $countones(TF ^ decodedTF)/7200;

assign BER = cycleBER / iiloop;

module Telemetry;

initial begin

$display("Telemetry");

$display("Transmitted Frame size(bytes) \t Successfully received Payload size (bytes)");

for (int i = 0; i <= (900\*ii); i = i + (900\*ii/10)) begin

$display("%d \t\t\t\t %d", tmTransmittedbits, tmSuccessfullyReceivedBits);

end

$display("BER %f - SNR %ddB", BER, SNR);

end

endmodule