7
Implementation of Modulation classifier over software defined radio

7.1 INTRODUCTION

Blind signal modulation recognition is an essential block for designing a cognitive radio. Different algorithms are developed in the literature, but few are given with detailed implementation. This chapter contains a detailed implementation of software-defined radio (SDR) based blind signal modulation recognizer (BSMR) on a field-programmable gate array (FPGA), which works without any prior knowledge of the received signal. The algorithm estimates all the communication signal parameters and correct offsets required for extracting the constellation points. The system developed classifies different orders of ASK, PSK, and QAM. The proposed algorithm is deployed on FPGA, using LabVIEW, for a reliable and reconfigurable platform. The algorithm is optimized to use minimum hardware resources and facilitate future up-gradation. The system developed by implementing the algorithm on the NI-FlexRIO-7975 FPGA module with the NI-5791 adapter detects modulation type in real-time without any training. Signals for testing are generated using NI-PXIe-5673 (RF transmitter), and BSMR identifies the modulation type in 81.451 msec under the AWGN channel.

BSMR recognizes the modulation in two stages. First is preprocessing of the received RF signal, in which different parameters like carrier frequency, symbol rate, channel state information (CSI), and symbol timing offset are estimated and corrected for offsets. The second stage includes methods to classify among different modulation formats.

The commercially available blind signal modulation detection system is a black box with no scope of further modification. Many papers with FPGA implementation based on wavelet [Digdarsini *et al.*, 2014], ANN [Keshk *et al.*, 2015], and conventional features [Iglesias *et al.*, 2015; Cutno and Cheng, 2017] are given in the literature, but only a few have been designed for real-time operation and provides little information about implementation. With the evolution of SDR, the implementation of a radio receiver can be easily achieved through LabVIEW coding. This paper includes a detailed description of the implementation of the algorithm on LabVIEW FPGA, so that it can be replicated and further worked upon the improvement over results concerning latency, complexity, and real-time performance. Most of the existing methods consider the specific order of modulation and depend on the offline training of the system. This makes the task complex and the system is not usable for other than the considered order of modulation. Unlike these methods, the present system can work for any order of modulation among ASK, PSK, and QAM with a slight modification. Hence SDR based on the FPGA hardware platform is chosen, as they are highly flexible, versatile, reconfigurable, and can be modified by making required changes in the software. Different modules of the algorithm are designed using the LabVIEW FPGA module of National Instruments (NI).

7.2 LABVIEW FLEXRIO SOFTWARE COMMUNICATION ARCHITECTURE

Hardware implementation of the algorithm over FPGA is done using LabVIEW virtual instruments (*VIs*). LabVIEW is a visual programming language, and its program subroutines are

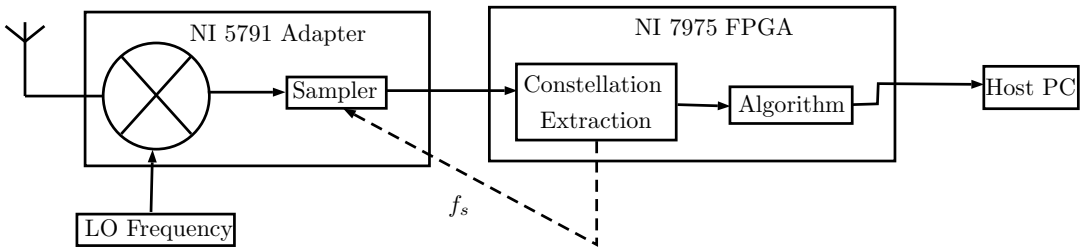


Figure 7.1 : System model

termed as virtual instruments (*VIs*). Two *VIs* are used for implementing the algorithm on LabVIEW FPGA (a) *Host VI* (b) *FPGA VI*. *Host VI* runs on the computer. It is used to send information between the host computer and the FPGA target, controls the FPGA, and provides a user interface to the system. It interacts with *FPGA VI* and displays the results.

Another *VI* is the *FPGA VI* (*targetVI*). To transfer small and frequent data between *Host VI* and *FPGA VI*, front panel communication is used. To transfer large data, first in first out (FIFO) is used. Another FPGA element used for storing large data is block memory, which can perform both read and write operations from random addresses. FPGA module compiles target VI to hardware. The graphical code is translated to VHDL. Further, the Xilinx compiler synthesizes this code into the hardware realization of LabVIEW design. Finally, a bitfile is generated that contains programming instructions that are mapped onto the FPGA target. Any change in the algorithm can be done by making the necessary changes on the *targetVI* and generating a new bitfile and can be reconfigured using the host. *VIs* created with the LabVIEW FPGA module can be targeted to FPGAs on RIO hardware, which is NI PXIe-7975R (Kintex-7 XC7K410T) FPGA based FlexRIO module in the system used. The system model is shown in figure 7.1. Basic elements used in different modules of FPGA programming, which are specific to the LabVIEW FPGA module, are explained in the next section.

7.3 FPGA ELEMENTS USED

In this section, different elements that are specific to the LabVIEW FPGA module are discussed. For storing data or read/write data, memory elements are used. Other than *memory* elements, loops named *Flat sequence* and *Single Cycled Timed Loop (SCTL)* specific to LabVIEW are used. Figure 7.2 shows the LabVIEW project window and how different $FPGA\ I/O$, $FIFO$, $Memory$, etc., are defined.

7.3.1 First In First Out (FIFO)

FIFO is a common buffering technique used with RIO devices. It is a fixed-length queue in which data can be written to and read from memory. Figure 7.3 shows different FIFOs in read and write mode. Three types of FIFOs are there in the LabVIEW FPGA module:

1. **Target Scoped FIFO**: This FIFO transfers data to and from different sections of code in multiple VIs on the same target with a single FIFO.

2. **VI-defined**: It is used to transfer data in multiple loops of the same VI.

3. **Direct Memory Access (DMA) FIFO**: To transfer data from $host-VI$ to $target-VI$ and vice versa, DMA FIFOs are used.

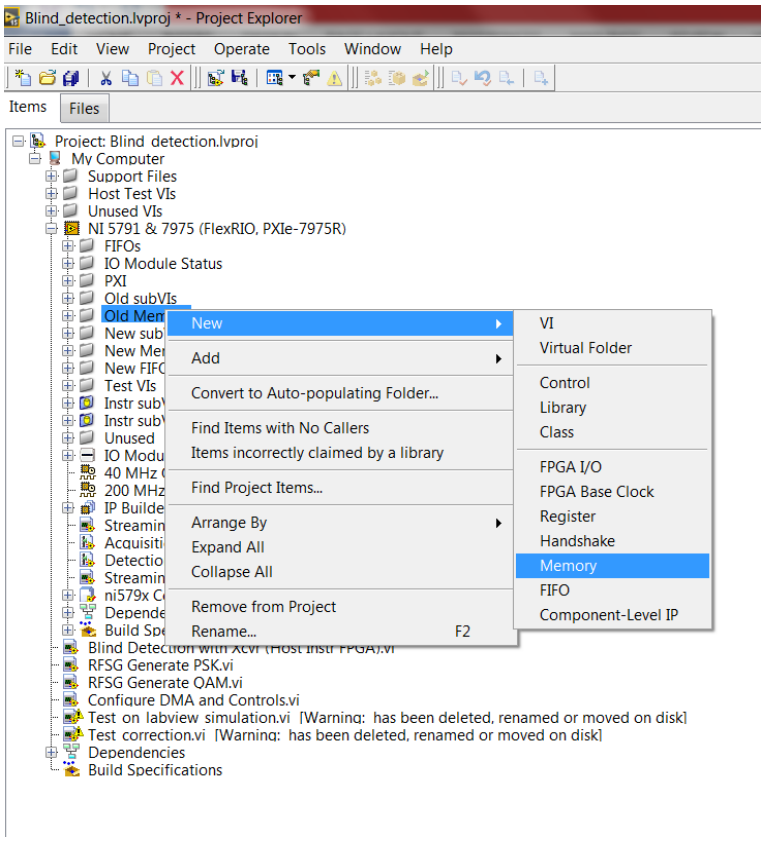


Figure 7.2 : Project explorer window for defining memory

7.3.2 Memory elements

The primary means of data storage in a single clock domain using an FPGA application is a memory item. The LabVIEW FPGA Module has two types of memory items:

1. **VI-defined memory items**: Use VI-defined memory items along with Memory name controls to create reentrant subVIs and avoid resource conflicts.

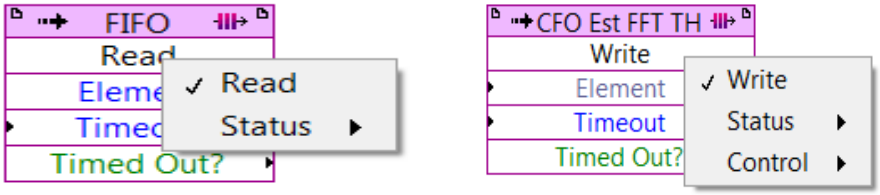
2. **Target-scoped memory items**: Use target-scoped memory items if the memory item to be visible and configurable from the Project Explorer window.

7.3.3 Flat Sequence

A flat sequence structure consists of one or more sub-diagrams, or frames, that execute sequentially. Frames in a *Flat Sequence* structure execute from left to right and when all data values wired to a frame are available. The data leaves each frame as the frame finishes executing. This means the input of one frame depends on the output of another frame. Figure 7.5 shows a flat sequence structure with three frames.

7.3.4 Single-Cycle Timed Loops (SCTL)

LabVIEW automatically optimizes code inside a single-cycle Timed Loop (SCTL) to execute more quickly and consume less space on the FPGA target, compared to the same code inside a *while* loop. The number of clock cycles that the While Loop takes for each iteration is dependent on the code inside the loop. Whereas the SCTL executes all functions inside the loop within a single clock



(a) (b)

Figure 7.3 : (a) FIFO named *"FIFO"* in read mode. (b) FIFO named *"CFO Est FFT TH"* in write mode.

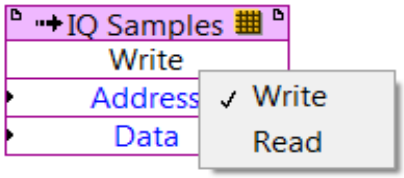


Figure 7.4 : Memory named "IQ Samples"

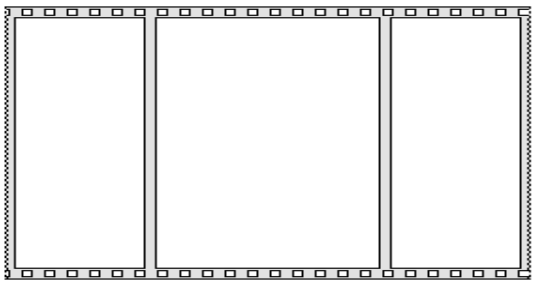


Figure 7.5 : Flat Sequence

cycle. Using the SCTL on FPGA targets reduces execution cycles and saves resources. The register bus and configuration loop, along with the streaming transceiver loop is configured in an SCTL. We do not use SCTL for the algorithm because SCTL does not support functions like FFT, high throughput functions, etc. The default selection is the 40 MHz FPGA global clock. We can use the SCTL with derived clocks to clock the loop at a rate other than 40 MHz. Figure 7.6 shows an SCTL with a default 40 MHz onboard clock.



Figure 7.6 : Single-Cycle Timed Loops

7.4 EXTRACTION OF CONSTELLATION POINTS

As the algorithm depends on the extracted constellation signature, the system depends on prior methods for baud rate estimation, carrier frequency offset estimation, and correction. The symbol rate is estimated using the squared envelope of the complex baseband signal, and CFO is estimated using eighth order non-linearity of baseband sequence. Symbol rate and CFO are estimated in parallel using [Sethi and Ray, 2013], which reduces the overall modulation estimation time. Figure 7.7 shows the block diagram of the preprocessing of signal and extraction of constellation points. After symbol rate estimation, the signal is resampled to m samples per symbol and compared with the previously estimated sampling frequency. The resampled signal is passed for CFO correction only when the calculated difference between the present and previous sampling frequency is less than a set threshold. The CFO corrected signal is passed through the FIR filter for matched filtering. After preprocessing and acquiring signal constellation, the developed

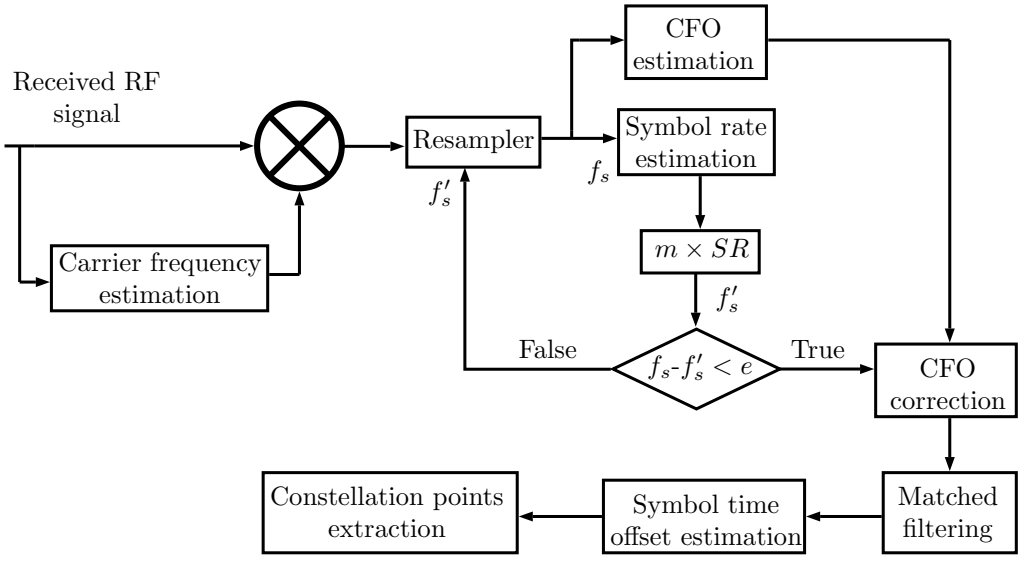


Figure 7.7 : Preprocessing and extraction of Constellation Points.

algorithm shown in Figure 7.8 is applied to the data points for detecting the type of modulation.

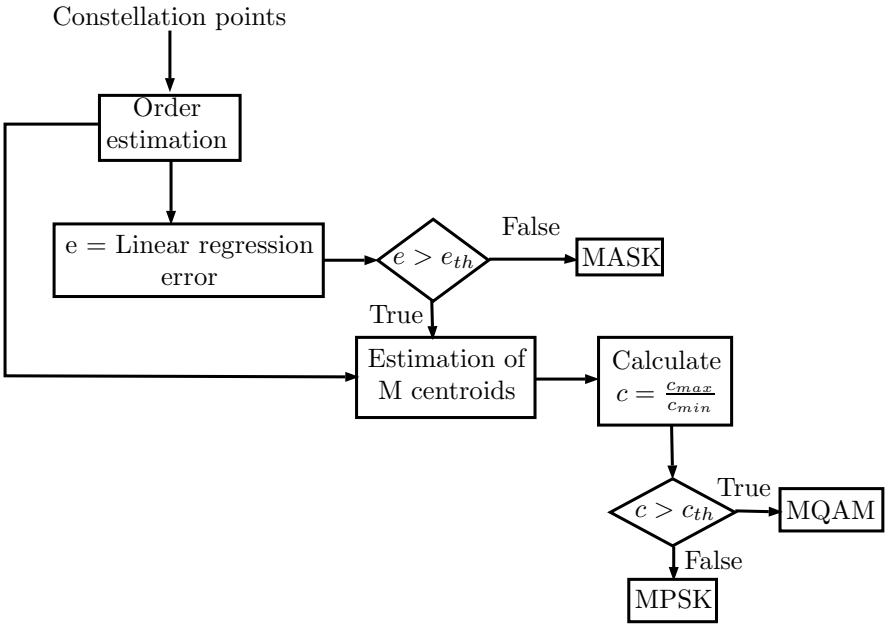


Figure 7.8 : Block diagram of algorithm for domain classification among ASK, PSK, and QAM.

7.5 ESTIMATION OF ORDER OF MODULATION

The order of modulation is detected by estimating the number of clusters in the I/Q plane. The steps described are followed for order estimation.

Input - N constellation points, Seed point (the first point is taken as seed point in the present case).

Output - Optimum number of clusters i.e., order of modulation (M).

Default - Distance values (parameter corresponding to each data point) of all data points are initialized with a large value.

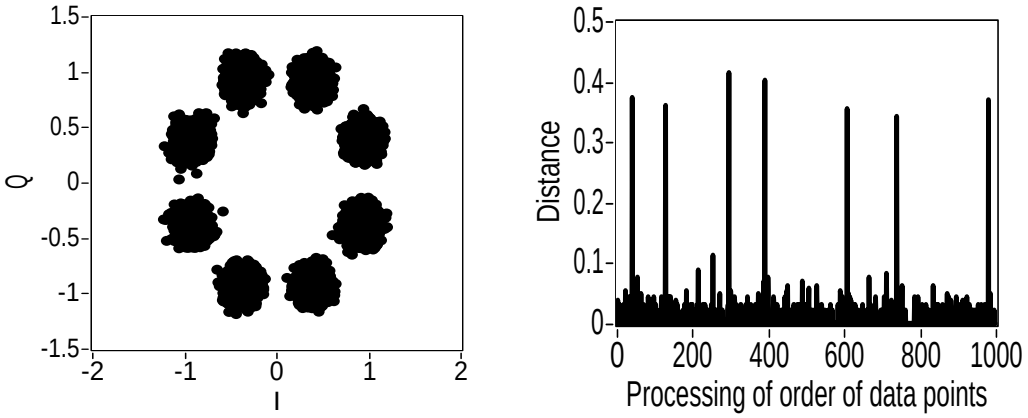
Step1 For total N points, distances between seed point and (N-1) points are calculated. Seed point distance is saved as a default (large) value, and its order of processing is set to 1.

Step2 The nearest data point is taken as a *present point* for further calculation, and its order of processing and distance value (from the previous point) is saved.

Step3 Distances between *present point* and all unprocessed points are calculated. If the distance of any unprocessed point from *present point* is less than the distance from any previous processed point, it is replaced by a lesser distance else, it is not changed.

Step4 Go to step 2 until all points are processed.

Figure 7.9 shows the 8PSK constellation points and its graph between distance and order of processing for extracted data points. Data points within the same cluster (i.e., same symbol) have less value of distances, and when a *present point* switches from one cluster to another, its distance increases. As modulation order can only be in the form of 2^K, distances greater than the threshold and nearest to 2^K, (for $K = 1, 2, 3, 4..$) is chosen as order.



(a) Extracted constellation points of 8PSK

(b) Distance v/s processing order of data points for 5(a).

Figure 7.9 : Estimation of modulation order.

7.6 MODULATION DOMAIN ESTIMATION

Three domains of modulation have been considered viz. ASK, PSK, and QAM. To detect ASK from the pool of ASK, PSK, and QAM, error through line regressed on data points is calculated. This error is less for ASK compared to PSK and QAM, and ASK is identified. If the modulation is not ASK, the following steps are taken for centroid calculation:

Input - L (number of centroids to estimate) seed values

Output- L centroid values

Step1 Calculate distance between the first point and L centroid values. Assign this point to the group to which this point is nearest.

Step2 Repeat step 1 for all points.

Step3 Calculate the new centroid values by taking the mean of data points assigned in the same group and update L values by calculated mean.

Step4 Repeat Step 1 to 3 for 10 iterations.

The ratio of maximum and minimum absolute values of centroids is used to differentiate PSK and QAM. This ratio is lesser in PSK compared to QAM, as symbols have the same amplitude.

7.7 FPGA IMPLEMENTATION OF THE AMC ALGORITHM

With the rapid development of SDR in the past few years, radio receivers can be implemented quickly and efficiently using LabVIEW block diagram coding. To calculate the FFT of the signal and matched filtering, two *ExpressVIs* specific to LabVIEW are used. All other *VIs* are self-designed. Hence the algorithm can be easily replicated to any other tool of interest.

7.8 LABVIEW FLEXRIO ARCHITECTURE

NI FlexRIO system is used to implement the developed algorithm. It provides flexible, customizable input and output for FPGA. FlexRIO system is composed of an adapter module (NI 5791 in the present system) and FPGA (PXIe-7975R consists of Kintex-7 XC7K410T). NI 5791 adapter module defines the physical inputs and outputs of the system and provides continuous frequency coverage from 200 MHz to 4.4 GHz with 100 MHz bandwidth. Second is the NI FlexRIO FPGA Module, which is NI PXIe-7975R FPGA.

7.8.1 Hardware realization of AMC algorithm

AMC system starts acquiring the signal once the *detect* signal is asserted. Carrier frequency is estimated through the power spectrum of the signal. The same frequency signal is generated through the adapter module for downconversion. The signal is sampled with user-specified frequency or with a default value of 10 MS/s.

Acquiring the data

Inphase and Quadrature phase components are received through *RxI* and *RxQ* pin from 14-bit ADC in $I16$ format shown in Figure 7.10. These values are digitized at the desired rate and transferred to another loop after packing in $U32$ format via target scoped FIFO in block memory named *IQSamples*. All the I/Q data retrievals for further processing will be done from *IQSamples* memory by unpacking $U32$ in $I16 - I16$ formats and further converting to the fixed-point data type.

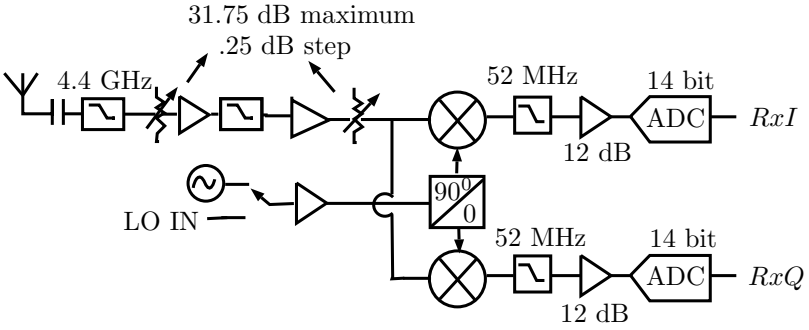


Figure 7.10 : Model of 5791 for acquiring data.

Symbol rate estimation, resampling and CFO estimation

The symbol rate is estimated using 8192 point FFT of the absolute square of the complex envelope. For n point FFT, latency is 2n. Hence, 24576 $(2 \times 8192 + 8192)$ points are acquired and given to the FFT block. After discarding the first 16384 data outputs, further 8192 valid points are continuously compared to find the index corresponding to the maximum value. The frequency corresponding to the desired index is the estimated symbol rate. After the first estimation, new data is continuously acquired with a rate of eight times the estimated symbol rate. A local variable is used to change the sampling rate in real-time. The new symbol rate is estimated from the further acquired waveform. Eight times of new estimated symbol rate, i.e., a new sampling rate is compared with the previous sampling rate. The signal is forwarded to the CFO correction block only when the new estimation is desirably close to the previous estimation of symbol rate to achieve accuracy.

CFO is estimated simultaneously with the symbol rate. The same 24576 points used for symbol rate estimation from *IQSamples* are passed through eight order nonlinearity using a high throughput complex multiply. These functions are similar to the numeric functions but support automatic pipelining and achieves a higher throughput rate inside a single-cycle timed loop. The

frequency corresponding to the location of the maximum value is used to estimate and correct frequency offset.

From a single loop running for 24576 iterations, CFO and symbol rate are estimated. CFO corrected signal is passed to the FIR filter for matched filtering. FIR filter coefficients are set to root raised cosine pulse shaping with eight samples per symbol. Filtered data is used for the estimation of modulation order and domain. Implementation for order and centroid estimation are given in further sections.

Modulation order calculation

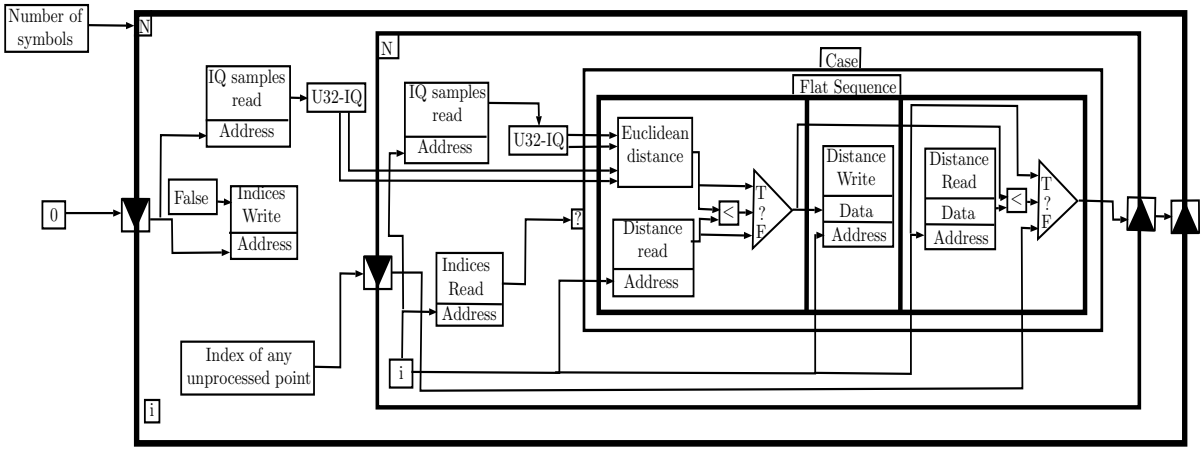


Figure 7.11 : LabVIEW implementation for modulation order calculation.

Figure 7.11 shows the block diagram of the LabVIEW FPGA implementation of the algorithm for modulation order calculation. Three block memories are used for order calculation: *IQ samples*, *Indices*, and *Indices*. *IQ samples* has IQ points, *Indices* initialized with *True* at all addresses and takes count of all unprocessed and processed points. *Distance* is initialized with large value (in present case $8E+0$). Two *For* loops are used. Outer *For* loop runs N times and inner *For* run N times for each iteration of outer *For* loop. The outer shift register is initialized with zero, and the inner shift register is given an index of the unprocessed point from *Indices* (in the first iteration, 0 is marked processed and fed to the inner shift register). When the point is processed, *Indices* at that address is given *False* and case structure is used for considering only unprocessed points.

Case structure contains a Flat sequence, which runs sequentially from left to right. In the first block of flat sequence, for every iteration of the outer *For* loop, the euclidean distance between *present point* and all unprocessed points (addresses wherever indices are true) is calculated and compared with its previous distance. This distance is replaced in the second block of flat sequence if found less. The third block always takes count of the index value, which is nearest to the present point. The final output of the inner *For* loop is the index value of the unprocessed point, which is nearest to the *present point*. That particular point is read from *IQ samples*, and False is written in *Indices* to mark it processed. This is done for all symbols. The final output is the *Distance* array is shown in Figure 7.9(b).

Centroid calculation

Implementation of FPGA for centroid calculation is shown in Figure 7.12. This implementation is explained if the highest modulation order is 16. The same can be replicated for any other order modulation. Two *For* loops are defined of which outer *For* loop runs for ten iterations considering the optimal centroid estimation with less processing time and efficient

resource utilization over hardware. Inner *For* loop runs N (number of symbols) times for each iteration of outer *For* loop. Initial seed values for I and Q are taken as ideal PSK constellation points. One limitation of FPGA is that it does not allocate dynamic memory at run time. So to define the initial centroid points for I and Q, an array of 30 elements is used. This array is initialized with the first 16 values (0-15) as ideal 16-PSK points, 15 to 23 elements with 8-PSK points, 24 to 27 values with QPSK, and the last two with ideal points of BPSK. Depending on the estimated order, an array subset is passed through *Iseeds* and *Qseeds* to the shift registers. An array of 32 elements with I and Q interleaved (initialized with 0) is defined for summon points belonging to the same cluster. An array of 16 elements (initialized with zero) is defined to store the number of points falling in the same cluster. Actual implementation over the hardware has two arrays implemented separately, but these arrays are shown together to understand implementation clearly.

Euclidean distances between each IQ point and seed values are calculated by D given in Figure 7.12. The index of the nearest centroid is assigned to each data point based on the distance values. Present I and Q values are added to the previous values with the same index, and the respective count is incremented by 1. The process is repeated for all the data points. Out of 16 values, the first M values are updated where M is the modulation order. The ratio of maximum and minimum absolute centroid value is calculated. In PSK, this value will be near 1, while in QAM, it is greater than PSK. The ratio is compared with a threshold ($c_{th} = 1.3$) and PSK is differentiated from QAM.

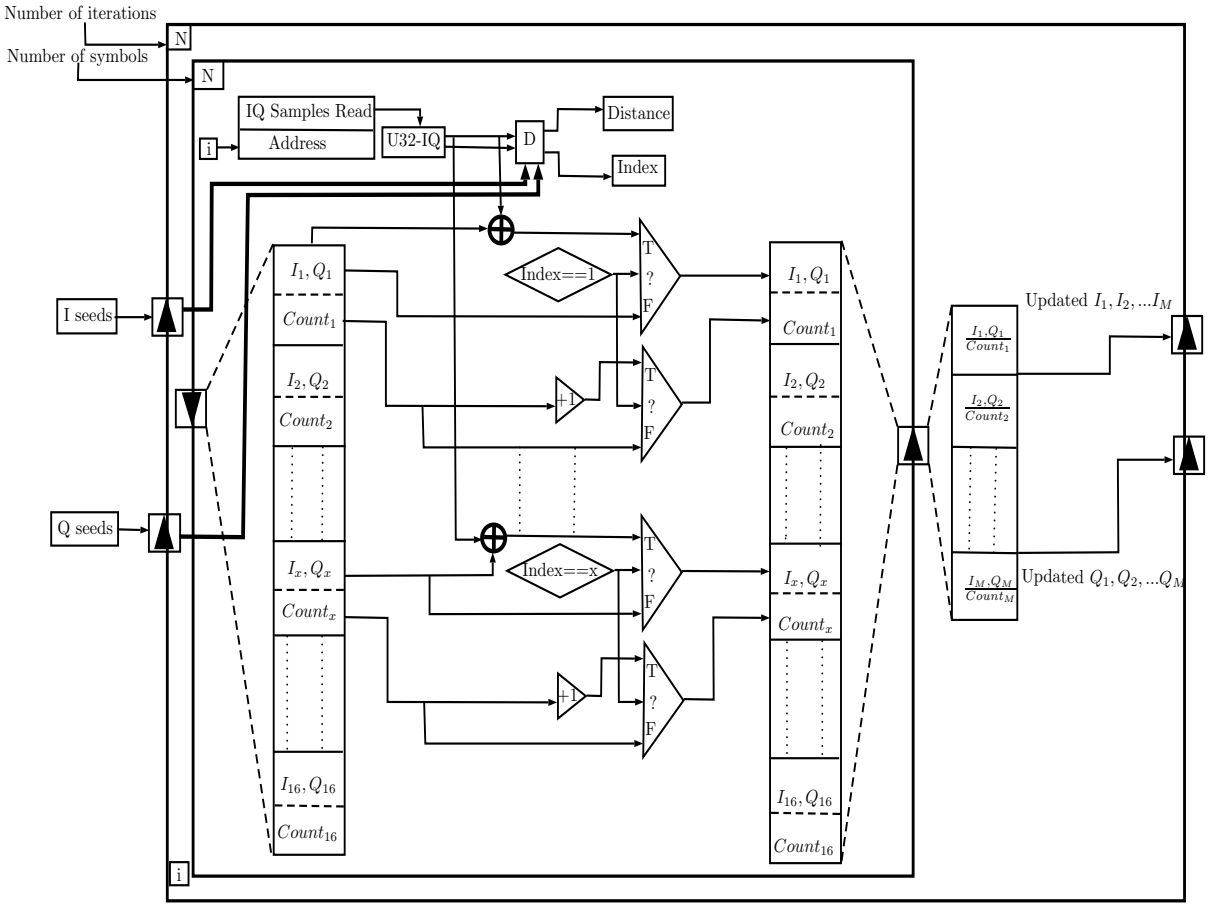


Figure 7.12 : LabVIEW implementation for centroid calculation.

If the highest modulation order considered is changed to 64, the only modification required is to initialize the array of centroid values with points considering 32QAM and 64QAM and passing

the seeds as centroid subset accordingly. Also, the inner shift register will be initialized with a length of 128 values for interleaved IQ centroids and count with a length of 64.

7.9 RESULTS AND DISCUSSIONS

In this section, the performance of the proposed system is evaluated with respect to SNR and compared with some of the existing systems. RF signals for testing are generated using NI PXIe-5673 and received using NI 5791 shown in Figures 7.13(a) and (b), respectively. As the noise in the wireless environment cannot be controlled, to evaluate the classification accuracy, transmitter and receiver are connected through an SMA cable. Signal with known SNR is transmitted through SMA cable, and the accuracy of the algorithm is evaluated. The algorithm is also validated by transmitting an RF signal through a wireless channel and classifying the signal with the receiver placed at a distance of 6.1 mt shown in Figure 7.13(c). In this arrangement, low noise amplifiers (ZX60-P103LN+) are used at the receiver for amplifying the RF signal. Further thresholds used at different parts of the algorithm are analyzed.

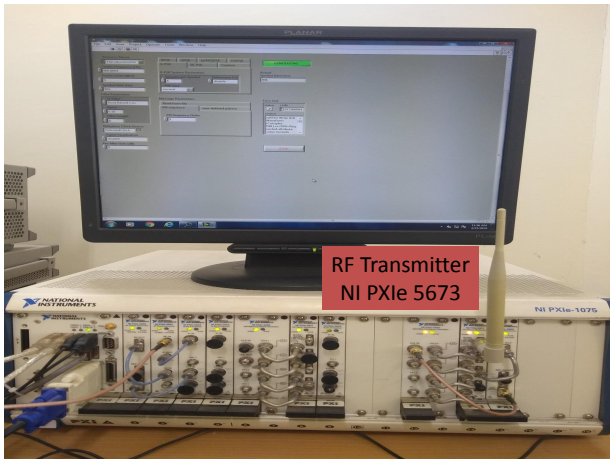
7.9.1 Experimental outcomes

Different magnitudes of Gaussian noise were added in transmitting signals to address the classification accuracy over various SNR values. Classification accuracy is calculated from 1000 signals, each containing 1000 symbols. 2ASK/BPSK and QPSK achieve an accuracy of more than 90% above 8.5 dB and 11 dB SNR, respectively, as shown in Figure 7.14(a). 8PSK and 8QAM require a signal of around 15.2 dB SNR and above to achieve 90% accuracy, while 16QAM requires 16.8 dB for the same accuracy shown in Figure 7.14(b). The number of peaks between 12 and 24 is considered for order 16 as for peaks less than 12; the order will fall into 8QAM or 8PSK while for greater than 24, the order will be near to 32 and will fall into modulation with order 32.

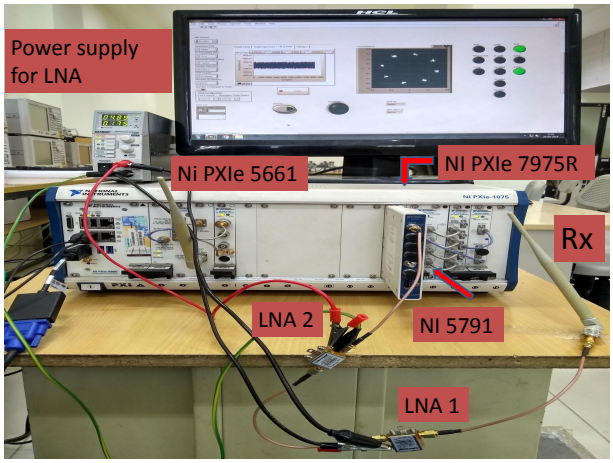
Many algorithms are developed which correctly identify the basic modulations like PSK, FSK, ASK, QAM, etc. but cannot be modified further. Hence this research tries to propose an algorithm to detect the higher modulation scheme, and the system developed can be modified and updated for future requirements. Like for orders 32 and 64, the output array for comparison with the number of peaks has been changed such that it contains values 32 and 64. Rest no change is required, which makes the developed algorithm flexible. Results for 32QAM, 64QAM, 16PSK, and 4ASK are shown in Figure 7.14(c).

Thresholds for linear regression error (e_{th}) and centroid ratio (c_{th}) are defined to achieve maximum accuracy for considered SNR. The values of regression error for all modulations are plotted in Figure 7.15(a). It can be observed from the figure that, regression error for all ASK (with order 2, 4, 8) has a value less than e_{th}, i.e., 0.3. This threshold differentiates ASK from PSK and QAM for all SNR. Also, as SNR increases, regression error for ASK tends towards 0, while for PSK and QAM, it is around 0.5. Similarly, to separate PSK and QAM, the ratio of maximum and minimum absolute centroid value ($\frac{c_{max}}{c_{min}}$) has been used, which is shown in Figure 7.15(b). Figure 7.16 shows the overall accuracy of QPSK, 8PSK, 8QAM, and 16QAM for threshold with different percentages of the maximum peak value. It can be observed that 60% of the maximum peak covers the maximum area, which is set as c_{th}. For higher SNR, as PSK constellation has a circular shape, $\frac{c_{max}}{c_{min}}$ attains a value of 1 while for QAM with order 8 and above, this value is above 1.5. All these analyses are done for 100 signals, each with 1000 symbols.

Hardware execution provides higher performance than most processor-based software programs, but the performance decreases if the resource usage is not proper. Table 7.1 shows that 55.4 % of total resources have been used for implementing the algorithm. Therefore, more algorithms can be developed and can be executed in parallel with the designed algorithm.



(a) Setup for RF signal generation.

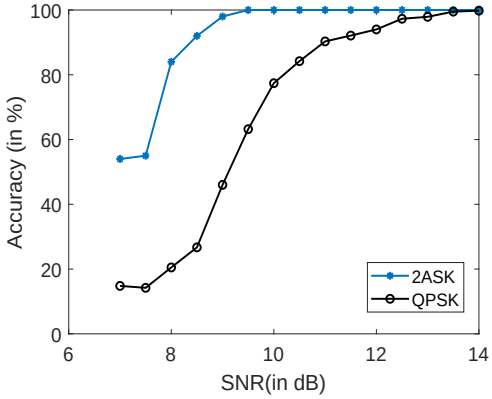


(b) Receiver setup for RF signal reception.

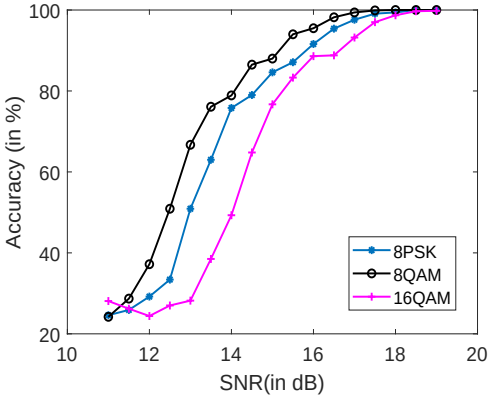


(c) Setup of Transmitter and Receiver in the laboratory for validation of algorithm

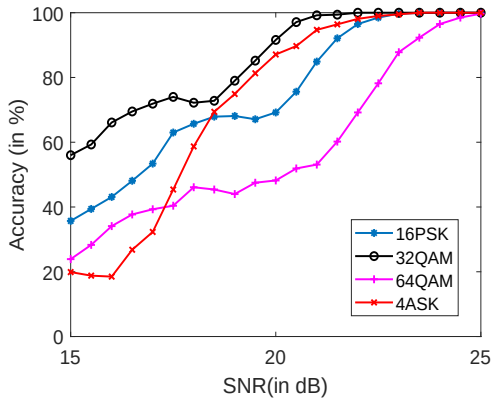
Figure 7.13 : Transmitter and receiver for RF signal



(a) Accuracy versus SNR for 2ASK/BPSK and QPSK modulation schemes.

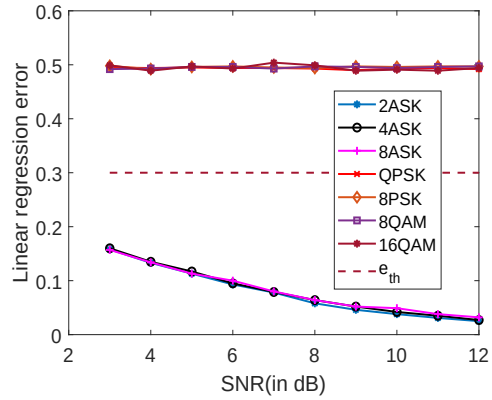


(b) Accuracy versus SNR for 8PSK, 8QAM and 16 QAM.

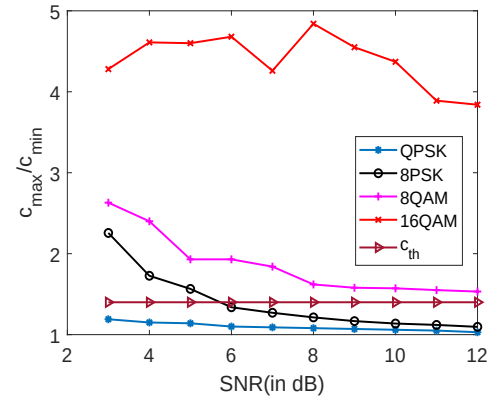


(c) Accuracy versus SNR for 16PSK, 32QAM, 64QAM and 4ASK modulation schemes.

Figure 7.14 : Accuracy versus SNR for different modulation schemes



(a) Linear regression error for different modulation schemes versus SNR.



(b) Value of ratio of maximum and minimum absolute centroid value versus SNR.

Figure 7.15 : Threshold calculation for domain classification

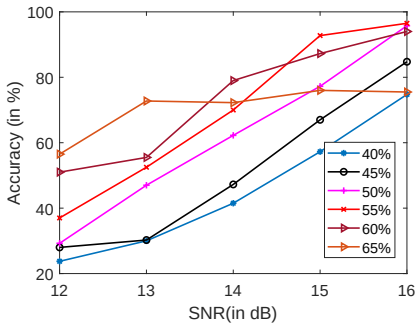


Figure 7.16 : Overall accuracy of correct classification for threshold with different percentage of peak distance value. These distances are obtained while estimating modulation order. This graph shows that for threshold of 60% of maximum value, accuracy is overall higher.

Hardware Resource Utilisation in AMC			
Device Utilisation	Used	Total	Percentage Usage
Total Slices	35236	63550	55.4
Slice Registers	101918	508400	20.00
Slice LUTs	85151	254200	33.5
Block RAMs	260	795	32.7
DSP48s	187	1540	12.1

Table 7.1 : Hardware Utilization

Optimization of the algorithm is done by partitioning the application code into smaller processes.

Real-time performance for different modules has been found and shown in Figure 7.17. Once the detect button is pressed, the program goes into a case structure starting with a *start* case in which the signal is acquired. Figure 7.17 shows that the *algorithm* block throws the output in 56.638 msec, and the preprocessing block for extracting constellation points is taking 24.813 msec. Overall, the system is taking 81.451 msec for computing the modulation scheme after acquiring the signal.

7.9.2 Comparison with other classifiers

SNR	BPSK		QPSK		16QAM	
	[Azza *et al.*, 2018]	Proposed	[Azza *et al.*, 2018]	Proposed	[Azza *et al.*, 2018]	Proposed
10	85	100	85	77.4	75	-
15	90	100	90	99.8	80	76.7
20	94	100	94	99.7	90	99.6
25	96	100	94	99.8	90	99.5

Table 7.2 : Comparison of [Azza *et al.*, 2018] and proposed classifier

There are not a lot of real-time classifiers implemented on FPGA to directly compare the results. Most of the implementations either differ in the choice of the modulation scheme of interest

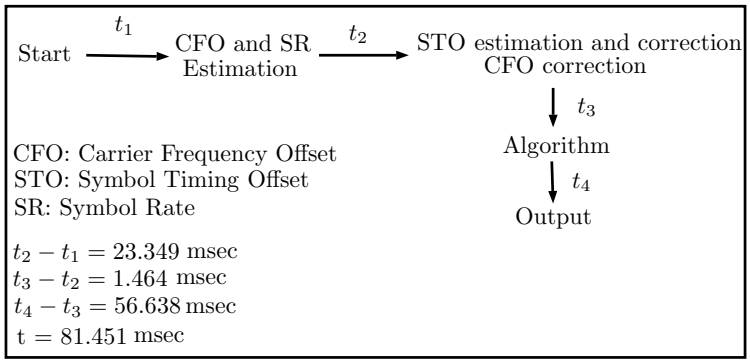


Figure 7.17 : Time utilization for different modules

or detailed implementation is not explained, so it is not possible to reproduce the results. So for comparison, results at a range of SNR covered in this paper are directly compared for the typical modulation schemes. Figure 7.18 shows the comparison between [Ureten and Yensen, 2017] and the present method for the BPSK modulation scheme. This choice of comparison is taken based on results obtained from practical signals with SNR. Figure 7.18 shows that the present method dominates [Ureten and Yensen, 2017] from an SNR of 7.5 dB and above. [Nie *et al.*, 2017] reports 'Shortest distance based on C_{63}' feature can achieve a maximum of 92% accuracy and 'Shortest distance based on C_{63}/C_{21}^3' achieves 100% accuracy above 25 dB for QPSK, 16PSK and 64QAM. From the results obtained, our method outperforms for all these modulation schemes. [Keshk *et al.*, 2015] used DFT and MFCC features with SVM and ANN as classifiers for different orders of PSK. At 20 dB SNR, for 8PSK, the accuracy of [Keshk *et al.*, 2015] is always less than 100% (in OFDM over AWGN with signal features), while our method achieves an accuracy of 100% with testing over 1000 signals. Similarly, accuracy for 16PSK with the same scenario achieves greater accuracy with VLR, RPROP classifiers. Table 7.2 compares the proposed method with [Azza *et al.*, 2018] for BPSK, QPSK, and 16QAM modulation schemes. Accuracy of classifiers based on Moment,

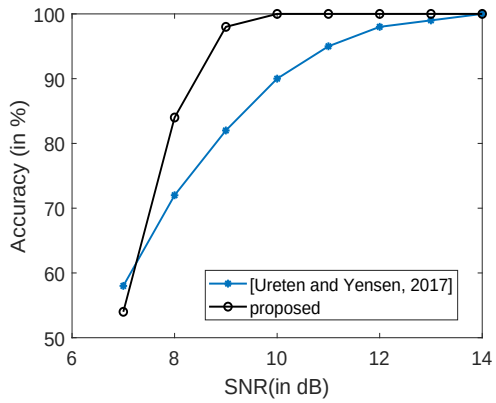


Figure 7.18 : Comparison between [Ureten and Yensen, 2017] and proposed algorithm for BPSK modulation scheme.

Cumulant, GP-KNN, and EM-ML changes with carrier phase offset as given in [Zhu and Nandi, 2015] while the developed algorithm is independent to such phase offsets. By correcting the phase offset, information bit can also be extracted as the system is extracting the symbols and modulation scheme. The main limitation lies in the FFT size, which is 8192 (maximum size available), due to which resolution of spectrum in case of CFO and symbol rate estimation decreases.

...