Flexible Implementation of Multi-standard Demapper Using LabVIEW

BY

MUHAMMAD WAQAS

01-244142-048

SUPERVISED BY

DR. ATIF RAZA JAFRI

Session-2014

A Report submitted to the Department of Electrical Engineering

Bahria University, Islamabad

in partial fulfilment of the requirement for the degree of MS (EE)

CERTIFICATE

We accept the work contained in this report as a confirmation to the required standard for the partial fulfilment of the degree of MS (EE).

Head of Department	Supervisor
Internal Examiner	External Examiner

DEDICATION

I dedicate my thesis to my family and my teacher without whom it would have been impossible.

DECLARATION OF AUTHORSHIP

I hereby declare that content of this thesis is my own work and that it is the result of work done during the period of registration. To the best of my knowledge, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

ACKNOWLEDGEMENTS

I acknowledge, my heartfelt gratitude, debts of thanks to my advisor Dr. Atif Raza Jafri for the continuous assistance, support, patience, motivation, and immense knowledge. His inputs in my research are invaluable. The door to his office was always open whenever I ran into trouble or had a question. I could not have imagined having a better advisor and mentor for my research. It was a pleasure working with him.

Beside advisor I am grateful to National Instruments Pakistan specially Ahmed khalid and Adnan for assisting me in using National Instruments Hardware and tools for Rapid Prototyping

ABSTRACT

Wireless communication standards are evolving rapidly and hence new waveforms are introduced to achieve better spectral efficiency. In a wireless communication system constellation mapping on transmitter side and soft demapper on receiver side are major building block of transmission chain. To construct these different waveforms various standards, support different waveforms constellation mapping ranges from BPSK to 256-QAM are used in different standards. The other related parameters could be use of rotated constellation. In order to comply with multi standards, the demapper should be capable of handling all constraint imposed by the transmitter. This thesis is aimed at hardware implementation of reduced complexity future DVB demapper capable of supporting QPSK to 256-QAM with rotated style in a non-iterative demodulation context. Parameterized hardware implementation is carried out to access the trade-offs between earlier used algorithms and techniques in terms of rapid prototyping in addition to flexibility, throughput and area consumed. The implementation is carried out on NI USRP RIO Kintex 7 device. For rapid prototyping LabVIEW tool kit and associated design flow has been adopted. Synthesis results of our design show that our architecture achieves better throughput over area ratio while comparing with other implementations for most recent DVB-T2 standard.

TABLE OF CONTENTS

LIST OF FIGURES

Introduction

CHAPTER 1. INTRODUCTION

Wireless communication standards are increasing swiftly from decades. Due to rapid increase in standards compatibility issues rise in wireless standards. Some common wireless standards are GSM, LTE, HSPA, DVB-T. DVB-T2 etc.

Increase in wireless communication technology give exponential rise to the rapid, flexible implementation of Software defined radio blocks that include turbo encoder for forward error correction techniques (FEC), BICM Interleaver and Mapper on transmitter side. While on receiver side same blocks are used in reverse manner like demapper, deinterleaver and decoder.

The main aim in SDR is to ensure flexibility to overcome compatibility issues that arise in various standards. In early stages due to non-flexible hardware when technology changes it affects both cost, development effort and time. So from last decades as number of users and communication applications increase it gives rise to flexible and rapid development of radio platform which configure the hardware with respect to the input parameters.

1.1. Motivation

Demapper is necessary part in any digital communication system, which is assigned to generate LLRs. Demapping depends on mapping techniques used on transmitter side. Different standards support different modulation types mentioned in table 1-2.

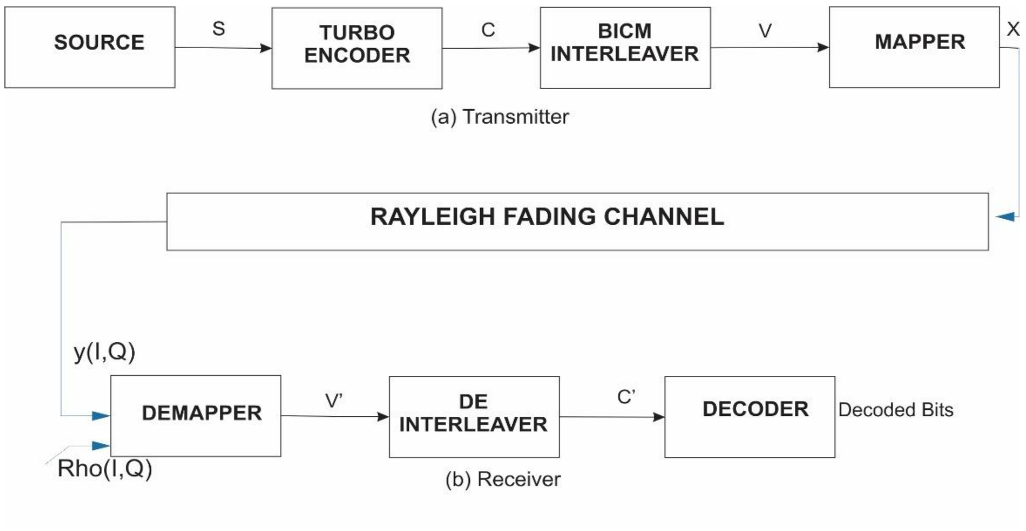


Figure 1-1 System Diagram

Modulation is the process in digital communication in which digital data is mapped to the set of signal waveforms. In thesis we concentrate on QPSK to 256 QAM modulation type. Common modulation type used in different standard is explained below.

1.1.1. Quadrature Amplitude Modulation (QAM)

In Quadrature Amplitude Modulation two signals 90 degrees apart are used and their amplitude is varied with respect to sequence of incoming data. Symbol modulated by QAM are represented by formula in equation a.

$$S(t) = A_c \cos 2\pi f_c t - A_s \sin 2\pi f_s t \qquad \text{(a)}$$

Signals mapped with gray coded constellation is given in figure 1-2.

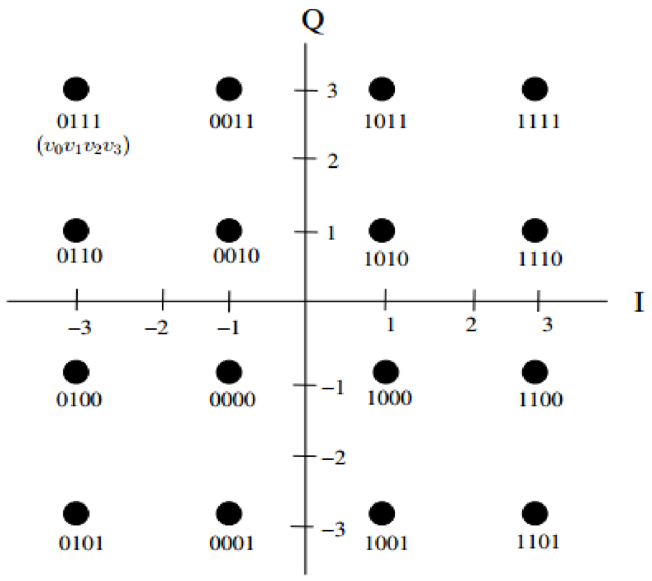


Figure 1-2 Grey Coded 16 QAM

In any QAM modulation type each constellation point of M-Ary type can represent $\log_2(M) = C\ bits$. For Simple 16 QAM the constellation used is given with I and Q axis {-3, -1, +1, +3}. Bits mapping in 16 QAM Gray Coded is given in table 1-1.

V0V1	**I**	***V2V3***	**Q**
00	-1	00	-1
01	-3	01	-3
11	+3	11	+3
10	+1	10	+1

Table 1-1

1.1.2. Correlated I & Q

In Gray Mapping the QAM constellation is divided is divided in to PAM (Pulse Amplitude Modulation) on every component at I and Q axis. As shown in figure 2 and table 1 that *V0* and *V1* are used to map I axis while for Q axis *V2* and *V3* are mapped. So I & Q cannot be uniquely separated.

To compensate this independency correlation between I & Q is required on all constellation points. So due to above every point of constellation will be uniquely identified from both axis.

1.1.3. Constellation Rotation and Cyclic Q Delayed

In DVB-T2 when constellation rotated the normalized values of mapped symbols are rotated in both complex plane and imaginary part is cyclic Q delayed by one cell to uncorrelate the I & Q. The rotation angles proposed in DVB-T2 standard [1] are given in table 1-2.

Modulation	QPSK	16-QAM	64-QAM	256-QAM
Degrees	29,0	16,8	8,6	Atan(1/16)

Table 1-2

These proposed rotation angles in DVB-T2 increase demapper hardware complexity. However rotation angles proposed in [2] reduce the complexity and no of distance computations. Where constellation rotated by angle proposed in [2] for 16QAM is shown in figure1-3.

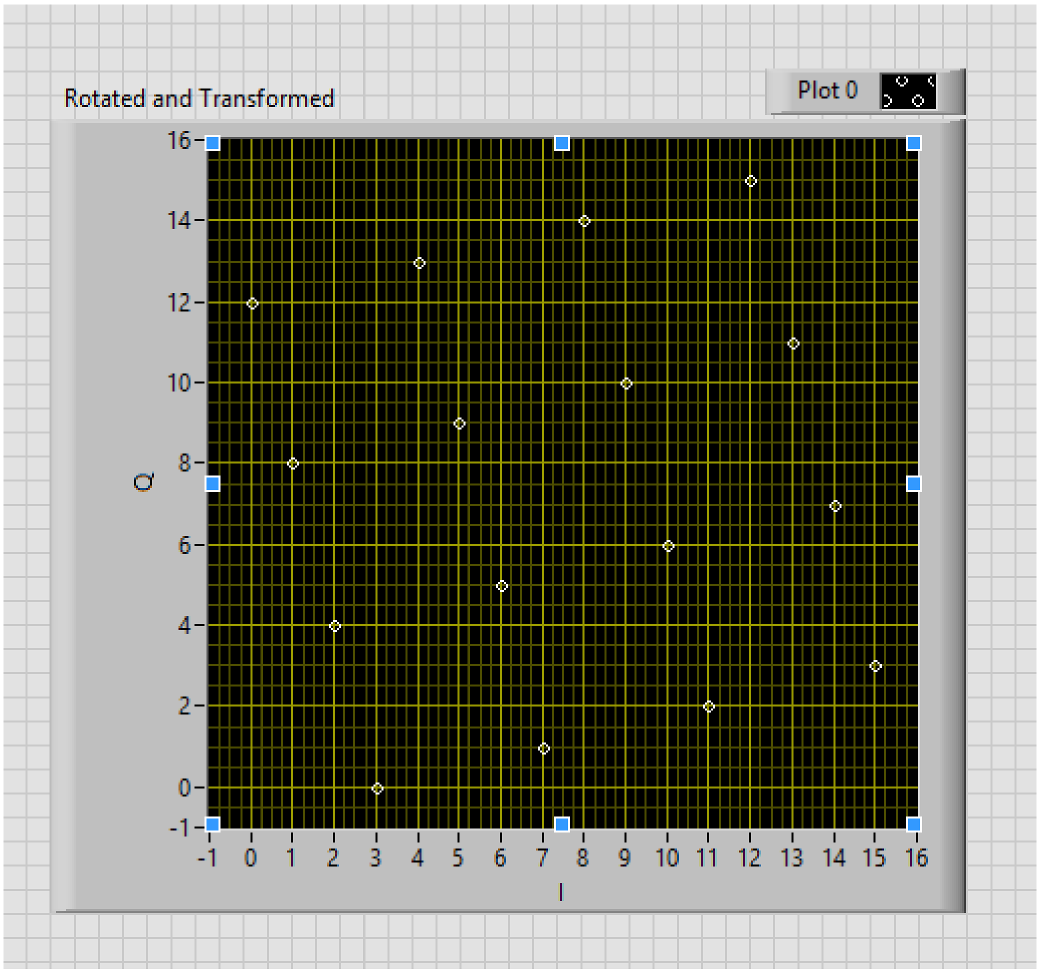
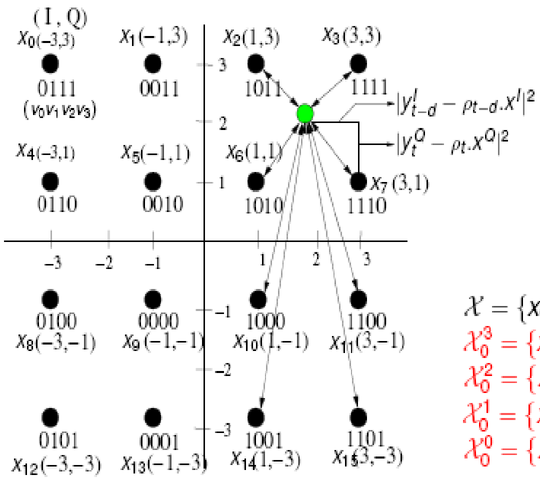


Figure 1-3 Rotated 16-QAM

1.1.4. ML Demapping

One of existing demapping technique is Maximum Likelihood (ML) demapping used in [3]. This technique compute distances from all the points in M-Ary constellation. Over all constellation is divided in to small subsets with respect to bit location and bit value. ML demapping require 2^m computations [3] for one Log Likelihood Ratio(LLR) where 'm' is no of bits per symbol. After 2^m computations the minimum finder computes the minimum distance among each subset to generate LLR. The overall ML demapping 2^m distance computations are shown in figure 1-4.

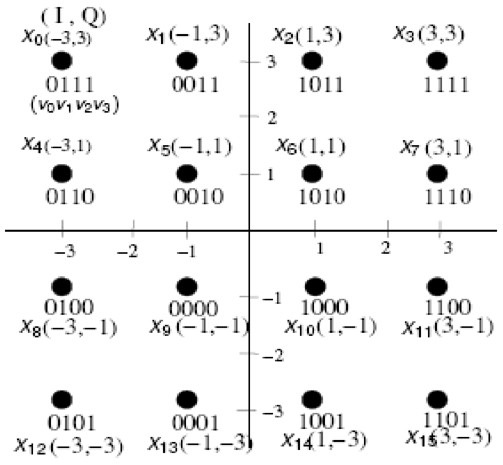


$$\mathcal{X} = \{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}$$
$$\mathcal{X}_0^3 = \{x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}\} \quad \mathcal{X}_1^3 = \{x_0, x_1, x_2, x_3, x_{12}, x_{13}, x_{14}, x_{15}\}$$
$$\mathcal{X}_0^2 = \{x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\} \quad \mathcal{X}_1^2 = \{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$$
$$\mathcal{X}_0^1 = \{x_1, x_5, x_9, x_{13}, x_2, x_6, x_{10}, x_{14}\} \quad \mathcal{X}_1^1 = \{x_0, x_4, x_8, x_{12}, x_3, x_7, x_{11}, x_{15}\}$$
$$\mathcal{X}_0^0 = \{x_0, x_4, x_8, x_{12}, x_1, x_5, x_9, x_{13}\} \quad \mathcal{X}_1^0 = \{x_2, x_6, x_{10}, x_{14}, x_3, x_7, x_{11}, x_{15}\}$$

Figure 1-4 ML demapping [3]

1.1.5. Demapping for Gray Coded Constellation

Gray Coded constellation mapping style is explained in figure 1-5 and table 1. At demapper gray coded constellation requires $2^{\frac{m}{2}}$ -1 computations to generate one LLR [3-5]. In this the whole constellation splits around I&Q axis as shown in table 1 and figure 3. The constellation distances will be computed among both I and Q i.e half for I and half for Q.



$$\mathcal{X}(Q)^3_0 = \{1,-1\} \quad \mathcal{X}(Q)^3_1 = \{3,-3\}$$
$$\mathcal{X}(Q)^2_0 = \{-3,-1\} \quad \mathcal{X}(Q)^2_1 = \{1,3\}$$
$$\mathcal{X}(I)^1_0 = \{1,-1\} \quad \mathcal{X}(I)^1_1 = \{3,-3\}$$
$$\mathcal{X}(I)^0_0 = \{-3,-1\} \quad \mathcal{X}(I)^0_1 = \{1,3\}$$

$$L(v_t^i;O) \approx \frac{1}{2\sigma^2}\left[\min_{x_t^I \in \mathcal{X}(I)^i_0}\left(|y_t^I - \rho_t . x_t^I|^2\right) - \min_{x_t^I \in \mathcal{X}(I)^i_1}\left(|y_t^I - \rho_t . x_t^I|^2\right)\right] \quad \text{for } i = 0 \ldots \frac{m}{2} - 1$$

$$L(v_t^j;O) \approx \frac{1}{2\sigma^2}\left[\min_{x_t^{(Q)} \in \mathcal{X}(Q)^j_0}\left(|y_t^Q - \rho_t . x_t^Q|^2\right) - \min_{x_t^Q \in \mathcal{X}(Q)^j_1}\left(|y_t^Q - \rho_t . x_t^Q|^2\right)\right] \quad \text{for } j = \frac{m}{2} \ldots m - 1$$

Figure 1-5 demapping for Gray Coded [3]

1.2. Functional Flow of SDR

The functional flow of overall system is shown in Fig.1 started from **Source** provide source bits to the FEC for turbo encoding which adds redundant bits in a manner that ensure data reliability on receiver in turbo encoding two encoders are placed one get bits in original sequence while second encoder gets same bits in known interleaved sequence the code rate of turbo encoder is given by $\frac{k}{n}$ where for 'k' source bits the encoder generates 'n' encoded bits the output of encoder may contain both 'parity' and 'source' bits at output.

If the channel is fading channel and fades are deep and encoded bits of source are in sequence the fades may destroy the whole symbol to handle with fading the BICM interleaver introduced to shuffle the encoded bits in known order so that each symbol not contain all the encoded bits of same source bit. Due to this if the symbol is affected by fading then it will be recovered at deinterleaver on receiver.

Channel condition and mapper define at what symbol rate transmission can be done. At last mapper is used to map the interleaved bits. There are different constellation schemes from BPSK to 256 QAM i.e simplest to complex. Each has its symbol rate upto 8 Bits/ Symb. Different wireless standard uses different schemes from BPSK to 256 QAM in non-rotated and rotated both styles. Normally symbols are mapped by gray coded constellation, for different standard modulation types supported are shown in table1-3.

Standard	**Modulation Type**
802.11n	BPSK, QPSK, 16-QAM and 64-QAM
802.16e	QPSK, 16-QAM, 64-QAM
3GPP-LTE	QPSK, 16-QAM, 64-QAM
DVB-SH	QPSK, 8PSK and 16APSK
DVB-S2	QPSK, 8PSK, 16APSK and 32APSK
DVB-T2	QPSK, 16-QAM, 64-QAM and 256-QAM

Table 1-3

At receiver side Demapper demaps the received and distorted symbol by computing Euclidean distances there are different techniques used to demap the symbol each have its own complexity level.

The demapped symbol’s LLR then passed to deinterleaver to gather the bits in original sequence. In last turbo decoder decodes the bits using LLR in iterative process and generates source bits. In existing algorithms when point received on receiver its distance is computed among all points for both and I and Q then global minima defines the LLR to be either 0 or 1

This thesis mainly aims at flexible implementation of low complex DVB-T2 demapper at certain angles which includes QPSK to 256 QAM constellation in rotated manner. It achieves improved performance compared to DVB-T due to using Rotated Cyclic Q Delayed (RCQD). Idea of RCQD is to uncorrelate the real and imaginary parts of the constellation points and it delays the imaginary point that is Q by one cell as shown in figure 1-6 [6].

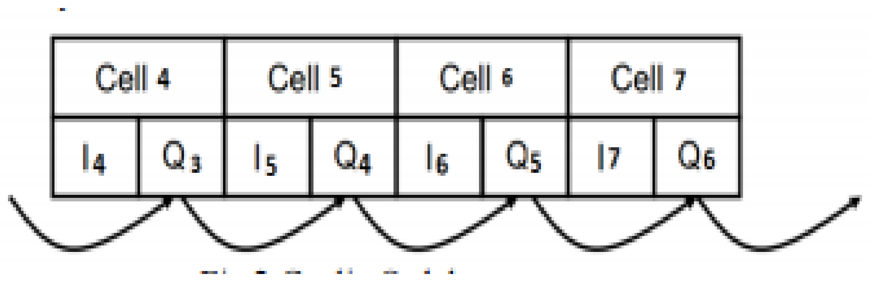


Figure 1-6 Cyclic Q Delayed

1.3. Problem Description

There are several demappers and demapping techniques are proposed for receiver which have their own architecture and algorithm complexity which is related to the constellation mapping used at transmitter end.

However, in DVB-T2 when the rotated flexibility is added to the constellation it gives rise to the hardware complexity and exhaustive search specially in case of higher constellation i.e 256-QAM.

This thesis aims to study already proposed architectures, demapping algorithms and suggest a flexible low complex and high throughput demapper architecture.

1.4. Thesis Objective

Objective of the thesis is:

- Software Modelling and analysis of low complex demapper.
- Rapid prototyping of low complex demapper
- Hardware implementation of low complex future wireless broadcast demapper in multiple domains to analyse with respect to throughput and resources using different implementation approaches and different directives

1.5. Thesis Organization

The thesis comprises on study of digital communication with SDR environment and study of demapper complexity, algorithms and architecture. Different algorithms and architecture explained in Chapter 2 for demapper. In our case we took demapper of DVB-T2 which uses rotated constellation whose complexity prevents it use in wider application. The Thesis implementation is done with experimental setup on NI-USRP RIO (Kintex 7) using LabVIEW FPGA. This organized in following manner:

1. Chapter 2 (Literature review)
2. Chapter 3 (Methodology)
3. Chapter 4 (Implementation and Results)
4. Chapter 5 (Conclusion)
5. References

Literature Review

CHAPTER 2. LITERATURE REVIEW

Wireless communication in SDR context require different data rates to be supported. So with respect to the channel conditions different data rates used like 2 bits/ Sym to 8 Bits/ Sym. These symbol mapping done at transmitter end. While at receiver end demapping of these mapped symbols are required. In SDR to make flexible demapper different architectures and algorithms are proposed such that ML Demapping, Gray Coded Constellation, Demapping for rotated constellations, ASIP Demapper, NISC Demapper, etc. However, in DVB-T2 when constellations are rotated with described angles in table 1-2 and cyclic Q delayed it increases the hardware complexity due to exhaustive search specially in case of higher order constellation.

2.1. Available State of Art Demappers

Different demappers proposed in literature have different architectures and complexity. Some of these demappers are explained below.

2.1.1. Altera Wimax Demapper Model [7]

Altera builds all the reference designs that accelerate the development of an IEEE 802.16e-2005 model. The demapper proposed by Altera support hardware configurable demapper at run time from QPSK to 64-QAM modulation schemes. All symbols normalized to divide equal average power to each symbol. In mapping constellation points are multiplied with scaling factor.

For efficient hardware the constellation demapper is fully time division multiplexed. In 16-QAM modulation case it acquires data at every six clock cycles. As Altera demapper support maximum 64-QAM so its bus is not fully utilized in QPSK and 64-QAM case.

In Altera demapper IQ data is converted from parallel to serial the passed to decision calculator which determine the decision metrics used for bit polarity and again data is converted to parallel for quantization interval the block diagram of Altera Demapper is shown in figure 7.

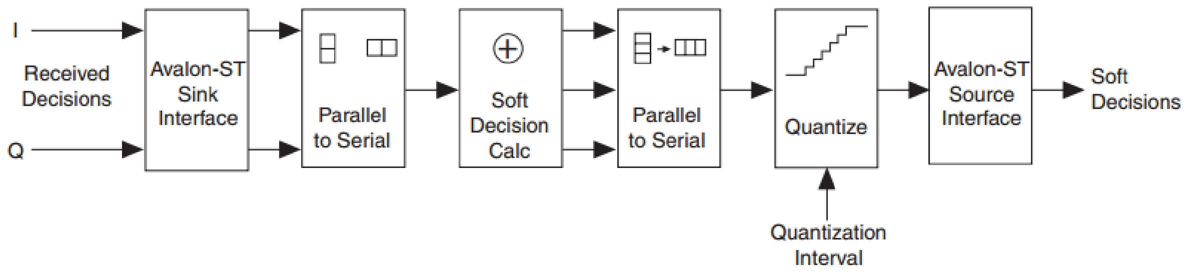


Figure 2-1

The synthesis results for Altera designed demapper are shown in table 4.

Device	Combinational ALUTs/LUTs	Logic Registers	Memory		9×9 multipliers	f_{MAX} (MHz)
			M512	M9K		
Stratix III EP3SE80F780C3	338	809	0	1	0	258
Cyclone III EP3C80F780C6	378	741	0	1	0	224

Table 2-1

Whereas our design explained in chapter 3 take input at every clock cycles that gives high throughput.

2.1.2. Rapid Design of Universal Soft Demapper [4]

Conducted the research on rapid design and prototyping of universal soft demapper. Their research provided steps in development of rapid design, validation and prototyping with the purpose of creation of multi standard ASIP based universal soft demapper. They presented the results in which ASIP provided flexibility to support large numbers of modulation types with the usage of up to 8 bits per symbol in context of turbo and non-turbo as shown in figure below.

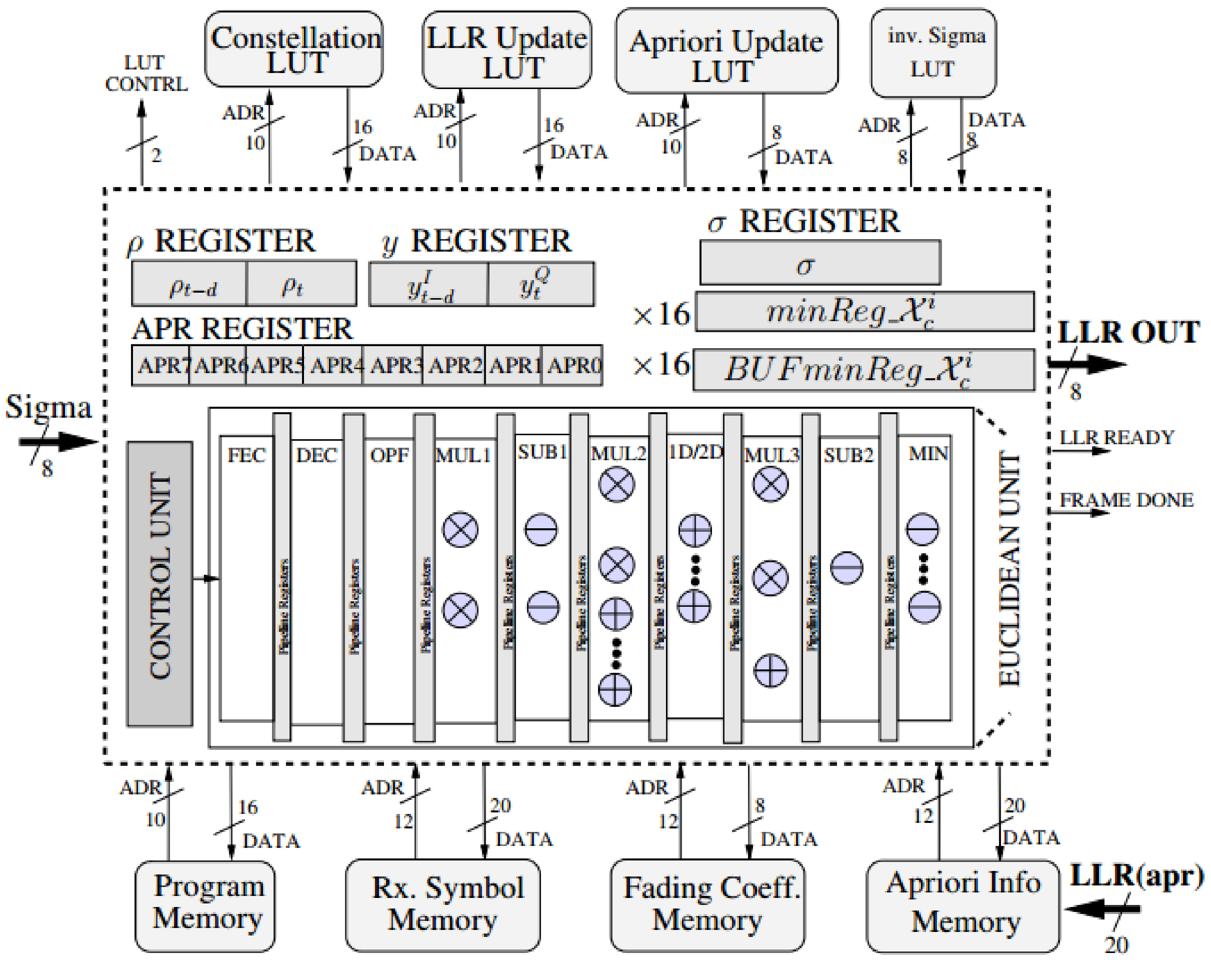


Figure 2-2 ASIP Architecture

For rapid design development ASIP modelling was done in LISA ADL till FPGA implementation. Logic emulation board with the integration of Virtex 5 LX330 FPGA prototype reached at a throughput of 102 Mega LLR/Sec for Grey mapped 16-QAM constellation at a clock frequency of 156 MHZ with 9 pipeline stages. 1596 slice registers, 2627 slice LUTs and 6 DSP48Es were used in the reduced size of ASIP with the purpose of increased and higher throughputs.

Our work model the demapper in C language and implemented in LabVIEW FPGA and LabVIEW FPGA IP Builder contributes in designing a parameterized hardware approach and implementing a new low complex demapper for DVB-T2 which consumes less resources and achieve throughput of 836.82MLLR/Sec in QPSK.

2.1.3. NISC Model for SISO Demapper [5]

Carried out research in the field of wireless digital communication to deal with the problem of increased level of complexity and diversity. The major purpose of researchers in this field is to reduce complexity and increase the performance of the system without any disturbances and errors. Flexibility along with increased performance is the major concern but it requires design approach that can provide the better controlling and managing hardware resources. ASIP design approach provides flexible design with imposed dynamic scheduling of a set of instructions relevant to decoding. NISC design approach is helpful in reduction of this kind of instructions overhead and ultimately improves the performance. This research explored NISC approach in the case of universal demapper for various wireless standards. Different state-of-the-art ASIP were used in the main architecture for comparison purposes. These results indicated that proposed design showed significant improvement in execution time and implementation area while using same flexibility parameters. The proposed demapper design supports iterative demodulation and produces LLRs in different modulation schemes initiated from BPSK up to 256 QAM with and without SSD using any mapping design as shown in below figure.

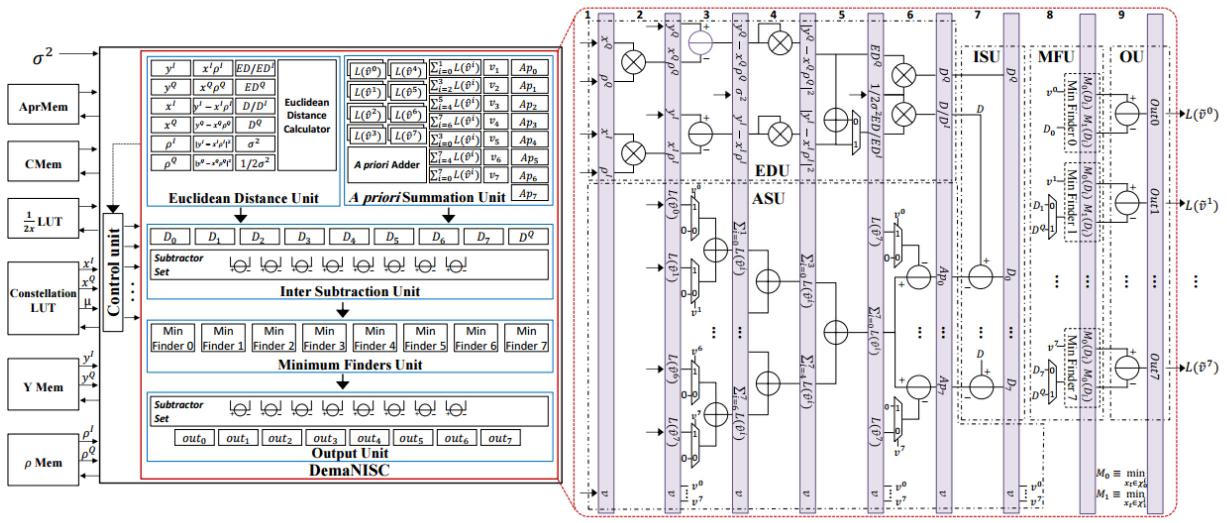


Figure 2-3 NISC Architecture

Our work contributes by following the parameterized hardware approach which also reduce the instruction overhead that results in reducing initiation interval and give rise to throughput of the demapper. Two approaches implemented one using LabVIEW FPGA and LabVIEW FPGA IP Builder best results reflected by IP Builder because it gives much controls over designing techniques like pipelining, loop unroll, initiation interval memories implementation and Clock Rate.

2.1.4. Design of rotated DVB-T2 QAM Mapper/ Demapper [8]

[8] Worked on signal space diversity (SSD) in association with rotated QAM mapper and demapper for DVB-T2 standards. They proposed the design with the purpose of efficient SSD and improved performance of QAM constellation over fading channels. They examined the decomposition of constellation into two-dimensional sub-regions in signal space. These were also associated with the algorithmic simplification led to novelty of this work. It was observed that overall complexity of the demapper reduced significantly as their algorithm requires 25 computations for 64-QAM as whole constellation is divided such that each region has 25 points so instead of computing distance among 64 points it uses only 25. This research also provided the description of design and FPGA prototyping of resultant proposed architecture. These least complex design and improved performance showed the efficiency of detection method.

2.1.5. Low Complexity Soft-Decision Demapper for high order modulation of DVB-S2 System [9]

Worked on implementation of soft-decision demapper for high order modulation to reduce the complexity. They tested the proposed model which could operate at a symbol rate. By replacing the parallel to serial converter between M-PSK demodulator and the soft-decision demapper. It was observed that soft-decision demapper reduced the complexity of the hardware due to reuse of multipliers. These proposed modifications were supportive for high-order modulation modes and verified using of FPGA board having Xilinx Virtex II. Further they also tested that this proposed design can be adopted in different cables and wireless communication systems.

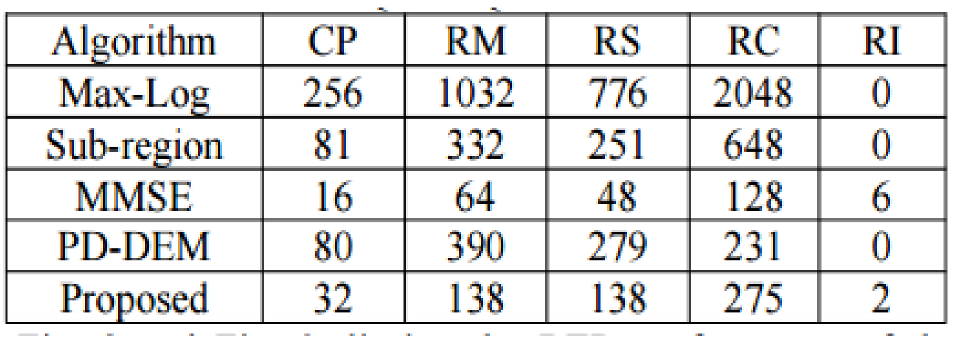
2.1.6. ASIP Based Universal demapper for Multi-Wireless Standards [3]

Designed a universal ASIP-based flexible demapper. In this concern their design showed full flexibility in a range of low complexity QPSK grey mapped constellation to high complexity 256-QAM support with turbo demodulation framework. This proposed architecture was also flexible in usage of demapper both in iterative and non-iterative receivers. All these results and design showed the throughput of 606 mega LLR per second for 16-QAM grey-mapped constellations.

2.2. A Low-Complexity 2D Signal Space Diversity Solution for Future Broadcasting Systems [2]

Examined the performance of current demappers and different standards used in industry for improving performance of over fading channels. For this purpose, they tested RCQD in DVB-T2 modulation and assessed its substantial gains in severe channel conditions. In this concern they suggested different rotation angles for various QAM constellations leading to low-complexity detection method. Different comparison of the rotation angles with the current standards produced the efficient results. They concluded that proposed solution reduced the complexity of the current system and it also simplified the transmitter and receiver leading to efficient performance in comparison of current proposed angles in DVB-T2. All of these results indicated that overall 60% complexity was reduced.

In our demapper we use this approach to verify, model, design and implement low complex and high throughput demapper. The demapper proposed in [6, 7, 9-11] do not use RCQD however this algorithm uses RCQD and reduce the complexity by order $O(2\sqrt{M})$. This proposed algorithm is used to design a low complex demapper with rapid prototype development tool and technique. The overall hardware resource usage of algorithm is given in below table with very good results although MMSE gives much lower computation complexity but on other side it has worst performance results in simulations as shown in table below.



Algorithm	CP	RM	RS	RC	RI
Max-Log	256	1032	776	2048	0
Sub-region	81	332	251	648	0
MMSE	16	64	48	128	6
PD-DEM	80	390	279	231	0
Proposed	32	138	138	275	2

Table 2-2

Methodology

CHAPTER 3. METHODOLOGY

The work started after studying different wireless standards and SDR blocks. All standards have their own complexity and architecture. Demappers were studied in detail with respect to algorithm and architecture complexity, as DVB-T2 uses rotated constellation whose complexity increase when used with SSD which prevent its use in large applications.

First of all, authentic literature is gathered which also include study of DVB-T2 standard, Altera Mapper/ Demapper Implementation for Wimax with respect to Gray Coded Mapping and ML demapping techniques as discussed above. The recently proposed algorithm explained in section 2.2 reduce the DVB-T2 complexity by 60% with RCQD was analyzed and modeled in C to verify the results and complexity, C modelling is explained in next section because software modeling gives good analysis on flexible parameters, memories, algorithm behavior and some idea about proposed architecture. In second phase algorithm was implemented in LabVIEW using fixed point datatype after getting successful results same code is imported in LabVIEW FPGA project under a target. At last same demapper is implemented in LabVIEW FPGA IP builder to get more optimized results.

3.1. C Modelling of Proposed Demapper

In DVB-T2 gray coded constellation mapping are used on all QPSK to 256-QAM [1] In the start all the constellations are normalized by their normalizing factor given in table 5.

Constellation Style	**Normalizing Factor**
QPSK	$\frac{z}{\sqrt{2}}$
16QAM	$\frac{z}{\sqrt{10}}$

64QAM	$\frac{z}{\sqrt{42}}$
256QAM	$\frac{z}{\sqrt{170}}$

Table 3-1

The specific angles proposed in [2] given in equation 2 are applied to all constellations points using equation1.

In proposed model mapper/ demapper is first of all normalized points are rotated by certain angles depending upon modulation type. This rotation describes some interesting properties of rotated constellation [2].

1. As I and Q are uniformly distributed over I&Q axis. So the distances among all M points will be given by $d_{1D,min} = 2\beta_s \sin\theta$ 1a.

$$\begin{cases} z_i = s_i \cos\theta - s_Q \sin\theta \\ z_Q = s_i \sin\theta - s_Q \cos\theta \end{cases} \qquad 1$$

$$\theta = \tan^{-1}\frac{1}{\sqrt{M}} \qquad 2$$

2. The points z_I &z_Q transformed after rotation such that the transformed integer belonged to [0-M-1], transformation is given by equation 6 & 7.

$$T_I = \frac{z_I}{d_{1D,min}} + \frac{1}{2}(M-1) \qquad 6,$$

$$T_Q = \frac{z_Q}{d_{1D,min}} + \frac{1}{2}(M-1) \qquad 7,$$

3. The difference between consecutive two transformed I and Q points will always be 1.
4. At receiver the received y_I and y_Q are equalized as equation 8:

$$Y_m = \frac{y_m}{d_{1D,min}\, h_m} + \frac{1}{2}(M-1) \qquad 8,$$

5. At receiver this rotation technique requires $2\sqrt{M}$ distances to be computed for 1 LLR.
6. The regions among which distances computed are given by equation 4.

$$T(Y_m) = \begin{cases} [0,2d-1], & if\ Y_m < d, \\ [M-2d, M-1], & if\ Y_m \geq M-d, \\ [\lfloor Y_m \rfloor - d + 1, \lfloor Y_m \rfloor + d & else, \end{cases} \qquad 4,$$

Where $d = \frac{\sqrt{M}}{2}$

Unlike ML demapping and demapping for Gray coded constellation which require all 2^m and $2^{m/2}$ computation for one LLR. The proposed algorithm requires maximum 2√M computations for one LLR in all four type of modulation.

In DVB- T2 standard grey coded mapping is applied to the points with the angles to rotate each constellation, so in our low complex demapper we also use grey coded mapping but with certain proposed angles as given in equation 2 rotation is done when points are mapped by equation 1 and theta is given by equation 2. Let the grey coded mapping points for 16-QAM are given by in step 1:

```
double simple_constellation_16QAM[16][2] = {{3.0, 3.0},{3.0, 1.0},{1,3},{1,1},
                                            {3, -3},{3, -1},{1,-3},{1,-1},
                                              {-3, 3},{-3, 1},{-1,3},{-1,1},
                                              {-3, -3},{-3, -1},{-1,-3},{-1,-1}};
```

Where index of all sixteen points are mapped in such a way that each index corresponds to symbol number of the point. After mapping all these points are normalized by factor given in DVB-T2 specification sheet as in code given by in step 2:

```
Nomlzd_simple_constellation_16QAM[i][j]=(simple_constellation_16QAM[i][j]/(sqrt(10.0)));
```

After normalizing all 16 points now rotation is required in mapping in step 3, so normalized points are rotated using equation 1 and 2 and in C given by:

```
        rotated_constellation_16QAM[i][0] =
Nomlzd_simple_constellation_16QAM[i][0]*cos(atan(1/sqrt(16.0))) -
Nomlzd_simple_constellation_16QAM[i][1]*sin(atan(1/sqrt(16.0)));
        rotated_constellation_16QAM[i][1] =
Nomlzd_simple_constellation_16QAM[i][0]*sin(atan(1/sqrt(16.0))) +
Nomlzd_simple_constellation_16QAM[i][1]*cos(atan(1/sqrt(16.0)));
```

Rotation is the last step on transmitter side, after rotation points are transmitted through the channel where there are no of parameters affects the transmission like noise and fading. Received point is then transformed by equation 8.

By observing equation 8 for 16-QAM it is noted that one portion here in equation is constant i.e 7.5 for 16QAM as M=16 while in second portion distance between two consecutive points are given by equation 1a so instead of using division we take reciprocal then multiply to perform operation. On receiver the transformation of received point is given in C code as:

```
Y_I = y_I_16QAM/(d_1D_min_16QAM*h_I) + 7.5;
Y_Q = y_Q_16QAM/(d_1D_min_16QAM*h_Q) + 7.5;
```

Where y_I_16QAM and y_Q_16QAM are the rotated and received points through noisy and fading channel. Transformation converts the received point into an integer point also receiver have rotated and transformed copy of all 16-QAM points then there is a bit wise operation performed on the points which lies in the region selected with respect to the received point the more detail of region selection is explained in implementation section. In 16-QAM case the region consists of 8 points among these 8 points distance will be computed four for I component and four for Q component. The transformed points of 16-QAM are sorted in order for both I and Q and initialized in receiver memory with index as symbol number. In second stage in receiver after region decision each point is fetch from transformed sorted memory to compute distance from received point the distance computation formula is given in equation 2b:

$$LLR\,(v_t^i : O) \approx \min_{x_t \epsilon x_0^i}\{h_I(Y^I - T^I))^{\,2} + (h_Q(Y^Q - T^Q))^{\,2} \,.\, (\tfrac{1}{\sigma^2})\} \;\; -$$

$$\min_{x_t \epsilon x_1^i}\{h_I(Y^I - T^I))^{\,2} + (h_Q(Y^Q - T^Q))^{\,2} \,.\, (\tfrac{1}{\sigma^2})\} \qquad 2b,$$

As equation 2b is used to compute distance where $Y^I$is the received and transformed point for I component while T^I is the transformed point fetched from demapper memory with respect to the region. In this all fetched point's symbol numbers are processed by bit wise operation as shown in C code below and in EU distance computation considering h_i = h_q=0.

```
                shifter=3-k;
                b[k]=((Sorted_I_16QAM[i][0]&bh[k+4])>>shifter);
                if (b[k]==0)
                {
temp_f[k][0]=(pow((Y_I-Sorted_I_16QAM[i][1]), 2))+ (pow((Y_Q-Sorted_I_16QAM[i][2]), 2));
```

Equation 2b in detail describes three main operations to generate one LLR i.e first to compute distances second these distance computations are bit wise dependent also shown from equation that if bit is 1 at bit location 0 or bit is 0 at bit location 0 this bit wise operation is performed on symbol number i.e eight-bit unsigned number gives eight bits from 00000000 to 11111111. Last operation is the minimum finder which computes the minimum distances among all distances computed with respect to bit location and bit value the decision is based on LLR value if the LLR<0 then decoded bit is 0 and if the LLR>0 then decoded bit will be 1. The output of demapper is LLR in case of iterative demapper, the erroneous output of demapper is handled with turbo decoder of using FEC techniques in the communication block.

3.1.1. Results and Analysis of C Modelling

C modelling code shown above is only for 16-QAM mapping and demapping, while in actual modelling is done from QPSK to 256-QAM to model the low complex demapper. The overall analysis of modelling shows that this demapper requires three Block Rams of depth 680 elements. One ram is of transformed sorted I components, second ram is of transformed sorted Q component and third ram is symbol numbers.

Also there is a need of control signal line which configure the demapper to required modulation type and inside demapper control signal configure memories, Loops, counters, regions etc.

The overall arithmetic operations involved in several blocks of demapper are:

Arithmetic Operation	**Transformation I**	**Transformation Q**	**Region I**	**Region Q**	**Distance Computation**
Addition	1	1	1	1	1
Subtraction	0	0	1	1	2
Multiplication	1	1	0	0	4

Table 3-2

The blocks involved in demapper has I and Q transformation, I and Q region and Distance computation with minimum finder. We have to design a high throughput but most optimized architecture of a demapper with least resources. If we look at blocks transformation for both I and Q can be executed simultaneously and same with region computation both can be processed in parallel. Whereas distance computation is a bit wise operation also it involves to fetch sorted I, sorted Q and symbol numbers from block memories using counters and counters start address depends on the selected region. Also this operation is performed in nested loops outer most loop executes twice one for I component and one for Q component whereas internal loop iteration depends on control signal i.e which modulation is selected for different modulation no of iterations are as follows:

1. For QPSK internal loop iteration equals to 2
2. For 16-QAM internal loop iteration equals to 4
3. For 64-QAM internal loop iteration equals to 8
4. For 256-QAM internal loop iteration equals to 16

The loop iteration number is also selected by control signal that which modulation is used for transmission control automatically configure the loop counter to required number for distance computation as internal loop is enclosed in another for loop which runs for two times one for I and one for Q so at last we can say that:

Modulation Type	**Points Required to demap one Symbol**
QPSK	4
16 QAM	8
64 QAM	16
256 QAM	32

Table 3-3

No of memories and counters used are:

Memories	**Depth**	**Counters**
Sorted_I	680	1 Counter to access points from these memories
Sorted_Q	680	
Symbol Number	680	
I_Offset	4	These memories are accessed by control signal i.e with respect to modulation type
Q_Offset	4	
Transform_Add_Offset	4	
Transform_Multiply_Offset	4	

Table 3-4

3.2. LabVIEW Modelling

Based on C modelling for rapid prototyping LabVIEW is selected. Model described in above section is implemented in LabVIEW with fixed point data type to ensure the results. The developed model tested in simulation mode on development computer to verify the results for next phase.

3.3. LabVIEW FPGA Development

By verifying LabVIEW simulation results. In this phase a LabVIEW FPGA project is created on FPGA target and same LabVIEW model file is imported under the FPGA project for synthesis and to ensure results on FPGA target.

3.4. LabVIEW FPGA IP Builder Development

To build a standalone demapper which can be used as a block in any SDR system which has reduced area and high throughput, LabVIEW FPGA IP Builder is used to get the required DVB-T2 throughput demapper and high throughput demapper. This is achieved by implementing the same algorithm but setting different directives set.

Implementation and Results

CHAPTER 4. IMPLEMENTATION & RESULTS

4.1.1. LabVIEW FPGA Basics

LabVIEW FPGA is used as a tool to implement proposed demapper in parameterized hardware approach reason for using LabVIEW FPGA is because it provides a good interface for rapid prototyping and deployment on dedicated hardware. LabVIEW FPGA is a graphical programming language which generates a LabVIEW vi for Xilinx compiler. Xilinx compiler extracts VHDL constraints from vi file and generates a bit file for target as shown in figure. Bit file generation process include synthesizing, resources estimation and timing estimation, Placing and routing.

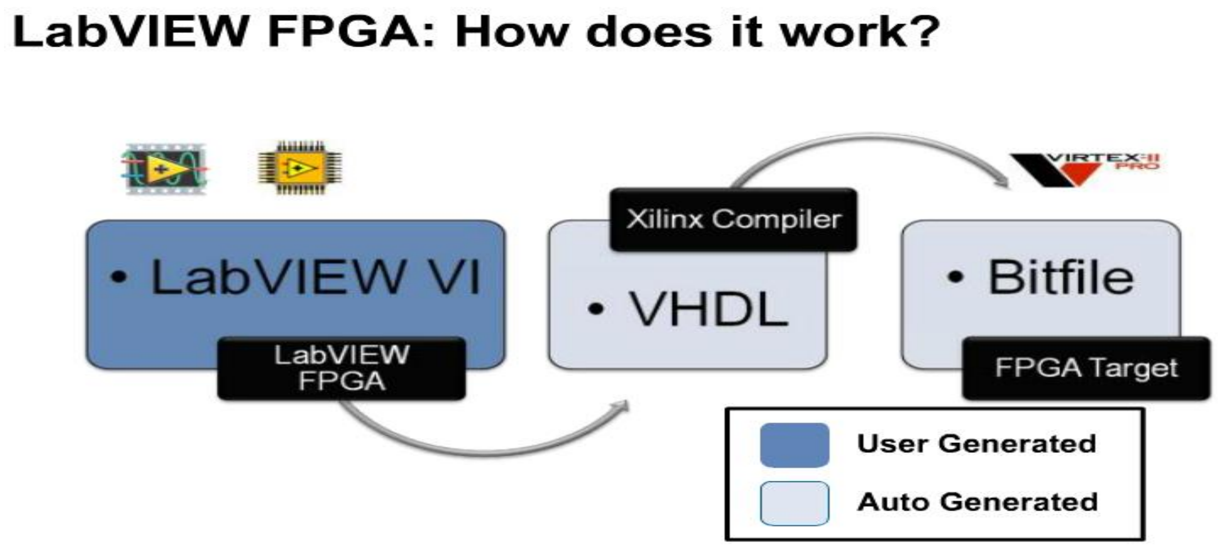


Figure 4-1 LV FPGA Working [12]

It is necessary to learn execution flow in LabVIEW FPGA. As it is a graphical programming language. LV FPGA to ensure data synchronization and execution flow uses enable chain. In LV, node executes when data is available at its input when node finishes execution the ata is passed to next node. Let's take example of NOT function execution in LabVIEW as shown in figure:

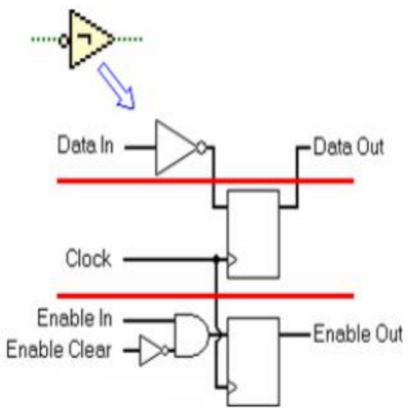


Figure 4-2 Enable Chain [12]

In this figure the code is transformed in to three sections Logic, Synchronization and Enable Chain. In this the first upper portion which correspond to logic and second portion for synchronization register which ensure that data will only available at output on rising edge only and the third portion enable chain is used to ensure that the logic implemented execute in hardware in same manner as implemented on Block Diagram. So LabVIEW implements the enable chain after every operation automatically to ensure auto pipelining and data synchronizing.

If user wants to avoid enable chain and wants to observe critical path using LabVIEW FPGA one must have to implement all the code inside Single Cycle Timed Loop (SCTL). SCTL remove all enable chain registers and one can observe the critical path inside the SCTL one of example of simple code with SCTL and without SCTL is shown in figure below. This example shows that the code inside SCTL takes 1 cycle to complete on other hand code in while loop takes 6 cycle to complete the same code as shown in below figure. Benefit of using LV FPGA is that one can run different piece of codes at different clock rate as LV FPGA support derived clock rates also.

Modes of a LabVIEW FPGA VI. LabVIEW has two main execution mode.

- Reentrancy Mode.

Reentrant allow the program to execute in parallel whenever this process called by multiple instances LabVIEW allocate its separate copy on hardware to execute the process. So this has lowest call overhead, maximum resources and fastest execution.

- Non Reentrant Mode.

Non Reentrant allow the program to use a single copy of hardware called by multiple process. This has minimum hardware resources and not as fast as reentrant mode.

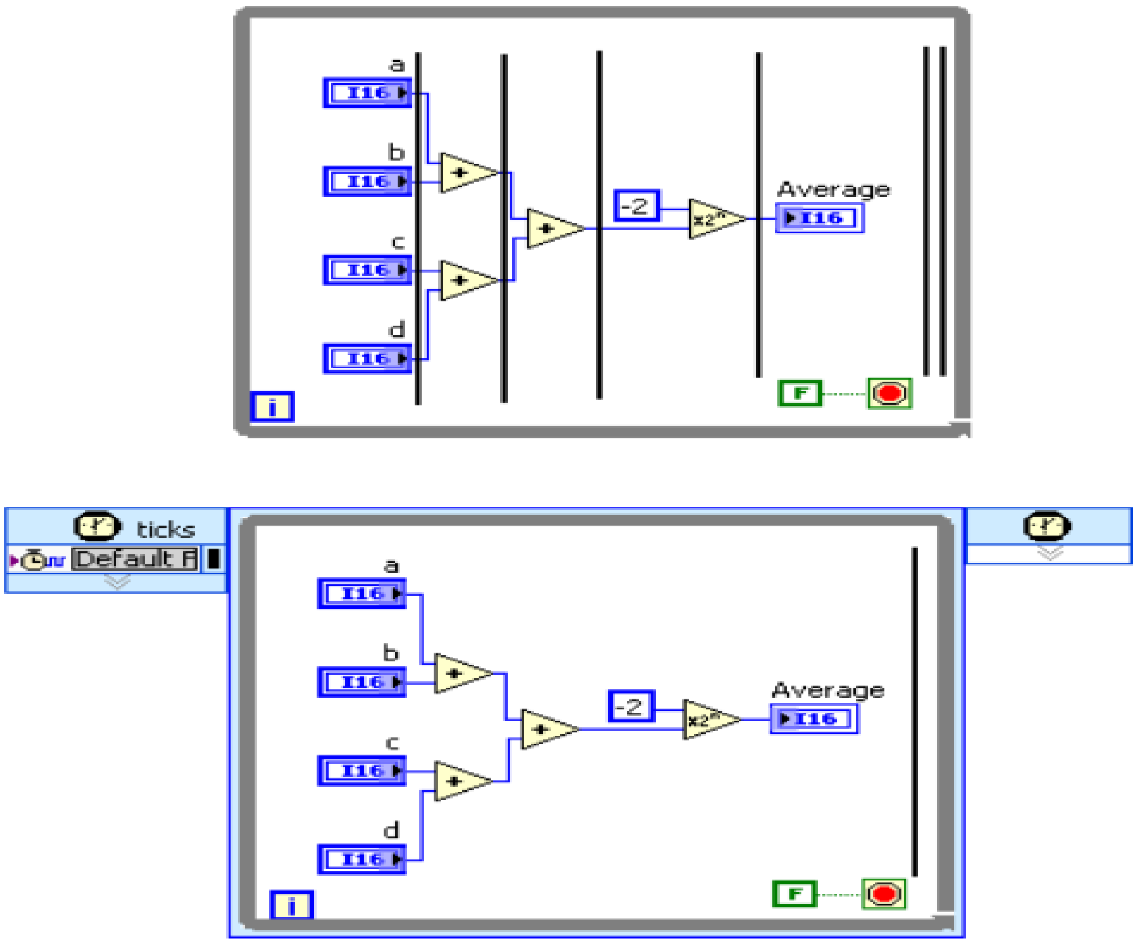


Figure 4-3 SCTL [12]

After implementation of all the hardware logic right click on VI and select target as development computer with simulated I/O to check and verify the algorithm results. The module wise hierarchy of implemented demapper is shown in following figure and its detail implementation in further topics.

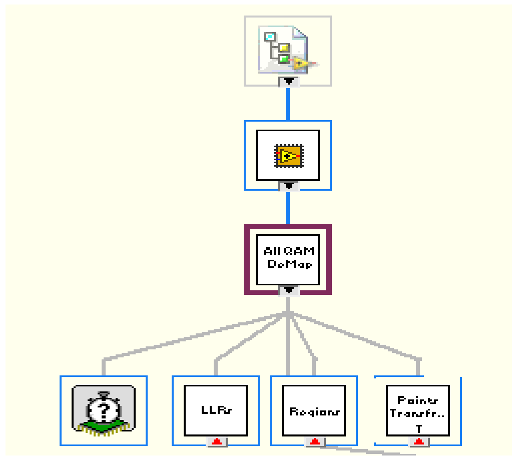


Figure 4-4 Proposed Demapper Hierarchy

4.1. Experimental Setup

Experimental setup of demapper includes a RIO platform USRP-RIO which supports 2x2 MIMO transceivers with tunable frequencies from 50MHz to 6GHz, Kintex 7 FPGA, 120MHz Bandwidth per channel and with an integrated GPS receiver. LabVIEW FPGA is used to program RIO platform which include all HDL and Vivado concepts in configuration. The experimental setup figure is shown in figure below:

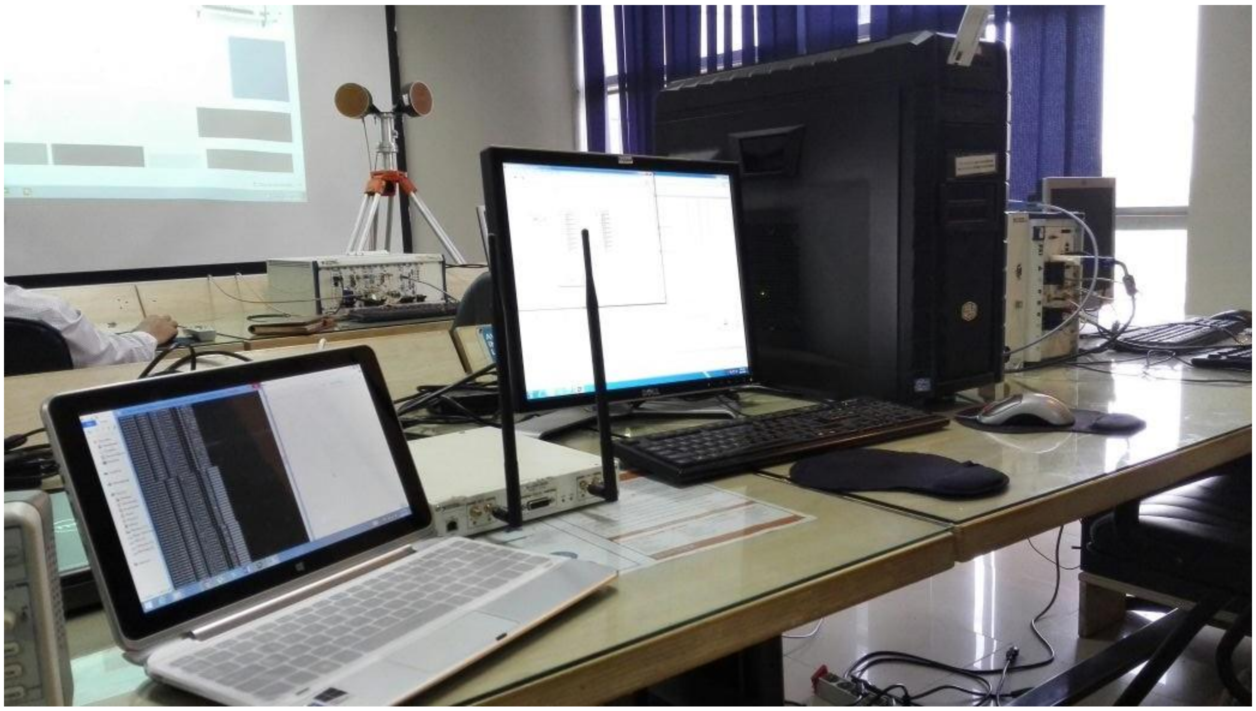


Figure 4-5 Experimental Setup

4.2. Proposed Demapping with LabVIEW FPGA

The algorithm proposed in [2] has three stages through which received points are processed to generate LLRs. Three stages include Transformation, Region and distance computation with minimum finder. The whole demapper is configured by a control signal to select type of constellation. Implementation of these three stages are following.

4.2.1. Transformation

Equation 6 and 7 gives the complete details of transformation which includes one adder, one multiplier and two memories for add value as constant and multiplier value as constant with respect to selected constellation scheme as shown in figure 8a. As M and $d_{1D,min}$ are the parameters in equation 6 and 7 have different values of add and multiply constant for all four constellations. The points received are first processed by transformation which has one 25-bit multiplier and one 26-bit adder for I component transformation and same will be for Q component. The two transformation block for I and Q will execute in parallel as shown in figure 8a. While architecture and implementation in figure 8b. When any constellation type is selected on control line the add and multiply offset memory values are automatically fetched for transformation, different values for different constellations as shown in table 6. In this table adder constant memory allocated 8 bits in total where out of 8, 7 will be designated to integer and 1 for fractional part.

Where multiply constant memory is represented in 13 bits out of them 7 for integer part and rest for fractional part.

Control Signal	**Adder Constant Memory**	**Multiply Constant Memory**
0	1.5	1.578
1	7.5	6.51562
2	31.5	26.125
3	127.5	104.5

Table 4-1

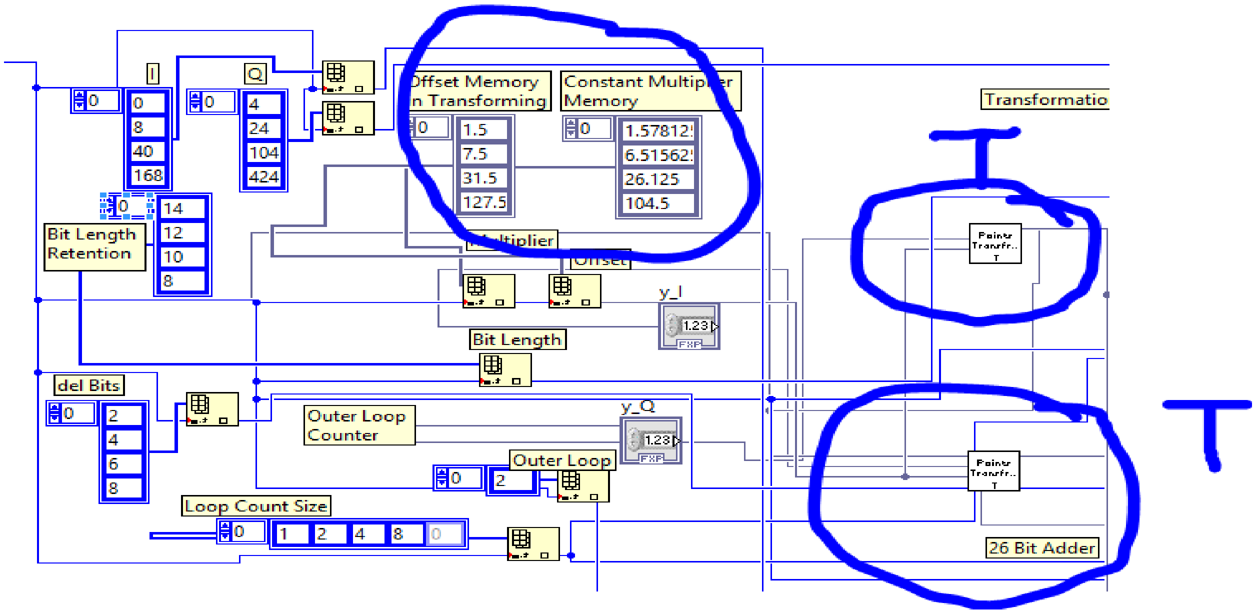


Figure 4-6 Transformation (a)

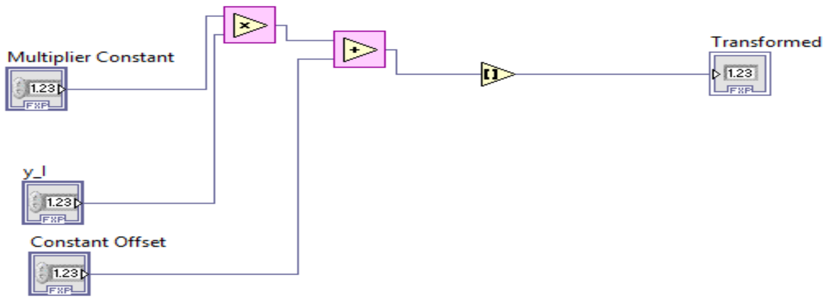


Figure 4-7 Transformation (b)

The parallel implementation of transforming blocks for I and Q is configured by execution mode i.e re-entrant or non-re-entrant explained in section 3.3.1.

Over all the transformation consist of following components:

- One Adder (26)
- One Multiplier (25)
- Two Memories (depth 4)

4.2.2. Region Computation

After the transformation y_I and y_Q both are converted to Y_I and Y_Q simultaneously and simultaneously regions computed using Y_I and Y_Q. In regions frame there are two Mux one for I and one for Q using case structure. The control line is mapped to Mux control line to select type of constellation. Region computation is a function which is used in reentrant mode. Which compares the incoming transformed values with respect to equation 4 either region 1 or region 2 or region 3 will be selected. The detailed implementation of region computation is explained in below section.

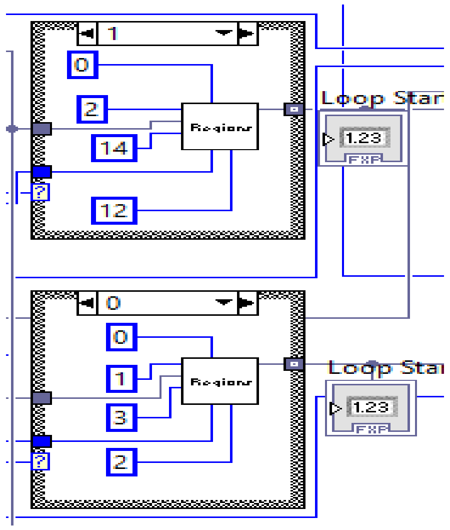


Figure 4-8 Region

4.2.2.1. QPSK Rotation & Demapping Region

First of all, at transmitter side gray coded mapped constellation is normalized by described factor, constellation diagram for gray coded style in QPSK is shown in figure 4-8. Also a LUT is used to store the values of gray coded QPSK constellation points where LUT index is the symbol number i.e at each symbol number there are two points I and Q. In second step the whole LUT is divided by normalizing factor.

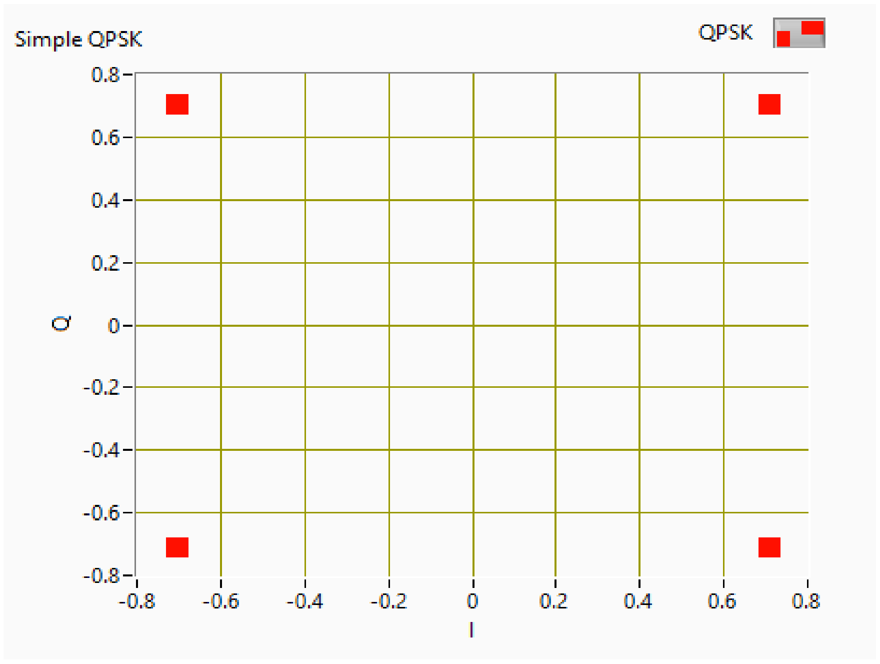


Figure 4-9 QPSK Normalized

These normalized points i.e four for each I and Q are processed through equation 1 and 2 to project them at specific angle. Equation 2 describe 26.5^{o}rotation angle of QPSK as shown in figure4-9. After rotation each symbol is transmitted through fading channel.

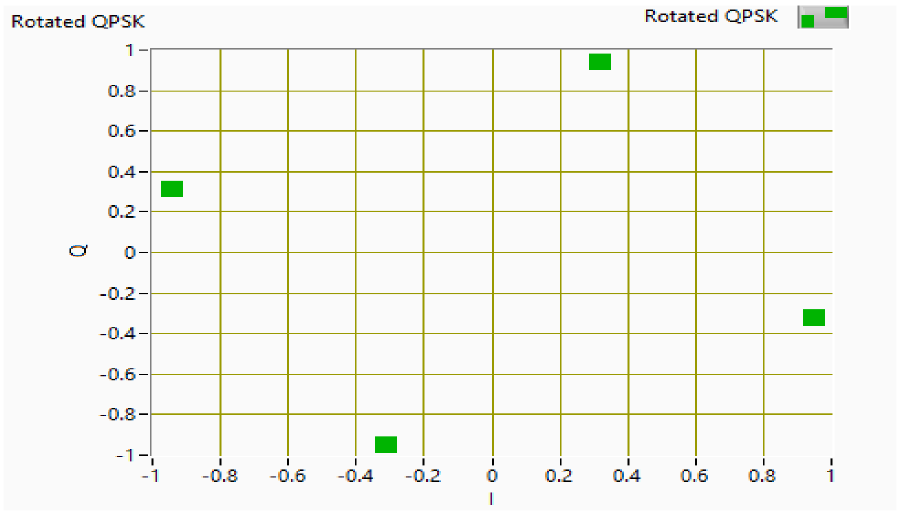


Figure 4-10 QPSK Rotated

Receiver has a rotated and transformed copy of each constellation in LUT for distance computation in a manner that sorted I and sorted Q. When rotated point is received through fading channel first of all received point is equalized and transformed shown in "red" color in figure 10. Regions are computed with respect to the received points. Radius used by QPSK demapping is 1. There are two constant regions in each constellation, in QPSK case

if $Y_I < 1$ then search region will be between 0 to 1 is constant region,

else if $Y_I \geq 3$ then search region will be between 2 to 3 is constant region,

and last region is dynamic with respect to the received point i.e if point received is ≥ 1 and less than 3 than dynamic search region will be in $[\lfloor Y_m \rfloor - d + 1, \lfloor Y_m \rfloor + d$.

From figure let if the point received is shown by red point in figure 4-10 (0.25, 1.75). As I component has value less than 1 so search space is region 1 for Q component whose value is 1.75 is case of dynamic region, the search space is between 1 to 2.

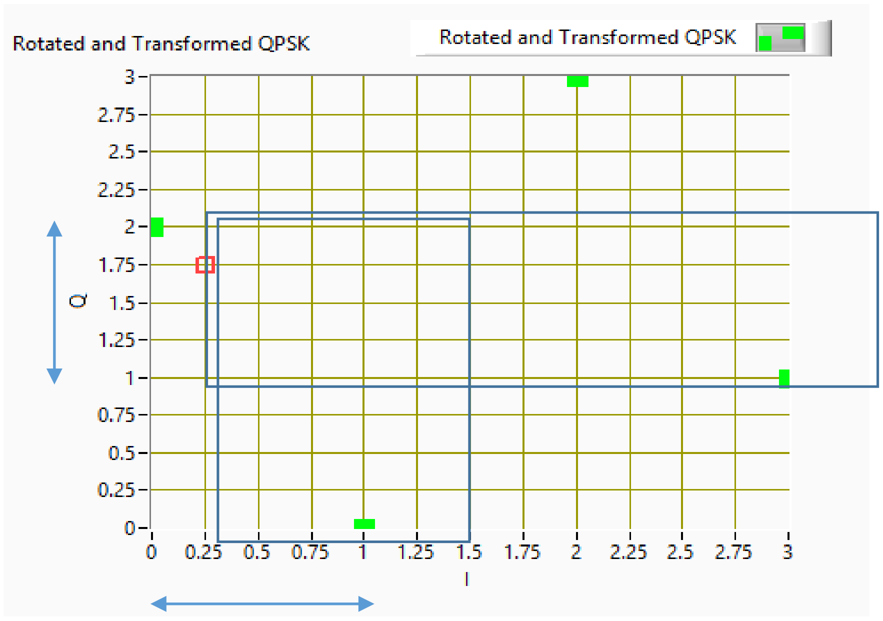


Figure 4-11 QPSK Rotated & Transformed

The distance computation on I axis will be done in I first constant region as shown in figure i.e there are two points in I first region Euclidean distance will be computed with these two points using equation 5, values will be saved in with respect to symbol binary number i.e 0 or 1 registers. Same process of region computation is done for Q point which has a dynamic region as shown in figure two points lies in Q dynamic regions so in this way distance will be computed between two pints using equation 5.

Here is QPSK it is observed that instead of using all 8 points four each for EU distance only points in regions give accurate LLR generation. This low complex distance computation gives more clear idea in higher order constellation demapping case.

4.2.2.2. 16 QAM Rotation & Demapping Region

In 16-QAM on transmitter side firstly all points are gray mapped as shown in figure 4-11 in array as a LUT with respect to the symbol number on LUT index. LUT has sixteen rows and two columns where column one corresponds to I component and column two corresponds Q component where belonging index is the symbol number.

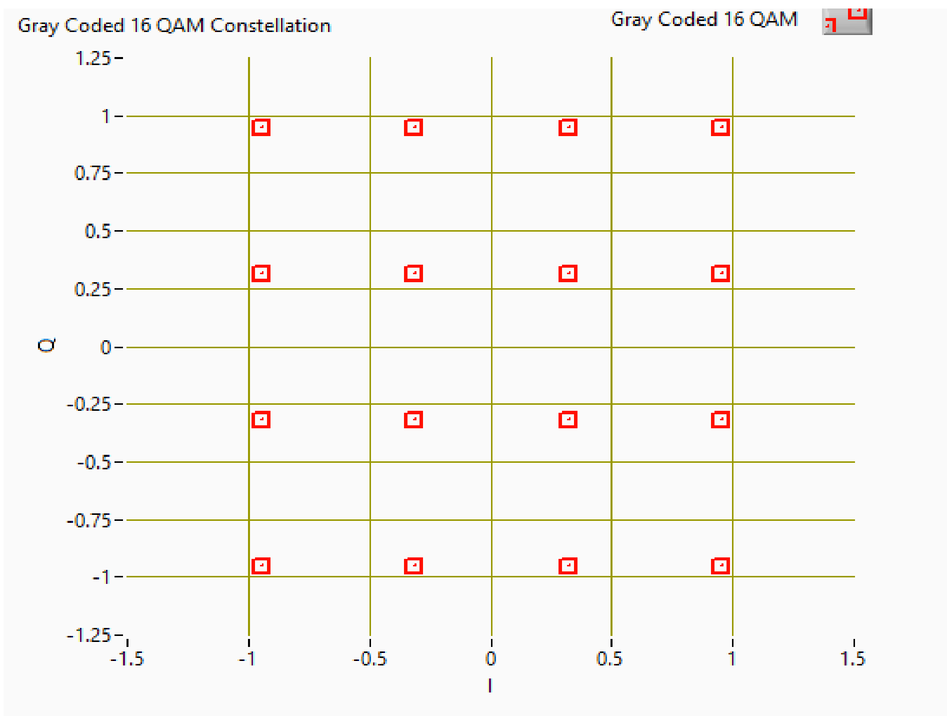


Figure 4-12 16QAM Normalized

In second step all these sixteen points were normalized by factor $\frac{1}{\sqrt{10}}$. The rotation angle computed for 16QAM is 14.03^o. The overall constellation is Rotated Cyclic Q Delayed by angle 14.03 a s shown in figure 4-12.

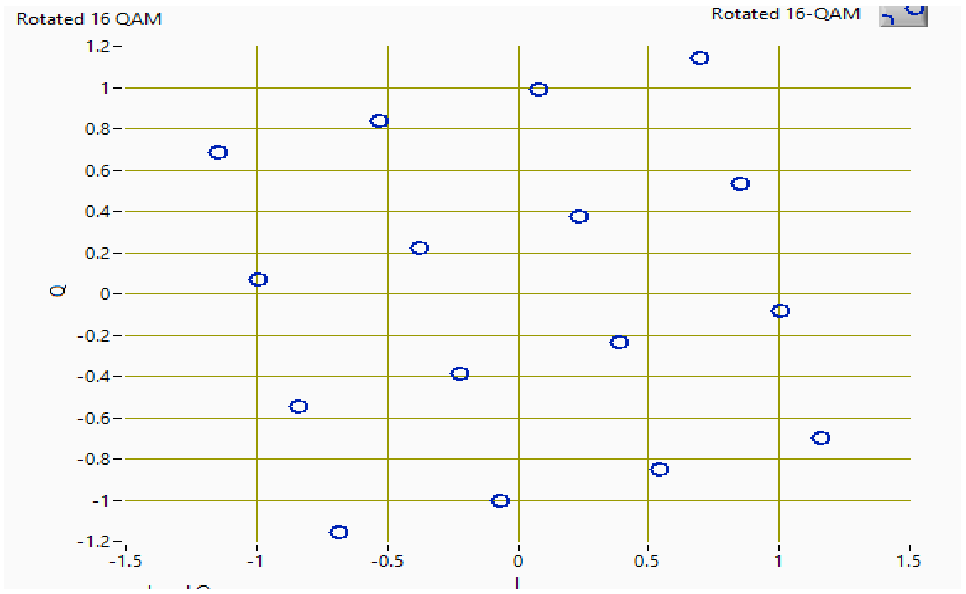


Figure 4-13 16QAM Rotated

To reduce algorithm and hardware complexity transformation is applied to rotated constellation the transformation parameters are constant for each constellation, these transformed points are placed in memories Sorted I and Sorted Q. The rotated point Z_M is divided by a constant distance between two consecutive points and a constant add offset converts the rotated point into rotated and transformed as shown in figure 4-13. With respect to the hardware limitations and complexity divide function is avoided so as distance is constant so multiplier is used by taking reciprocal of constant distance.

Let the received point in 16 QAM constellation is blue dot (10,14), then for Euclidean distance computation search region will be shown in figure 4-13 with overlapping region is the search space for received point. As 10 for I component lie in the dynamic region explained in equation 4 and 14 for Q lie in region 2. So overall there will be two search spaces one for I and one for Q and in total there will be eight points in both search spaces with respect to programming there will be two loops one for I and one for Q each will be running for four time. After traversing from eight point in regions in 16QAM each symbol will be recovered. In case of 16QAM there will be four LLRs for each symbol.

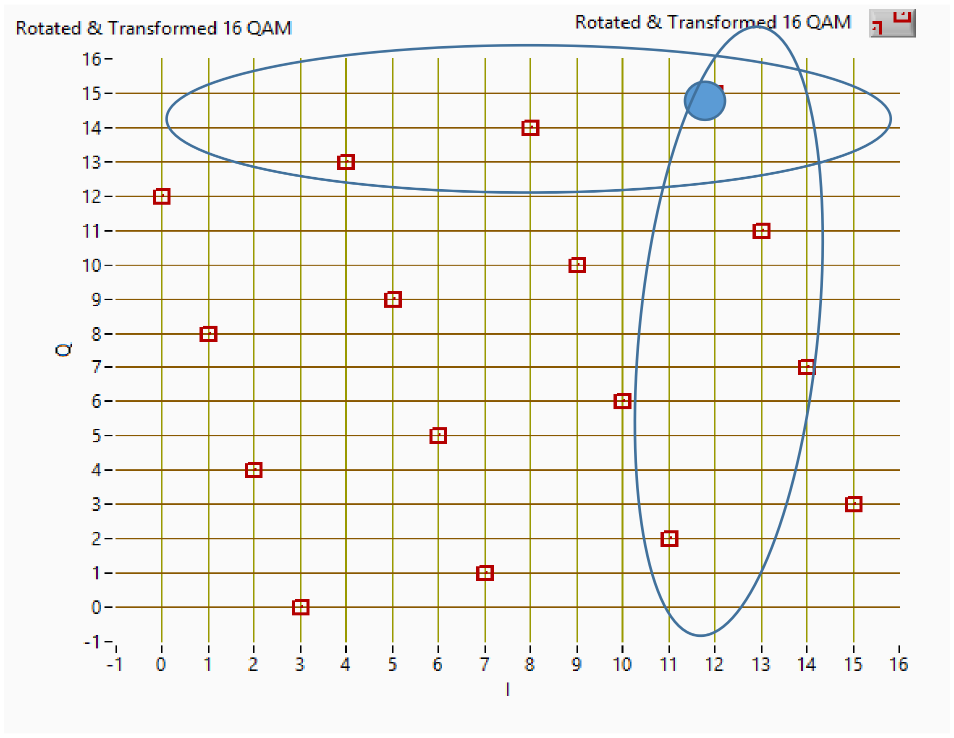


Figure 4-14 16QAM Rotated and Transformed

4.2.2.3. 64 QAM Rotation & Demapping Region

In 64-QAM modulation first of all on transmitter side constellation is gray coded mapped as shown in figure 14 using LUT and index of the LUT is symbol number in 64 QAM case the symbol number is represented in six bits. The LUT has 64 rows and 2 columns. First column of LUT contain I component while second column contains in gray coded style. In second step constellation needs to be normalized by factor $\frac{1}{\sqrt{42}}$. After normalizing 64-QAM constellation the normalized points are processed by equation 1&2 for RCQD. The rotation angle for 64-QAM constellation is 7.12^o computed by equation 2 in equation 1 as shown in figure 15. The main aim of this demapper is to design a low complex demapper to reduce complexity some functions and data types needs serious attention for example some constant numbers are used in division we can take reciprocal of constant and then use multiplier, also if there are fixed point negative and non-negative numbers in fractional format it will be more difficult to handle them with respect to overflow, wrap, saturate, rounding operations in hardware.

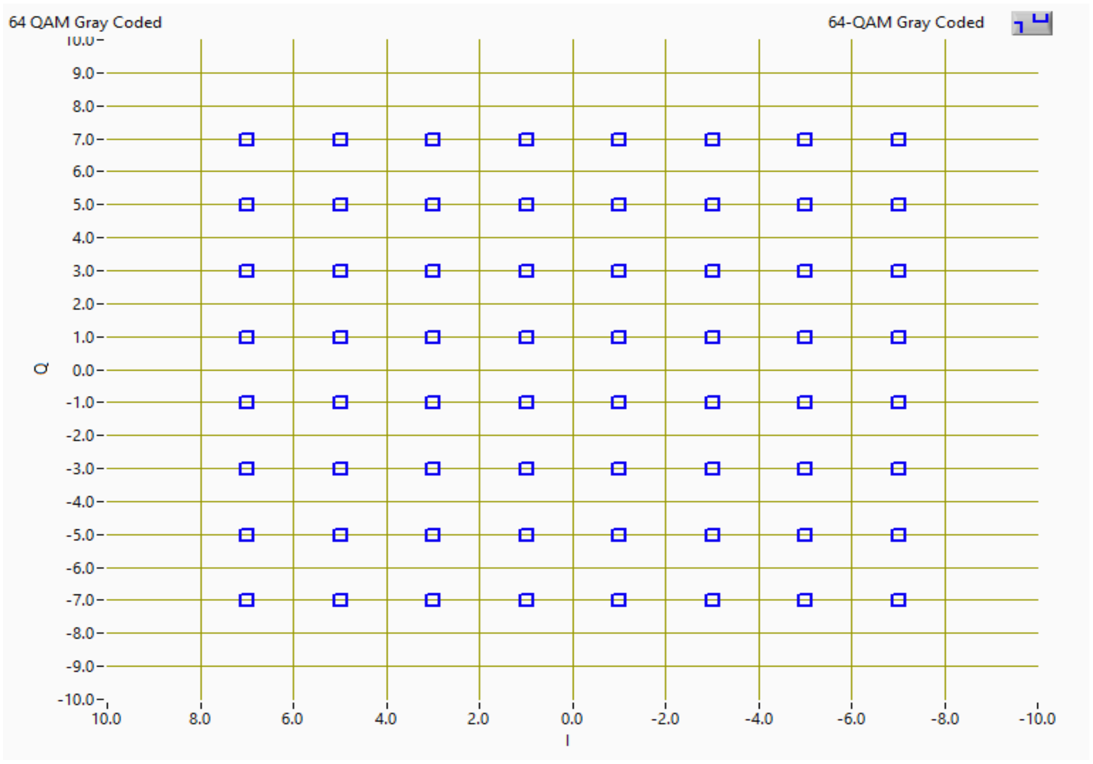


Figure 4-15 64QAM

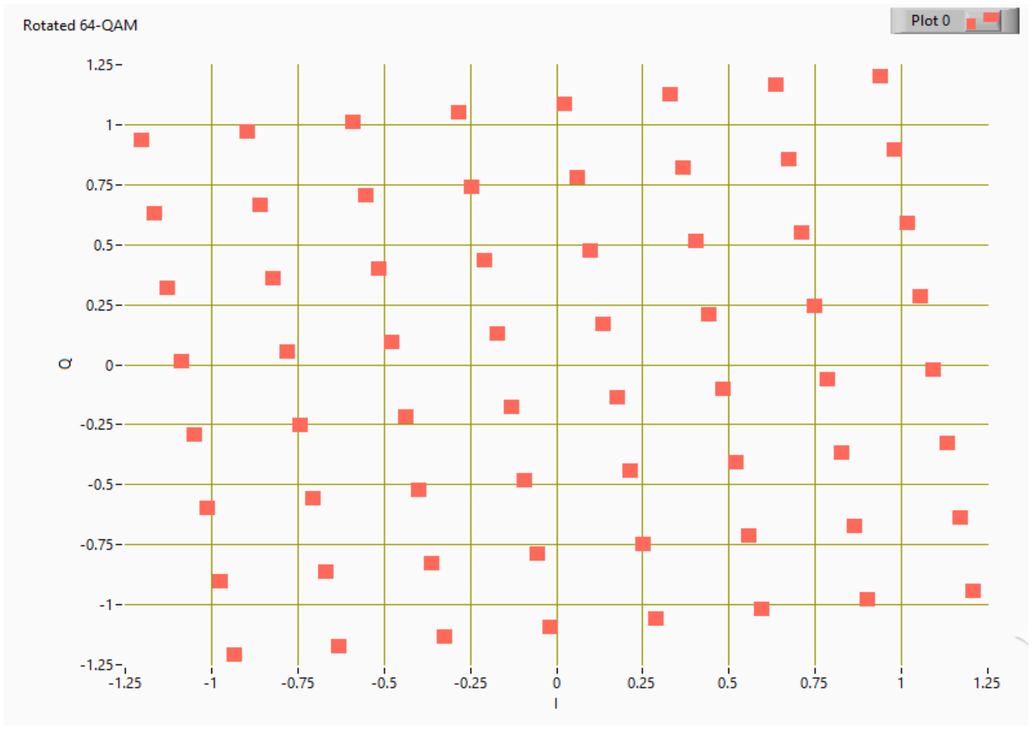


Figure 4-16 64QAM Rotated

So to overcome complexity transformation is needed, figure 4-15 shows all 64 points equally distributed in all four quadrants. In transformation all these 64 points transformed in to first quadrant in such way that last minimum point on I axis correspond to zero in first quadrant and maximum point on I axis correspond to 63 on I axis i.e from zero to M-1 for all constellation types, same case with Q component transformation.

This transformed copy is used at receiver side in memories sorted I and sorted Q to compute Euclidean distance with a received point as shown in figure 16 through channel. At receiver, received rotated point is transformed, then search regions computed among which Euclidean distance will be computed. Let’s take an example for 64-QAM region computation case in which point received is shown with “black dot” on points (14,32).

Applying equation 4 for search region computation for both I&Q the search region will be the dynamic regions having with floor function minus ‘d’ plus one. For I search points start from

11 to 18 while in case of Q the search points will start from 29 and will search up to 36 as shown in figure 16.

When the points received the counters of memories will be automatically adjusted to desired points for region computation the detailed implementation of memories and counter offsets will be discussed in next portion as this portion mainly focuses on region computation.

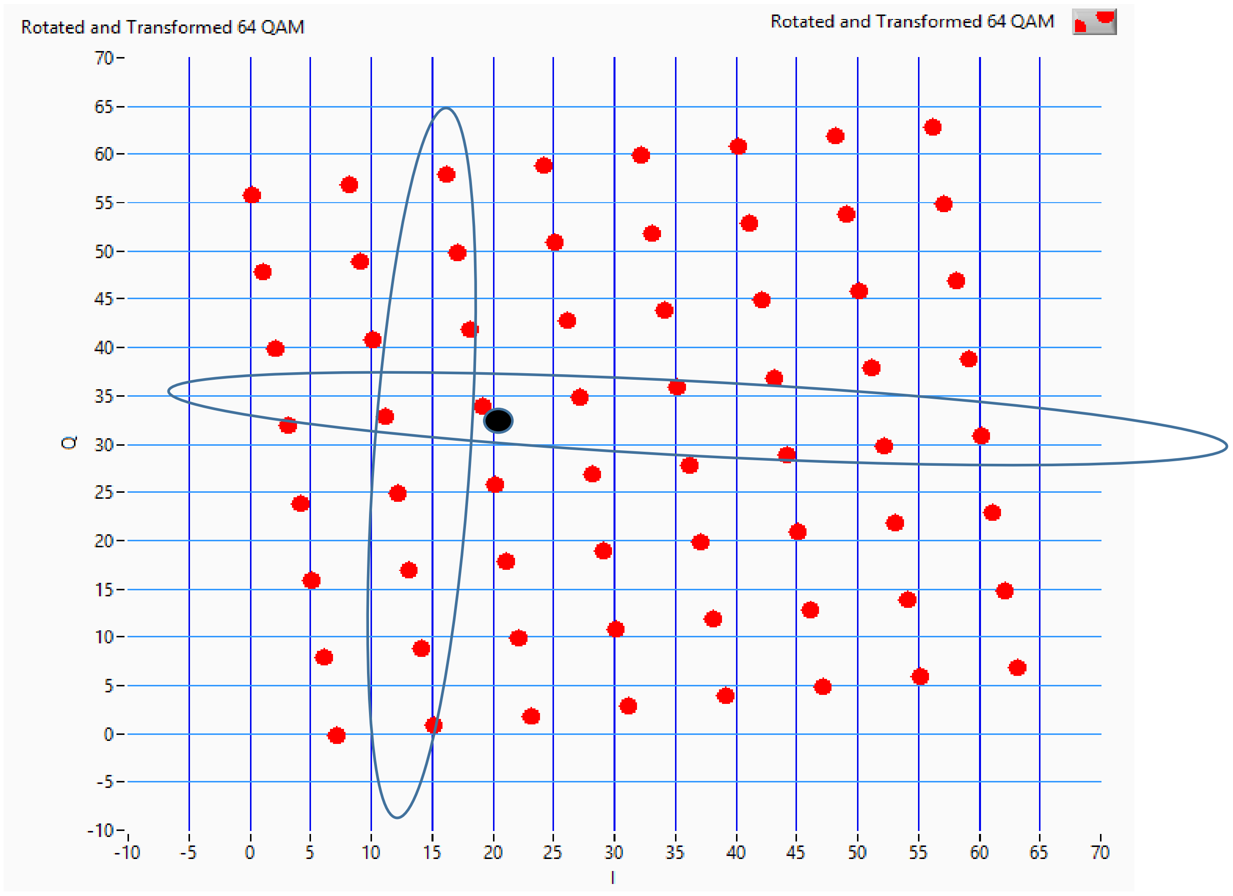


Figure 4-17 64 QAM Rotated and Transformed

4.2.2.4. 256 QAM Rotation and Demapping Region

In 256-QAM i.e 8 bits per symbol is the highest data rate supported by DVB-T2. First of all, on transmitter side the gray coded constellation is mapped in such a way that mapped points belong to LUT with having symbol numbers as index of the LUT as shown in figure 4-17 and these mapped points are then normalized. The normalizing factor for 256-QAM is $\frac{1}{\sqrt{170}}$ as shown in table 5. The gray coded constellation and rotated and normalized constellations as shown in figure 4-18 are equally distributed among four quadrants.

The constellation is then rotated by angle 3.57^{o} by equation in equation 1 as shown in figure 4-18.

In order to reduce hardware complexity and low complex search spaces among whole constellation transformation is applied to each constellation in such a way that the minimum and least index point in constellation correspond to 0 in first quadrant and maximum point and index in the constellation correspond to M-1 point in the first quadrant. Also transformation convert the all points to integer which gives easy access to the set counter and search in the LUT sorted I and sorted Q this portion will be explained in the hardware implementation portion in detail.

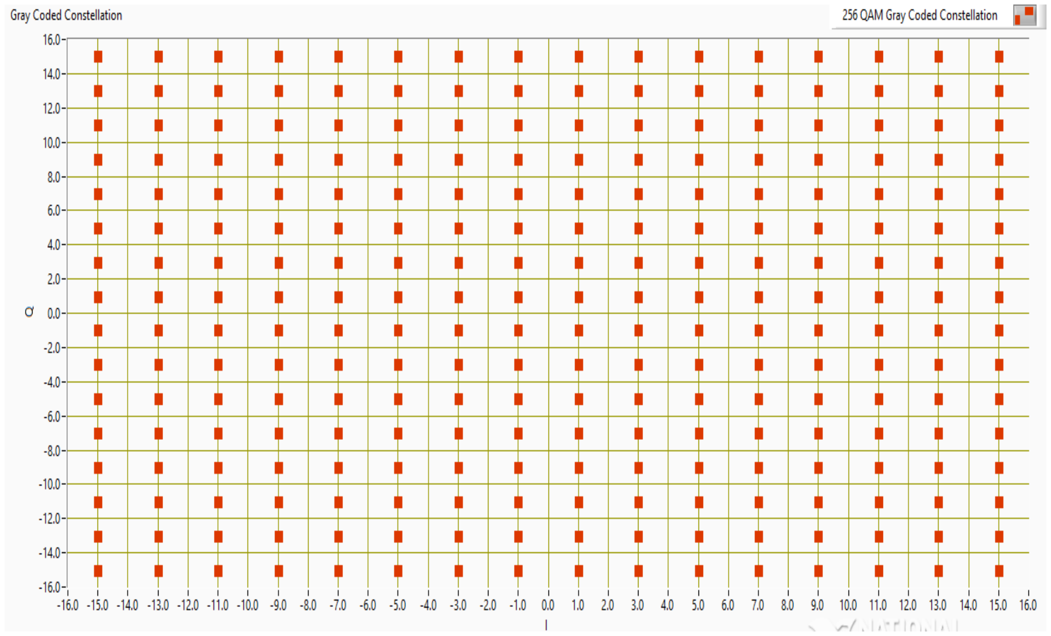


Figure 4-18

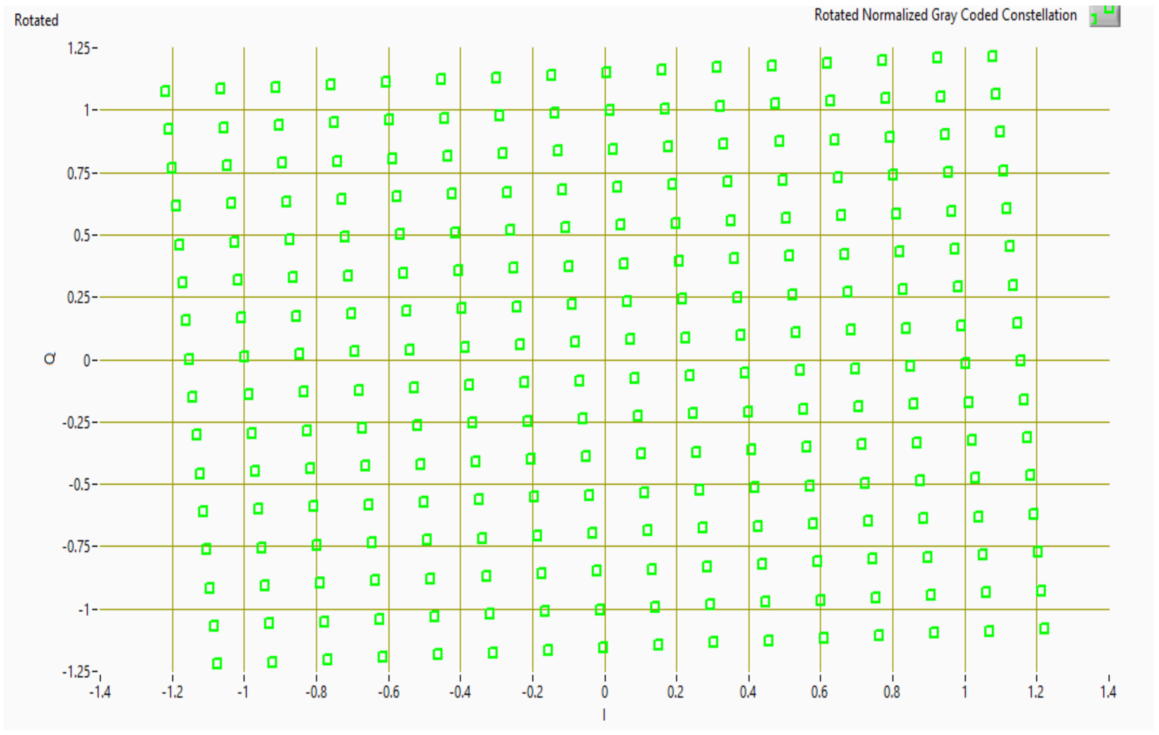


Figure 4-19 256 QAM Rotated

The point received through fading channel is used to compute the search region. Let the received point is shown with red dot at (14,1) in figure 19 is computed by processing with equation 8. The received point is transformed and is placed in figure 19 for region computation. As received point I value ‘1’ is in dynamic region and Q is in 1st region. The search space for received point on 256QAM is from 7 to 22 for I component i.e in dynamic region three for Q component 1 which is less than distance so the region for Q will be from 0 to 15. After receiving the point memory counter for search region will set from 0 to 15 for Q and from 7 to 22 for I.

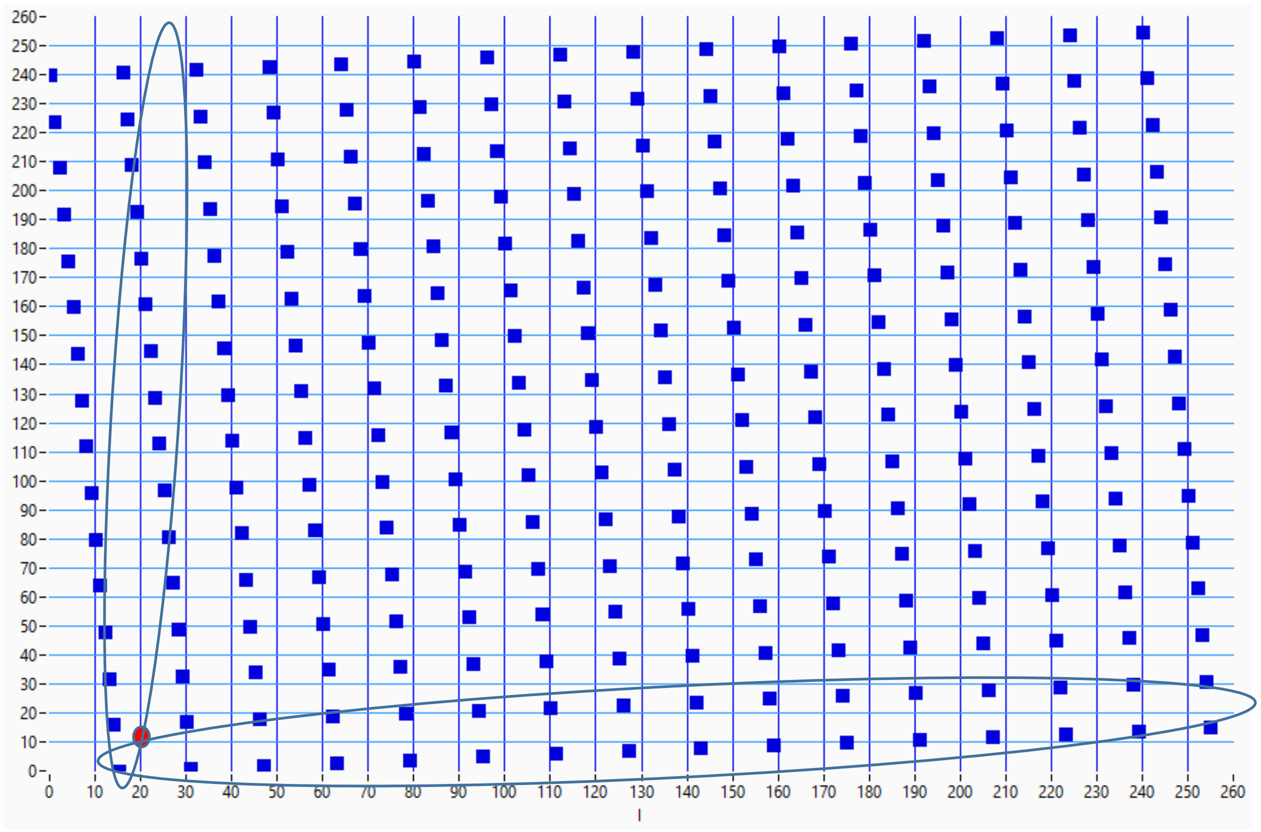


Figure 4-20 256 Rotated & Transformed

4.2.3. Block RAMs for Symbol Number, Sorted I and Sorted Q

In total three block rams are used in distance computation we know that symbol number corresponds to I and Q points. So the three representation of memories are of Symbol Number, Sorted I and Sorted Q in such a way that QPSK sorted I component start from location 0, 16QAM sorted I component start from location 8, 64 QAM sorted I component start from location 40 and 168 in case of 256 QAM.

In case of Q sorted component for distance search space QPSK start location is 4, in 16QAM 24 is start location, 104 is start location in 64 QAM and 424 is the start index of 256 QAM in sorted Q case.

In all Block Memories depth will be 680 of symbol number, I sorted and Q sorted. As these memory locations are filled with rotated and transformed values of all constellation as shown in above region figures. For every received point all these three memories have to be traversed with respect to the regions using memory start offset for each constellation.

The output of the region subvi is the search space start value which indicates that in constellation from which location traversing should start. This "start traversing value" adds with memory offset explained in above para give the final memory location from which search have to begin. Then there is a most outer loop which runs two time for each constellation one for I component and one for Q component. Inside the outermost loop a comparator checks that if the iteration value is equal to 1 then pass memory counter start value i.e region output plus sorted Q value inside the inner loop i.e for distance computation. Else use region output plus sorted I value as input to inner loop for memory location as shown in figure 21.

As from region figures of each constellation type it is clear that the distance points to be searched in each are as follows:

Sr. No	**Constellation Type**	**Points Used in Distance**

1	QPSK	4
2	16-QAM	8
3	64-QAM	16
4	256-QAM	32

Table 4-2

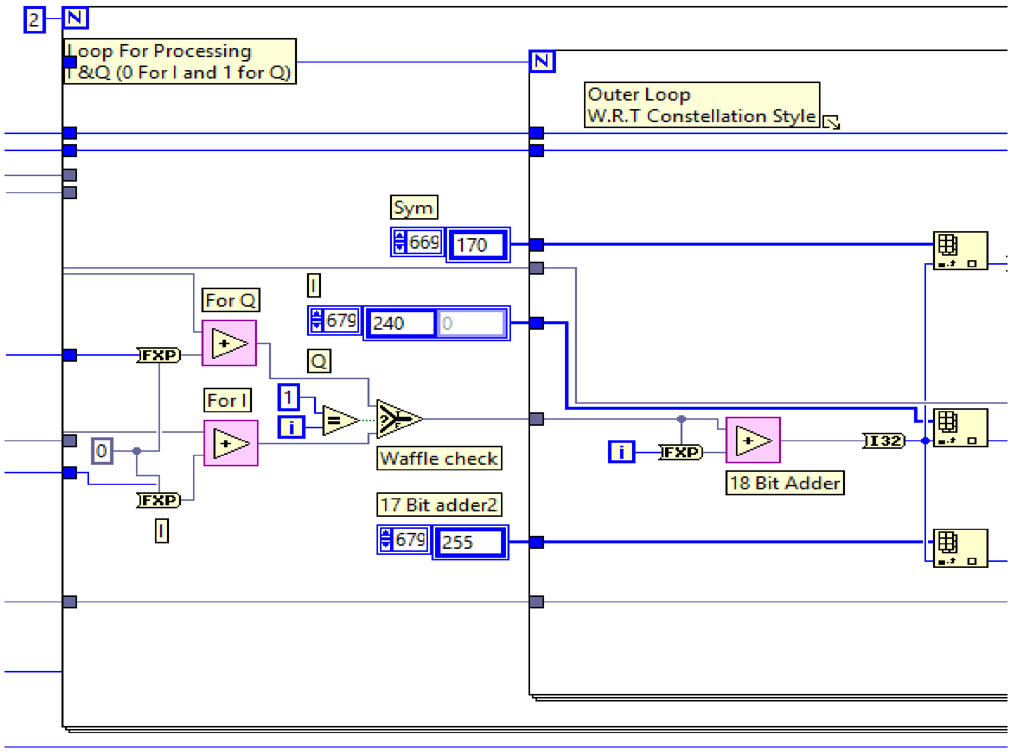


Figure 4-21 Memory Address Gen & Distance Computation

With the outer most loop which is for I and Q there is another loop inside whose iteration terminal is used as counter to read from block memories in such a way that region start value plus memory offset I and Q is the starting location for traverse and counts up depends on constellation selected using iteration of inner loop. Execution of both loops are as follows:

- For QPSK inner loop runs two times for I and two times for Q.
- For 16-QAM inner loop runs four times for I and four for Q

- For 64-QAM inner loop runs eight times for I and eight for Q
- For 256-QAM inner loop runs sixteen times for I and sixteen for Q

The inner most loop counter configured with control line when type of constellation is selected the counter index mapped to the block memories read the desired region points from memories (Symbol Number, Sorted I and Sorted Q). Where after fetching the symbol number it is converted to binary Boolean values e.g if symbol number is 255 gives 11111111.

4.2.4. Euclidean Distance and LLR Computation

Inside inner loop Euclidean distance has to be computed in the bit wise manner (Symbol Number converted to Boolean). Largest distance computed is represented is in eight bits and updated in two registers which were initialized with highest value will be updated with respect to incoming Boolean value 0 or 1 in such a way that if incoming Boolean value is 0 the register 0 will be updated and in case of incoming Boolean value 1 register 1 will be updated. The algorithm for minimum finder implemented in LabVIEW FPGA is given below and shown in figure 22:

```
Bit_Value   /// Symbol Number's Binary Value Bit Wise
Distance//  as shown in equation 5.
If (Bit_Value==0 )
If(Distance<Bit_0_Register[index])
Bit_0_Register[index]=Distance
Else Keep (Bit_0_Register[Index])
Else if(Bit_Value==1)
If(Bit_Value==1)
If(distance<Bit_1_Register[Index])
Bit_1_Register[Index]=Distance
Else Keep(Bit_1_Register[index])
```

At the end when registers are updated the log likely hood ratio(LLR) is computed by subtracting the 0 Bit registers and 1 Bit registers. The LLR decision is based on sign if sign is negative then 0 will be decoded and if LLR is positive 1 will be decoded and same information will be used as apriori information for further blocks.

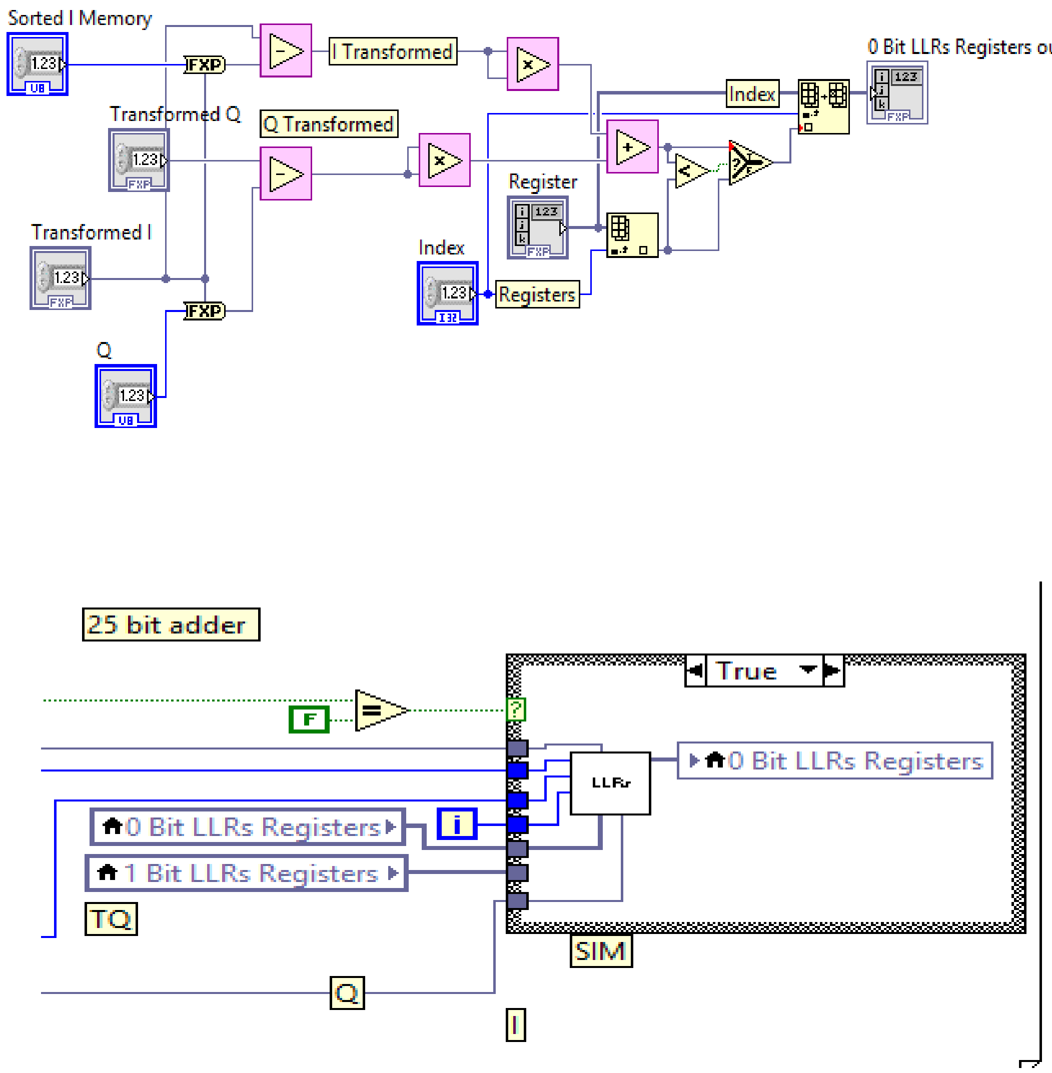


Figure 4-22 LV FPGA LLR Computation with Min Finder

This is a Minimum finder and Euclidean distance finder block implemented with a specific design, more optimized minimum finder and overall design with pipeline architecture increased throughput design using NI LabVIEW FPGA IP Builder with Vivado HLS is explained in next topic in which new directive were explored and a high throughput flexible demapper is implemented. While result of above explained demapper are shown in next section.

The arithmetic operation used in LabVIEW FPGA are:

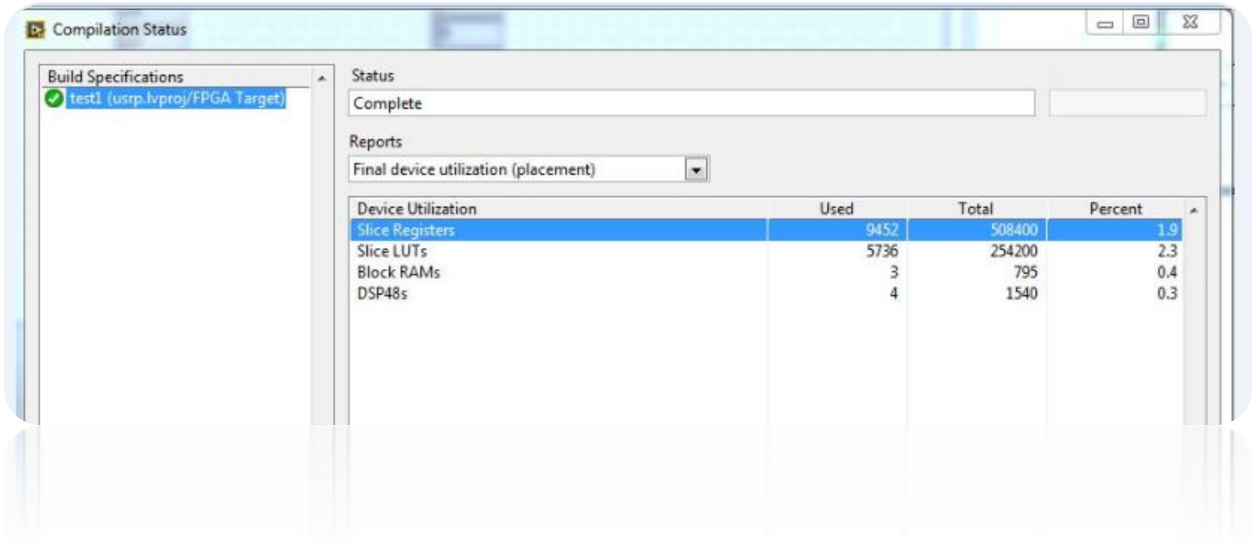
1. 2 Subtractors
2. 1 Adder
3. 2 Multipliers

4.2.5. Results

Before describing the results, the optimizing way of LabVIEW FPGA must be described. LabVIEW FPGA uses most optimized hardware architecture for the algorithm. For example, if we use a multiplier with two inputs. The values on two inputs are let's say at one input there is a variable input while second input is zero LabVIEW did not use any multiplier in this case LabVIEW do not use any DSP48E and same case if the multiplier's one input is 1 in this case by estimating the output LabVIEW do not use any DSP48E.

If the values wired as control, then LabVIEW will implement DSP48E as multiplier. In the code described in chapter 4 we cannot implement our pipeline stages because single cycle timed loop is not supported with nested loops as described in start of chapter 3 LabVIEW uses its enable chain on every operation as pipeline register to avoid enable chain it is necessary to use SCTL but SCTL do not support nested loops so, the code in LabVIEW FPGA is implemented with high throughput functions and all the synthesizing, optimization, device estimation, timing estimation, placing and routing is done by LabVIEW FPGA itself.

In result LabVIEW FPGA compiles the code at 248 MHz with 4 DSP48Es i.e 0.3% Block RAM 0.4% and slice register 1.9% all these results are with Rho value equal to 1 as shown in figure below.


Compilation Status

Build Specifications

test1 (usrp.lvproj/FPGA Target)

Status: Complete

Reports: Final device utilization (placement)

Device Utilization	Used	Total	Percent
Slice Registers	9452	508400	1.9
Slice LUTs	5736	254200	2.3
Block RAMs	3	795	0.4
DSP48s	4	1540	0.3

Figure 4-23 LV FPGA Resources

In these results 3 block rams corresponds to memories used for symbol number, sorted I and sorted Q. Where 4 DESP48E corresponds to 2 multipliers used in transformation function 1 in I component transformation and 1 in Q component transformation and 2 multipliers which are used as square for difference between YI and TI and YQ and TQ. Rho is assumed as 1.

This demapper is designed to take received symbols automatically while constellation type is mapped to control line whatever constellation type is on control line it will configure all registers, counters, memories and loops automatically and generates LLRs for QPSK it took 3 cycle, 16-QAM 6 cycle, 64-QAM 12 cycle, 256-QAM 24 cycle for the output at clock 248 MHz

Modulation Type	**Clock Cycle required to compute 1 LLR**	**Clock Rate**
QPSK	3	248MHz
16-QAM	6	248MHz
64-QAM	12	248MHz

256-QAM	24	248MHz

4.3. Proposed Demapping with LabVIEW FPGA IP Builder with Vivado HLS

LabVIEW FPGA module enables your LabVIEW skills to program FPGA with LabVIEW skills. In previous section we learn some programming concepts of LabVIEW FPGA and to model, design and implement a demapper in detail with constellation plots of all supported constellations in DVB-T2. There are some limitations or we can say algorithm constraints that we observed while using LabVIEW FPGA module which limits our throughput in the design like we can’t use SCTL for whole code as there are nested loops in the algorithms so in LV FPGA module we can’t use SCTL to avoid enable chain to insert our pipeline stages and latency. This constraint is better handled using IP builder.

Moreover, there are number of directives of Vivado HLS are available to optimize and increase throughput design. Directive setting is the iterative process in FPGA IP builder in both Vivado and LV FPGA as shown in figure 23. First define your interface using icon/connector pane from front panel. Each directive defines hardware approach and throughput of the algorithm. If required architecture and throughput not achieved, then go to previous step etc. The main benefit of using FPGA IP one can use that IP inside SCTL in LabVIEW FPGA module. Also when one builds IP HLS gives exact timing and device utilization report instead in LabVIEW FPGA synthesis report has more things than we used because in LV FPGA module is directly connected to I/O.

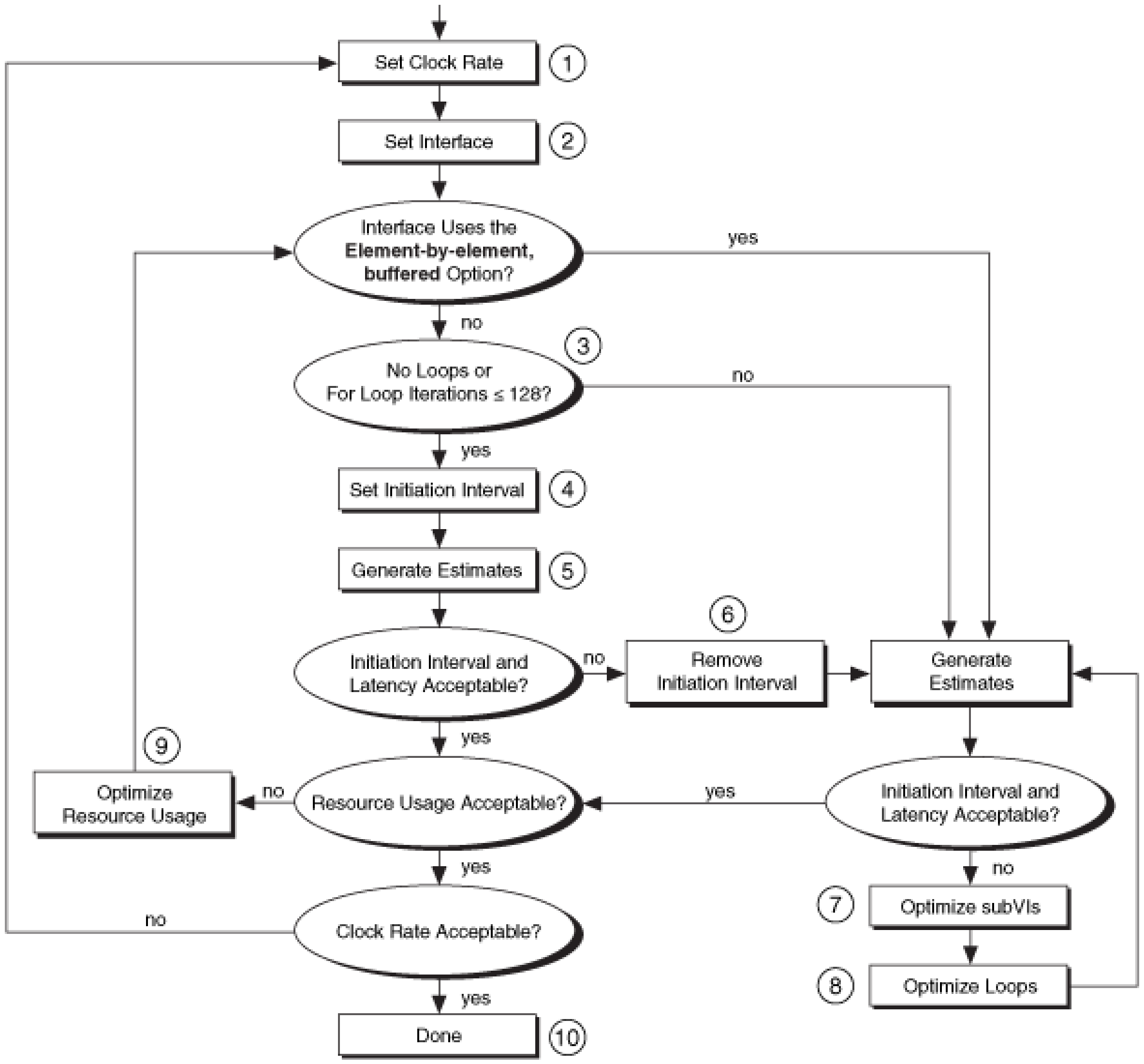

Set Clock Rate
1
Set Interface
2
Interface Uses the Element-by-element, buffered Option?
yes
no
3
No Loops or For Loop Iterations ≤ 128?
no
yes
Set Initiation Interval
4
Generate Estimates
5
6
Initiation Interval and Latency Acceptable?
no
Remove Initiation Interval
Generate Estimates
yes
9
Optimize Resource Usage
no
Resource Usage Acceptable?
yes
Initiation Interval and Latency Acceptable?
yes
no
7
Optimize subVIs
no
Clock Rate Acceptable?
8
Optimize Loops
yes
Done
10

Figure 4-24 Directives Setting Process [13]

Some common directives shown in flow chart and also used in designing the demapper are as flows:

Initiation Interval: in top level vi is used to define how frequently your hardware take next input or you can say the delay in clock cycles between two inputs. By default, if it is not configured HLS will assign any clock cycles but if it is set to 1 then HLS aggressively design the hardware architecture in a manner so that it must take next input on next clock cycle.

Clock Rate: whenever LabVIEW FPGA project is created there is a step when you choose an FPGA target in our case we use NI-USRP RIO as shown in experimental setup figure. In USRP which we use the maximum supported clock is 500MHz. when configuring the directives set the maximum clock rate after the quick estimates HLS will tell either that requested clock rate met or not if not then why not? Where is the critical path?

Interface: define the interface of your design's inputs and outputs in connector pane first. Then in directive portion if input is single scalar then directive will be "Data" in case of array it is required to set "Buffered Element by Element" or" Unbuffered" these directives will increase or decrease the throughput of design.

Unroll: the most interesting directive in HLS in loops is unroll. By default, loop unroll is unchecked which uses the code as hardware inside loop for no of time loop runs. When loop unroll is checked, one must have to specify the unroll factor of loop, loop can be fully unrolled or partially unrolled. In case of fully unroll the no of unroll factor must be equals to no of loop iterations. This complete unroll will make no of copies of hardware equals to the loop iteration this will increase the hardware area also increase parallelism.

After creating LabVIEW FPGA project on NI-USRP RIO target right click on IP Builder create new VI. This new VI is on LabVIEW FPGA module with IP Builder settings. On block diagram in IP Builder only very basic functions are available by using them one can build all advance level functions which are available in LV FPGA module.

Transformation subvi is built in IP builder using same algorithm in C modelling and LV FPGA keeping its execution mode on reentrant pre allocated shared clone. The transformation is configured by control line which selects type of constellation. Where output of transformation is the input to region computation block as shown in figure4-24.

The **region computation** block take input transformed Y_I and Y_Q takes decision on search region with respect to these points and generate output the start index from where the search begun. The execution mode of region subvi is also reentrant pre allocated shared resource the subvi in hardware is placed in such a way that comparison values are passed with respect to type of constellation using MUX in hardware as shown in figure 4-24.

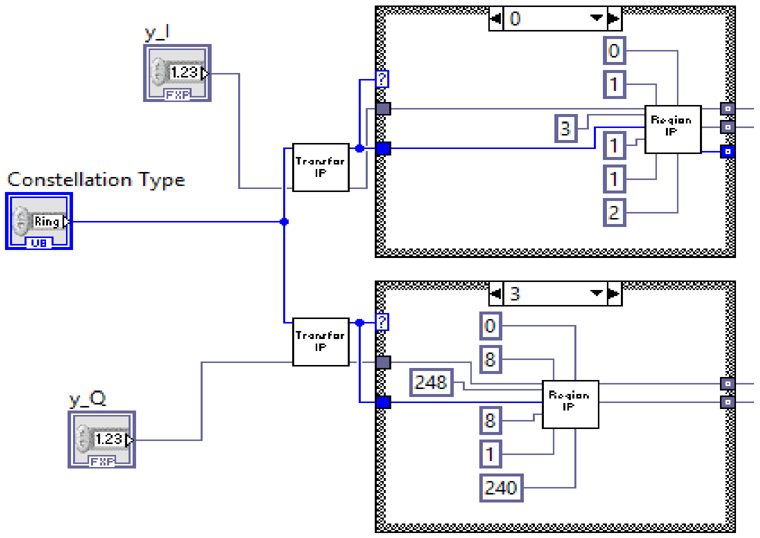


Figure 4-25 IP Transformation and Region

The **Euclidean distance finder** using IP builder has 4 multiplier using DSP48Es in this we are using Rho with some values to multiply with difference in both cases for I and Q. This Euclidean distance differ from explained in section 3.2.4 in this we use 8 parallel minimum finder which take 1 clock cycle to generate output. This EU distance finder block diagram is shown below in figure 4-25.

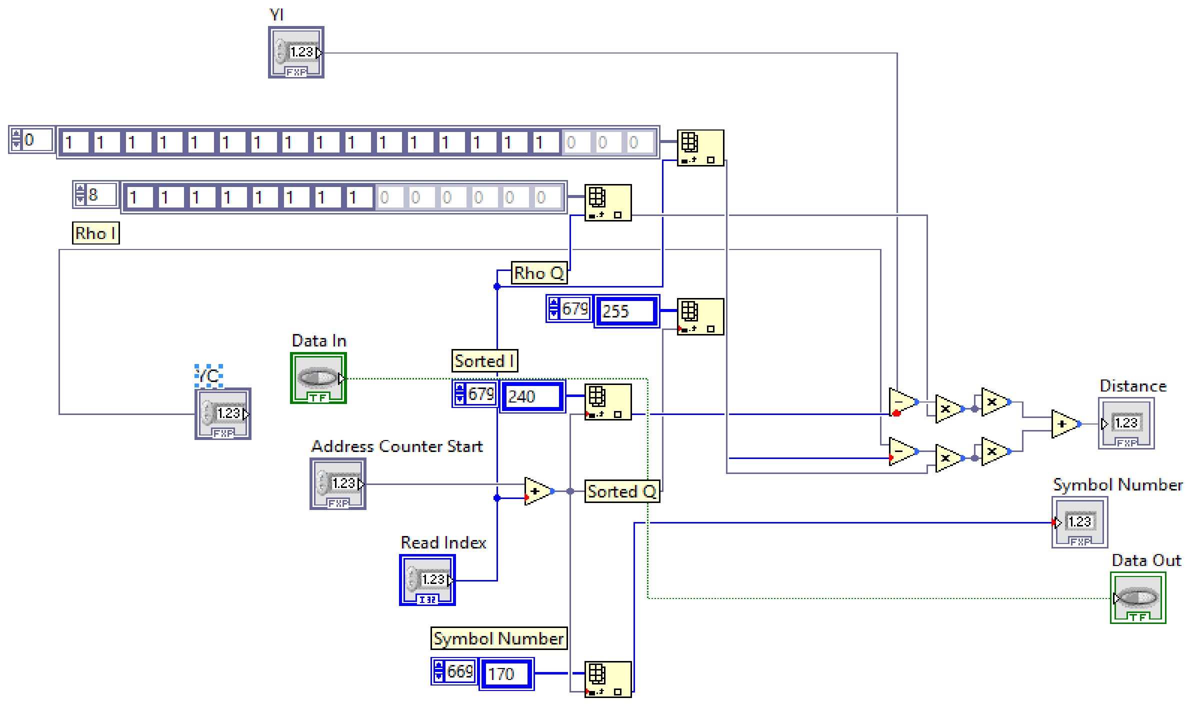


Figure 4-26 IP Distance Computation

This distance computed shown in figure and symbol number will be further used in minimum finder. In our case we implement 8 parallel minimum finder all these will execute in 1 clock cycle this technique will be implemented using loop unroll technique to unroll minimum finder in such a way that for 8 LLRs it will automatically implement 8 parallel min finders. Which is transformed in LabVIEW FPGA code for IP Builder as shown in below figure applying loop unroll directive to make 8 parallel min finders

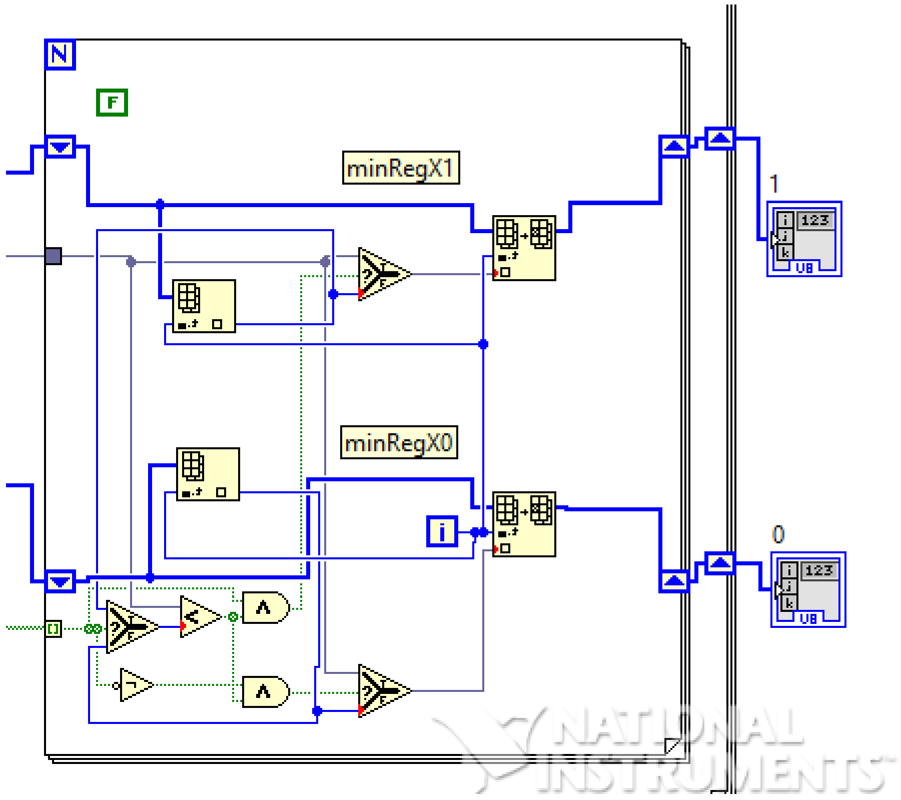


Figure 4-27 IP Minimum Finder

The throughput of any architecture is given by [13]:

$$Throughput = \frac{Clock\ Rate\ x\ Sample\ Per\ iteration}{Initiation\ Interval} \quad ,9$$

4.3.1. Benefits of Using LabVIEW FPGA

LabVIEW and LabVIEW FPGA is a graphical programming language which uses all concepts of C and VHDL. Benefit of using LabVIEW FPGA is to achieve rapid prototyping and deployment as it gives drag and drop environment and requires user configuration to configure different FPGA mathematics operations. Also almost all major RIO devices with I/Os are easily interfaced with it giving Target and Host access like both target and host can communicate and share data easily during run time. User can monitor FPGA acquired and processed data on user interface in both

FPGA target and host. Also one can directly model any algorithm in LabVIEW and use same code in FPGA target same file will be used as LabVIEW FPGA for all types of target. Also LabVIEW FPGA uses all concepts and directives of Verilog, VHDL and Vivado HLS in one tool, LabVIEW automatically introduces enable chain and pipeline register to reduce critical path although one can implement his own pipeline stages, LabVIEW can auto handle Bit growth, Wrap, Saturation, overflow etc. type operation in FXP arithmetic. User of any environment can quickly develop any architecture on target using LabVIEW FPGA.

4.3.2. Results

At compilation LV FPGA gives more optimized results with respect to both device and timing utilization and placing and routing as shown in following table.

Constellation Type	**DSP48Es**	**Slice LUTs**	**Slice Registers**	**Initiation Interval**	**Latency**	**Clock rate**	**Throughput**	**Pipeline Type**
QPSK	6 (0.4%)	0.5%	0.3%	1	43	418.41MHz	836.82MLLR/Sec	Full
16QAM	6 (0.4%)	0.6%	0.4%	1	44	418.4MHz	1673.6MLLR/Sec	Full
64QAM	6 (0.4%)	0.5%	0.3%	1	44	418.4MHz	2510.4MLLR/Sec	Full
256QAM	6 (0.4%)	0.5%	0.3%	1	45	418.4MHz	3347.2MLLR/Sec	Full
256QAM (DVB-T2)	6 (0.4%)	0.6%	0.2%	33	33	250.63MHz	60.7MLLR/Sec	Partially

Table 4-3

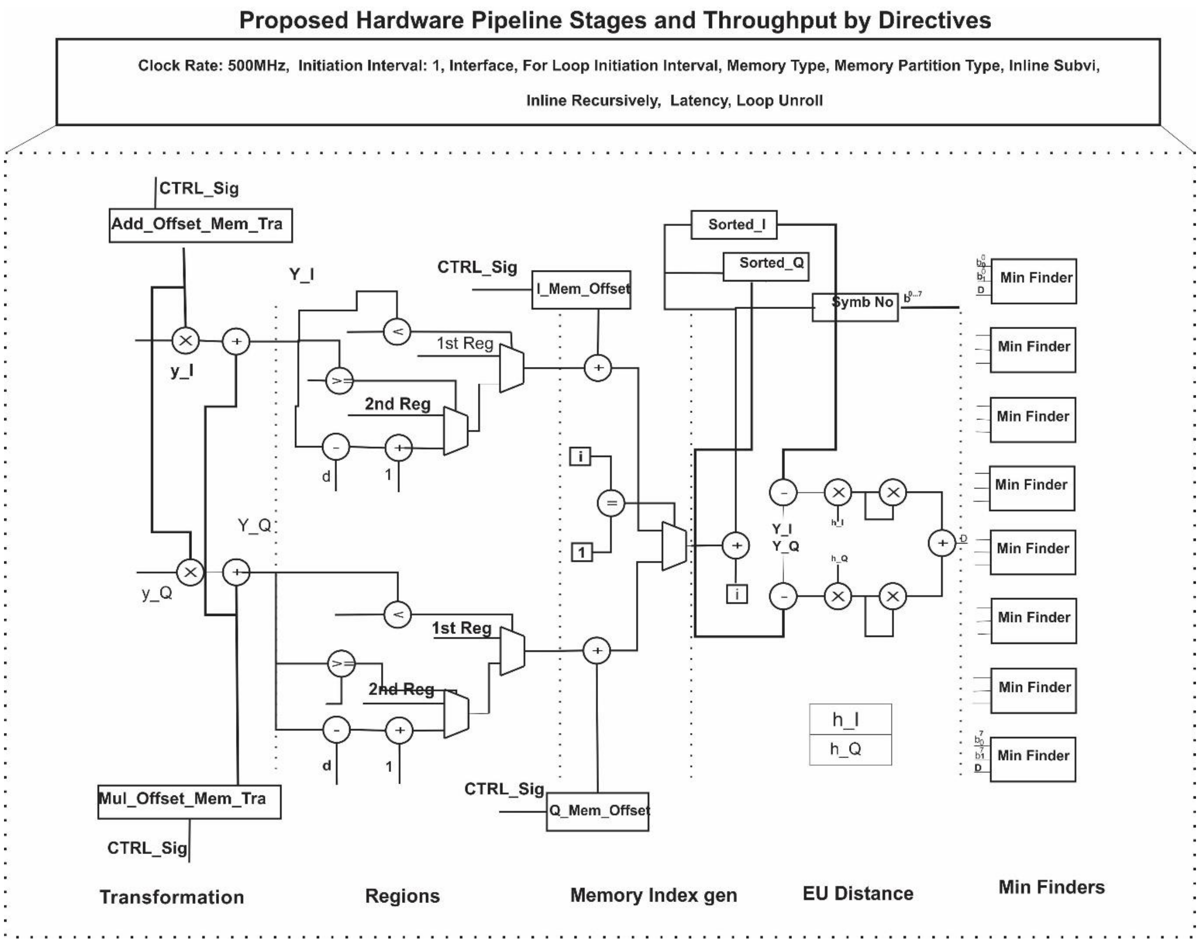
So the overall throughput of the system increases as the bits'/symbol increase because initiation interval set to 1 in directives. The snapshot compilation results for all four constellation type are in Appendix.

Arithmetic operations/memories/counters involved in demapper are:

	Additions	**Subtractions**	**Multiplications**	**Memories**	**Counters**	**Total**
Transformation	2	0	2	2	0	**6**
Region	2	2	0	0	0	**4**
Memory Address Generation	2	0	0	2	0	**2**
Distance Finder	2	2	4	3	1	**12**
Total	**8**	**4**	**6**	**7**	**1**	

Table 4-4

4.3.2.1. Proposed Architecture



Proposed Hardware Pipeline Stages and Throughput by Directives
Clock Rate: 500MHz, Initiation Interval: 1, Interface, For Loop Initiation Interval, Memory Type, Memory Partition Type, Inline Subvi, Inline Recursively, Latency, Loop Unroll
CTRL_Sig
Add_Offset_Mem_Tra
Y_I
y_I
Y_Q
y_Q
1st Reg
2nd Reg
d
1
I_Mem_Offset
Q_Mem_Offset
Mul_Offset_Mem_Tra
Sorted_I
Sorted_Q
Symb No
h_I
h_Q
Min Finder
Transformation
Regions
Memory Index gen
EU Distance
Min Finders

Figure 4-28 Low Complex Proposed Architecture

The number of pipeline stages are configured by latency directive before synthesis and initiation interval i.e how frequently the architecture takes next input set to every 1 clock cycle in HLS directive to achieve maximum throughput and auto pipeline architecture.

For required DVB-T2 throughput directive set explained in table 4-3 are applied to same design to achieve the required throughput.

4.4. LabVIEW, LabVIEW FPGA development and LabVIEW FPGA IP Builder Comparison

Initially the demapper model developed in LabVIEW using fixed point on development computer model explained in topic 4.2 as the tool provide rapid development but the main aim is to prototype a demapper on FPGA for this purpose NIUSRP RIO is selected with LabVIEW FPGA toolkit which provides utility to program FPGA using graphical programming in high throughput math functions with FPGA concepts and shows hardware resources and timings results shown in topic 4.2.5 of same model implemented in LabVIEW. These results are not much optimized due to some limitations as LabVIEW FPGA uses enable chain after each operation. In LabVIEW FPGA one can implement known pipeline stages by avoiding enable chain using single cycle timed loop (SCTL).[12] SCTL can be configured to run at any desired frequency supported by FPGA target. In our demapper limitation is that SCTL do not support nested loops and in our algorithm there are two loops one for I and Q where second is for EU distance. Also our main aim was to design a block which is independent of target can be used in any target. Results obtained by LV FPGA are shown in 4.2.5. By considering all these constraints same code is imported in LabVIEW FPGA IP Builder under LabVIEW FPGA project which uses Vivado HLS for synthesis. In this implementation Transformation, Region, EU distance blocks are converted in subvi as functions for optimization using directives for each block.[13] IP Builder implements different optimization technique like, Loop unroll, Interface for an IP, Pipeline, latency, Initiation Interval etc. So the limitation discussed are better handled in this approach for optimized results. The developed IP for the demapper with interface is now accessible in LabVIEW FPGA it can also be implemented now inside SCTL as a whole demapper block [13]. Final results obtained by this approach are shown in 4.3.2.

4.5. Comparison state of the art of some Demappers

The comparison is based on already proposed demapper in the literature and demapper designed by our development flow. [8] worked on DVB-T2 demapper and achieved throughput of 124MLLR/Sec with SSD whereas [4] design a universal soft demapper and achieved 35MLLR/Sec with SSD and 93.5MLLR/Sec with Gray coded style their results are compared with our demapper shown in table 4-5.

	Ref [8]	**[14]**	**Ref [4]**	**Our Demapper**	
Wireless Standard	DVB-T2	DVB-T2	Wifi, Wimax, LTE, UMB, DVB-SH, T2,S2	DVB-T2	
Frequency	62MHz	96MHz	156MHz	**For Max. Throughput**	**For required DVB-T2 Throughput**
				418.41MHz	250.63MHz
Throughput	**64-QAM**	**256-QAM**	**64-QAM**	**256QAM**	**256QAM**
	- 124MLLR/Sec (SSD)	- 96MLLR/Sec	93.6 MLLR/Sec (Gray) 35MLLR/Sec (SSD)	- 3347.2MLLR/Sec(SSD)	- 60.7MLLR/Sec(SSD)
Area					

• Slice Registers • Slice LUTs • DSP48Es • BRAM	791 4667 20 0	7637 32764 16	1596 2,627 6 8	1714 1258 6 5	981 1499 6 5

Table 4-5

Conclusion

CHAPTER 5. CONCLUSION

This thesis aims at study and implementation of rotated constellations and their demapper in general and in DVB-T2 standard specially. DVB-T2 demapper was studied in detail which shows that it has gray coded mapping style with some rotation angles. Rotation in constellation increase hardware complexity specially in case of higher order constellation. There are different demapping techniques proposed in literature like ML demapping, demapping for gray coded constellation, sphere demapping etc. Each technique has its own complexity with respect to algorithm and hardware both like ML demapping requires 2^m computations to compute one LLR. Also different architectures for demapper were studied some architectures are ASIP based which is used in both iterative and non-iterative manner. One architecture presented in literature is NISC based presents that in ASIP approach instruction fetch and decode is overhead.

Our work contributes in implementing and designing the optimized architecture for DVB-T2 demapper which has low complexity and high throughput. For this, algorithm explained in [2] studied and then modelled in C have verifies the results for QPSK to 256-QAM modulation also software modelling gives basic estimation of all arithmetic operations, memories and counters used in the design. Second modelling is done with LabVIEW FPGA using fixed point data type instead of double with high throughput arithmetic functions the concepts and results gathered in C modelling are implemented in LV FPGA modelling. After modelling design is first run on development computer then after verifying LLR results same code is used on FPGA target by just adding the file in FPGA project the synthesis results show 24 clock cycles for each LLR for 256-QAM case at 248MHz clock rate. Although this design is auto optimized by LabVIEW FPGA but throughput limits by certain constraints like known pipeline stages are not implemented etc. For most optimized hardware with maximum throughput same code is used in LV FPGA IP Builder.

IP Builder uses Vivado HLS for synthesis. Same code as implemented in LV FPGA IP Builder is used with little modifications in such way that transformation for I&Q placed parallel, regions for I and Q placed parallel and both these configured in a Subvi manner such that they take next input after every 1 clock cycle. After region computation next blocks (Memory Offset Gen. and Distance Computation) executes in loops in such a way that loop uses same hardware for N times. While in case of Minimum finder for loop is configured in such a way that its unroll factor is 8 and loop unroll implements 8 minimum finder hardware's in parallel because in each clock cycle when distance is computed with each symbol number, this distance and symbol number will be same for minimum finders so we can execute 8 parallel minimum finder.

The architecture is explained in above section granted clock rate of 418.41MHz as we requested 500MHz. In IP Builder code optimization and throughput is done with directive settings in our case:

Directive	**Requested**	**Granted**	**Requested (DVB-T2)**	**Granted (DVB-T2)**
Clock rate	500MHz	418.41MHz	200MHz	250.63MHz
Initiation Interval	1 Clock Cycle	1 Clock Cycle	33 Clock Cycle	33 Clock Cycle
Interface	3 Inputs , 2 Outputs	3 Inputs, 2 Outputs	3 Inputs , 2 Outputs	3 Inputs , 2 Outputs
Inline recursively	True	true	true	true
Memory Type	Block ROM	Block ROM	Block ROM	Block ROM
Latency	-	Auto-Granted 44	33	33

Table 5-1

So the directives are set in a manner that for every higher order modulation scheme throughput will be increased because initiation interval granted is always 1.

REFERENCES

[1] E. E. Standard, "Digital Video Broadcasting (DVB); Frame Structure Channel Coding and Modulation for Second Generation digital terrestrial television broadcasting system (DVB-T2)," *ETSI EN 302 755V1.4.1*, 2015 2015.

[2] J. Yang, K. Wan, B. Geller, C. Abdel Nour, O. Rioul, and C. Douillard, "A low-complexity 2D signal space diversity solution for future broadcasting systems," in *Communications (ICC), 2015 IEEE International Conference on*, 2015, pp. 2762-2767.

[3] A. R. Jafri, A. Baghdadi, and M. Jézéquel, "ASIP-based universal demapper for multiwireless standards," *Embedded Systems Letters, IEEE,* vol. 1, pp. 9-13, 2009.

[4] A. R. Jafri, A. Baghdadi, and M. Jézéquel, "Rapid design and prototyping of universal soft demapper," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010, pp. 3769-3772.

[5] M. Rizk, A. Baghdadi, M. Jézéquel, Y. Mohanna, and Y. Atat, "NISC-Based Soft-Input–Soft-Output Demapper," *Circuits and Systems II: Express Briefs, IEEE Transactions on,* vol. 62, pp. 1098-1102, 2015.

[6] H. A. Fahmy, S. Gasser, and K. A. Shehata, "Combining Cyclic Q Delay and Cell Interleaver for Enhanced Performance DVB-T2 System," in *The International Conference on Digital Information, Networking, and Wireless Communications (DINWC2014)*, 2014, pp. 98-102.

[7] Altera, "Constellation Mapper and

Demapper for WiMAX," 2007.

[8] M. Li, C. A. Nour, C. Jego, and C. Douillard, "Design of rotated QAM mapper/demapper for the DVB-T2 standard," in *Signal Processing Systems, 2009. SiPS 2009. IEEE Workshop on*, 2009, pp. 018-023.

[9] J. W. Park, M. H. Sunwoo, P. S. Kim, and D.-I. Chang, "Low complexity soft-decision demapper for high order modulation of DVB-S2 system," in *SoC Design Conference, 2008. ISOCC'08. International*, 2008, pp. II-37-II-40.

[10] N. M. Bahgat, D. S. Khalil, and S. H. El-Ramly, "Energy efficient design of DVB-T2 constellation demapper," in *Quality Electronic Design (ISQED), 2015 16th International Symposium on*, 2015, pp. 197-200.

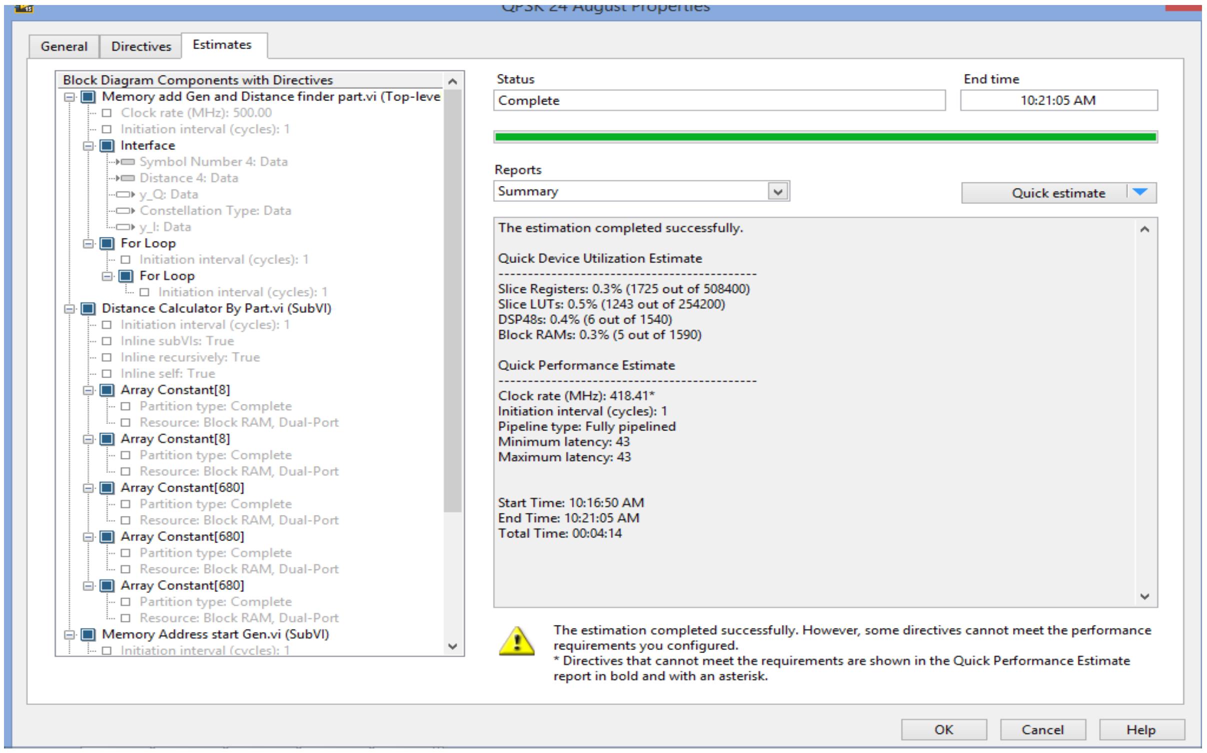
[11] S. Chen, K. Peng, and F. Yang, "Simplified universal soft demapper for gray-mapped constellation," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2015 International*, 2015, pp. 857-861.

[12] N. Instruments, "LabVIEW FPGA Concepys Manual," 2013.

[13] N. Instruments, "Using NI LabVIEW FPGA IP Builder to Optimize and Port VIs for Use on FPGAS," 2014.

[14] J. Yang, M. Li, M. Li, C. A. Nour, C. Douillard, and B. Geller, "Max-log demapper architecture design for DVB-T2 rotated QAM constellations," in *Signal Processing Systems (SiPS), 2015 IEEE Workshop on*, 2015, pp. 1-6.

APPENDIX



```
Block Diagram Components with Directives
Memory add Gen and Distance finder part.vi (Top-leve
    Clock rate (MHz): 500.00
    Initiation interval (cycles): 1
    Interface
        Symbol Number 4: Data
        Distance 4: Data
        y_Q: Data
        Constellation Type: Data
        y_I: Data
    For Loop
        Initiation interval (cycles): 1
        For Loop
            Initiation interval (cycles): 1
Distance Calculator By Part.vi (SubVI)
    Initiation interval (cycles): 1
    Inline subVIs: True
    Inline recursively: True
    Inline self: True
    Array Constant[8]
        Partition type: Complete
        Resource: Block RAM, Dual-Port
    Array Constant[8]
        Partition type: Complete
        Resource: Block RAM, Dual-Port
    Array Constant[680]
        Partition type: Complete
        Resource: Block RAM, Dual-Port
    Array Constant[680]
        Partition type: Complete
        Resource: Block RAM, Dual-Port
    Array Constant[680]
        Partition type: Complete
        Resource: Block RAM, Dual-Port
Memory Address start Gen.vi (SubVI)
    Initiation interval (cycles): 1
```

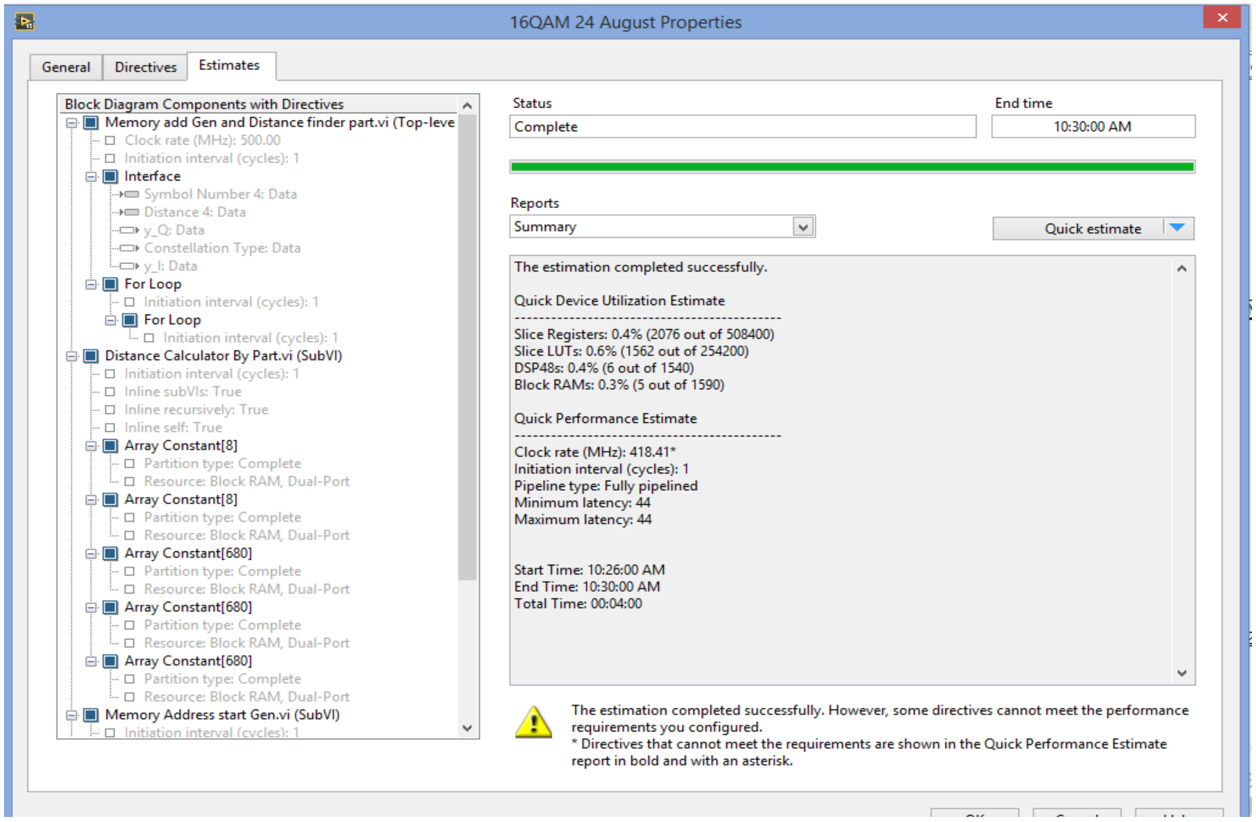
Status: Complete
End time: 10:21:05 AM

Reports: Summary

```
The estimation completed successfully.

Quick Device Utilization Estimate
-----------------------------------------------
Slice Registers: 0.3% (1725 out of 508400)
Slice LUTs: 0.5% (1243 out of 254200)
DSP48s: 0.4% (6 out of 1540)
Block RAMs: 0.3% (5 out of 1590)

Quick Performance Estimate
-----------------------------------------------
Clock rate (MHz): 418.41*
Initiation interval (cycles): 1
Pipeline type: Fully pipelined
Minimum latency: 43
Maximum latency: 43


Start Time: 10:16:50 AM
End Time: 10:21:05 AM
Total Time: 00:04:14
```

The estimation completed successfully. However, some directives cannot meet the performance requirements you configured.
* Directives that cannot meet the requirements are shown in the Quick Performance Estimate report in bold and with an asterisk.

QPSK Synthesis



16QAM 24 August Properties

General | Directives | Estimates

Block Diagram Components with Directives
- Memory add Gen and Distance finder part.vi (Top-leve
  - Clock rate (MHz): 500.00
  - Initiation interval (cycles): 1
  - Interface
    - Symbol Number 4: Data
    - Distance 4: Data
    - y_Q: Data
    - Constellation Type: Data
    - y_I: Data
  - For Loop
    - Initiation interval (cycles): 1
    - For Loop
      - Initiation interval (cycles): 1
- Distance Calculator By Part.vi (SubVI)
  - Initiation interval (cycles): 1
  - Inline subVIs: True
  - Inline recursively: True
  - Inline self: True
  - Array Constant[8]
    - Partition type: Complete
    - Resource: Block RAM, Dual-Port
  - Array Constant[8]
    - Partition type: Complete
    - Resource: Block RAM, Dual-Port
  - Array Constant[680]
    - Partition type: Complete
    - Resource: Block RAM, Dual-Port
  - Array Constant[680]
    - Partition type: Complete
    - Resource: Block RAM, Dual-Port
  - Array Constant[680]
    - Partition type: Complete
    - Resource: Block RAM, Dual-Port
- Memory Address start Gen.vi (SubVI)
  - Initiation interval (cycles): 1

Status: Complete
End time: 10:30:00 AM

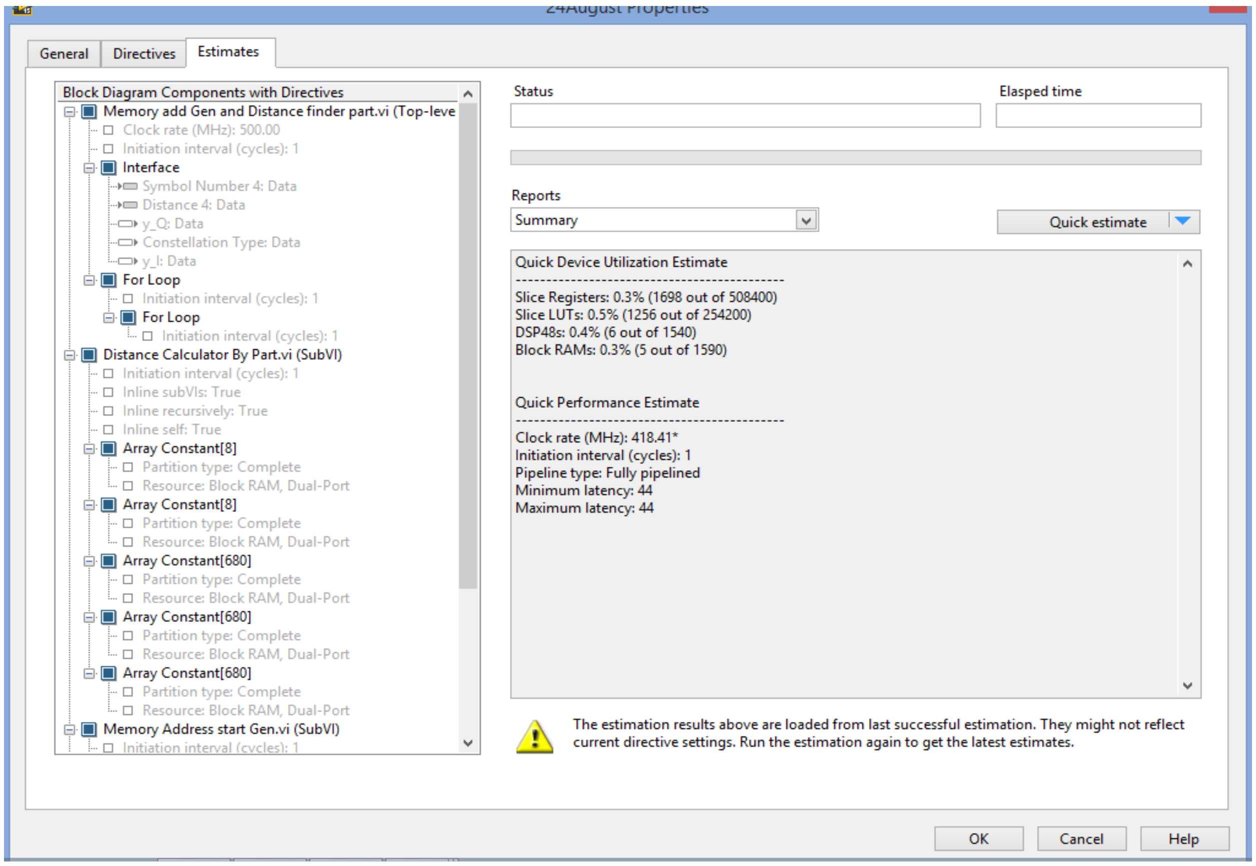
Reports: Summary

Quick estimate

```
The estimation completed successfully.

Quick Device Utilization Estimate
-------------------------------------------
Slice Registers: 0.4% (2076 out of 508400)
Slice LUTs: 0.6% (1562 out of 254200)
DSP48s: 0.4% (6 out of 1540)
Block RAMs: 0.3% (5 out of 1590)

Quick Performance Estimate
-------------------------------------------
Clock rate (MHz): 418.41*
Initiation interval (cycles): 1
Pipeline type: Fully pipelined
Minimum latency: 44
Maximum latency: 44


Start Time: 10:26:00 AM
End Time: 10:30:00 AM
Total Time: 00:04:00
```

The estimation completed successfully. However, some directives cannot meet the performance requirements you configured.
* Directives that cannot meet the requirements are shown in the Quick Performance Estimate report in bold and with an asterisk.

16QAM Synthesis

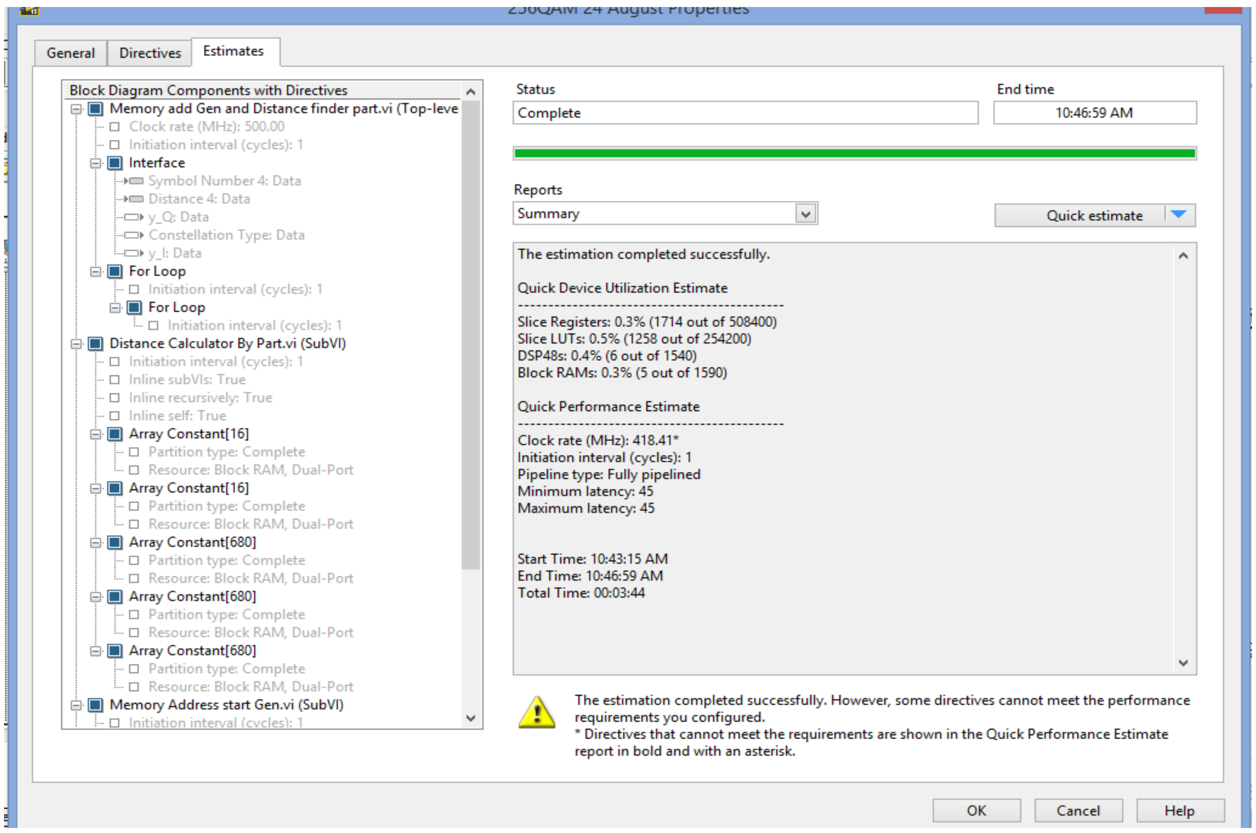


```
Block Diagram Components with Directives
Memory add Gen and Distance finder part.vi (Top-leve
  Clock rate (MHz): 500.00
  Initiation interval (cycles): 1
  Interface
    Symbol Number 4: Data
    Distance 4: Data
    y_Q: Data
    Constellation Type: Data
    y_I: Data
  For Loop
    Initiation interval (cycles): 1
    For Loop
      Initiation interval (cycles): 1
Distance Calculator By Part.vi (SubVI)
  Initiation interval (cycles): 1
  Inline subVIs: True
  Inline recursively: True
  Inline self: True
  Array Constant[8]
    Partition type: Complete
    Resource: Block RAM, Dual-Port
  Array Constant[8]
    Partition type: Complete
    Resource: Block RAM, Dual-Port
  Array Constant[680]
    Partition type: Complete
    Resource: Block RAM, Dual-Port
  Array Constant[680]
    Partition type: Complete
    Resource: Block RAM, Dual-Port
  Array Constant[680]
    Partition type: Complete
    Resource: Block RAM, Dual-Port
Memory Address start Gen.vi (SubVI)
  Initiation interval (cycles): 1
```

```
Quick Device Utilization Estimate
--------------------------------------------
Slice Registers: 0.3% (1698 out of 508400)
Slice LUTs: 0.5% (1256 out of 254200)
DSP48s: 0.4% (6 out of 1540)
Block RAMs: 0.3% (5 out of 1590)


Quick Performance Estimate
--------------------------------------------
Clock rate (MHz): 418.41*
Initiation interval (cycles): 1
Pipeline type: Fully pipelined
Minimum latency: 44
Maximum latency: 44
```

The estimation results above are loaded from last successful estimation. They might not reflect current directive settings. Run the estimation again to get the latest estimates.

64QAM Synthesis



Block Diagram Components with Directives

- Memory add Gen and Distance finder part.vi (Top-leve
  - Clock rate (MHz): 500.00
  - Initiation interval (cycles): 1
  - Interface
    - Symbol Number 4: Data
    - Distance 4: Data
    - y_Q: Data
    - Constellation Type: Data
    - y_I: Data
  - For Loop
    - Initiation interval (cycles): 1
    - For Loop
      - Initiation interval (cycles): 1
- Distance Calculator By Part.vi (SubVI)
  - Initiation interval (cycles): 1
  - Inline subVIs: True
  - Inline recursively: True
  - Inline self: True
  - Array Constant[16]
    - Partition type: Complete
    - Resource: Block RAM, Dual-Port
  - Array Constant[16]
    - Partition type: Complete
    - Resource: Block RAM, Dual-Port
  - Array Constant[680]
    - Partition type: Complete
    - Resource: Block RAM, Dual-Port
  - Array Constant[680]
    - Partition type: Complete
    - Resource: Block RAM, Dual-Port
  - Array Constant[680]
    - Partition type: Complete
    - Resource: Block RAM, Dual-Port
- Memory Address start Gen.vi (SubVI)
  - Initiation interval (cycles): 1

Status	End time
Complete	10:46:59 AM

Reports: Summary

```
The estimation completed successfully.

Quick Device Utilization Estimate
----------------------------------------------------
Slice Registers: 0.3% (1714 out of 508400)
Slice LUTs: 0.5% (1258 out of 254200)
DSP48s: 0.4% (6 out of 1540)
Block RAMs: 0.3% (5 out of 1590)

Quick Performance Estimate
----------------------------------------------------
Clock rate (MHz): 418.41*
Initiation interval (cycles): 1
Pipeline type: Fully pipelined
Minimum latency: 45
Maximum latency: 45


Start Time: 10:43:15 AM
End Time: 10:46:59 AM
Total Time: 00:03:44
```

The estimation completed successfully. However, some directives cannot meet the performance requirements you configured.
* Directives that cannot meet the requirements are shown in the Quick Performance Estimate report in bold and with an asterisk.

256QAM Synthesis