INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY **DELHI**

Exploring RF Data Converters on RFSoC Platform

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

M.Tech

BY

SOMYA SHARMA

Electronics and Communication Engineering

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI
NEW DELHI– 110020

July 4, 2022

Acknowledgement

I would like this opportunity to extend my sincere gratitude to all the people who have helped me in one way or another during my entire thesis. First and foremost, I would like to thank my advisor, Dr. Sumit J Darak for giving me a great opportunity to work with him for my MTech Thesis. He has been an excellent guide during my entire course of thesis for the past year. He has always been very active with weekly meetings and discussing the work updates which helped in maintaining a good flow of the thesis. The most inspiring thing was that he always has a good tentative plan for the entire thesis since the beginning. He has been extremely helpful in facing the challenges and gives the student chance to explore more ways to work around a problem and focuses more on learning aspects which is very crucial for the student's growth. Working with him made me learn non-technical project management skills as well. Besides, I would like to thank my friends Asrar and Aamir for their continuous emotional and motivational support during my thesis work and helping me manage other aspects of college life as well.

Abstract

The evolution of conventional field-programmable gate array (FPGA) platforms to all programmable multi-processor system-on-chip (MPSoC) platforms in the last decade has comprehensively addressed the scalability and flexibility requirements of next-generation electronic systems. To meet the large bandwidth and multi-standard requirements of upcoming wireless, satellite, and radar applications, the MPSoC platform with on-chip radio frequency (RF) data converters, RFSoC, has been introduced recently. Though RFSoC offers significant improvement in area, power and latency of the wireless systems over conventional multi-chip platforms, there is a significant gap in the existing literature on the configuration of the RFSoC platform for real-world demonstration. The work presented in this thesis aims to bridge this gap, thereby enabling engineers and researchers from academia and industry to efficiently and quickly configure the RFSoC platform.

The first contribution of this thesis is to study various features of RF data converter in RFSoC comprising multiple analog-to-digital converters (ADC) and digital-to-analog converters (DAC) along with analog-front-end. Next, a detailed configuration process of RF data converters for any desired carrier frequency and transmission bandwidth is discussed. This includes the clock generation and configuration in RF data converters and programming of in-built interpolation and decimation stages of the DAC and ADC, respectively. The second contribution involves the real-radio performance analysis of RF data converters using an end-to-end IEEE 802.11-based wireless physical layer (PHY). Specifically, an in-depth tutorial on the integration of wireless PHY with RF data converters for any given carrier frequency and data rate is presented via various illustrative examples. The work includes the design of hardware IP cores for digital-up converters (DUC) and digital down converters (DDC) on FPGA, their integration with baseband PHY and RF data converters via hardware-software co-design and PYNQ-based graphical user interface (GUI) on ARM processor for performance analysis. We validate the functional correctness of the designs in the presence of fixed-point word-length effects, quantization error due to data converters, and RF impairments via bit-error-rate (BER) performance on the RFSoC.

Contents

1 Introduction **1**
1.1 Motivation 1
1.2 Objectives 2
1.3 Thesis Outline 2

2 RF Data Converters **4**

3 OFDM PHY Layer Implementation **7**
3.1 Transmitter Block 8
3.2 Interpolate and Decimate Blocks 8
3.3 Receiver Block 9

4 TICS Software for Clock Generation **10**

5 Implementation Methodology **15**
5.1 Step 1: IP generation using System Generator and HDL Coder . . . 15
5.2 Step 2: Vivado IP Integrator 16
5.2.1 Architecture 18
5.3 Step 3: Clock Configuration File Generation 20
5.4 Step 4: PYNQ Setup 24

6 Results **26**

7 Future Work and Conclusion **29**

Chapter 1

Introduction

1.1 Motivation

there are no references

Next generation electronic systems demand scalability and flexibility from the hardware platforms in addition to the conventional requirements of area, power and cost efficient architectures. The scalable architectures enables the use of same platform for wide range of products while flexible architecture enables feature richness and future upgradability on-the-fly. The evolution of conventional field-programmable gate array (FPGA) platforms to all programmable multi-processor system-on-chip (MPSoC) platforms in the last decade has comprehensively addressed these requirements. To meet the large bandwidth and multi-standard requirements of upcoming wireless, satellite, and radar applications, the MPSoC platform needs to be integrated with data converters such as analog-to-digital converters (ADC) and digital-to-analog converters (DAC) and analog-front-end comprising of analog filters, mixers, amplifiers, antennas and their matching circuits as shown in Fig. 1.1. The presence of multiple discrete components results in lower data rate but large area and power consumption. Furthermore, the cost and design efforts to build the wireless transceivers are huge. The direct RF converters based approach shown in the Fig. 1.2 offer two chip solution comprising of MPSoC and analog-front-end (AFE) tightly integrated via FPGA Mezzanine Card (FMC) connectors.

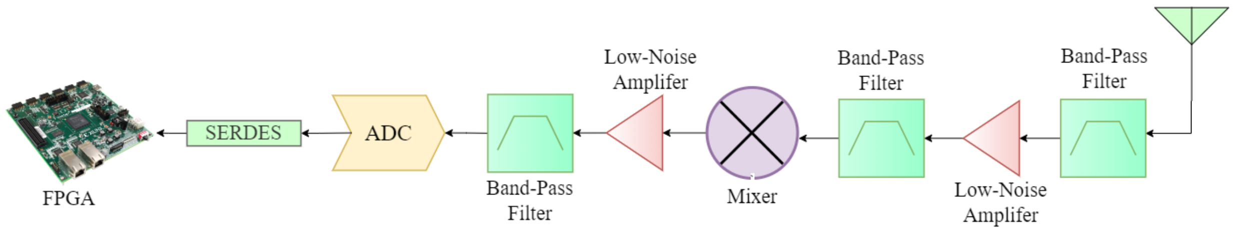


Figure 1.1: Conventional RF Signal Chain

The next obvious solution is to integrated RF data converters on the same chip as that of MPSoC. This is referred to as RFSoC which has been introduced recently by AMD-Xilinx. Though RFSoC offers significant improvement in area, power and latency of the wireless systems over conventional multi-chip platforms, there is a significant gap in the existing literature on the configuration of the RFSoC platform for

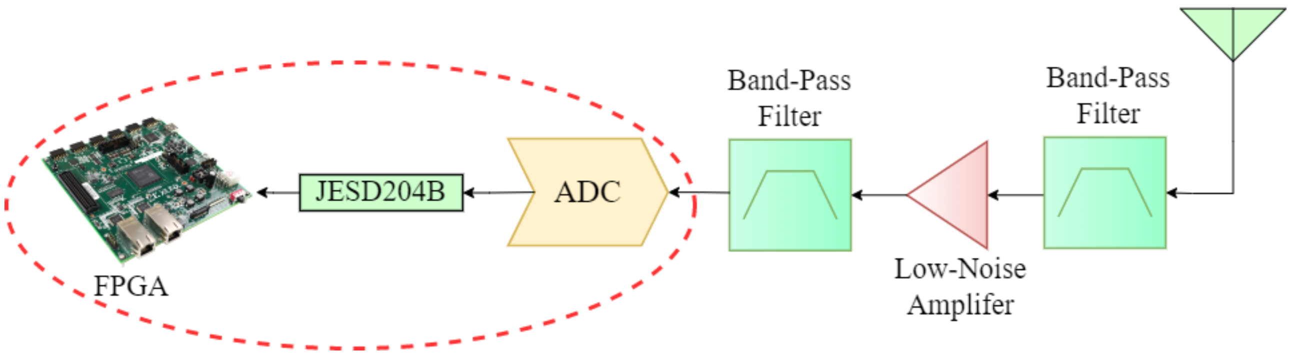


Figure 1.2: RF Signal Chain with Direct RF Converters

real-world demonstration. The work presented in this thesis aims to bridge this gap, thereby enabling engineers and researchers from academia and industry to efficiently and quickly configure the RFSoC platform.

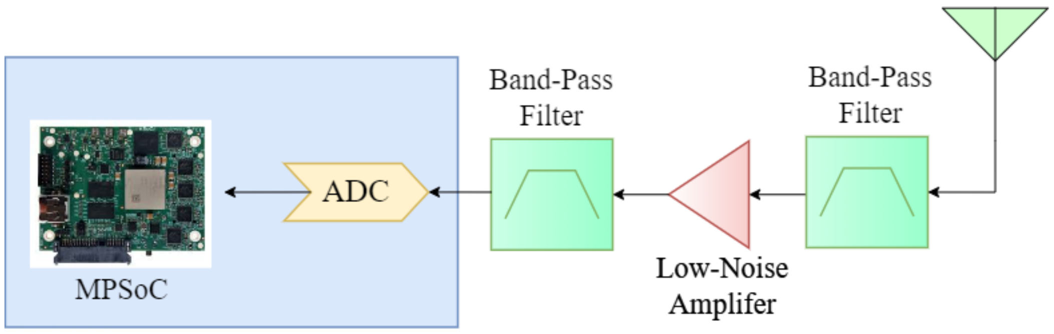


Figure 1.3: RF Signal Chain after RFSoC

1.2 Objectives

This work aims at exploring RF data converters of the Zynq RFSoC ZCU111 evaluation kit. The goal is to explore how to configure the data converters for a given sampling rate, reference frequencies available for a particular sampling rate, generating clock configuration files to configure the clock modules. For theis, an OFDM example implementation on RFSoC by University of Strathclyde- Software Defined Radio Research Laboratory is used as a base model which will be discussed in the further chapters. This design is an end-to-end transceiver implementation on the ZCU111 kit. This design is first understood in depth and then exploited to test its functionality for varying sampling rates and data converters' frequencies.

1.3 Thesis Outline

The thesis is organized as follows. The Chapter 2: RF Data Converters discusses the architecture of the RF DACs/ADCs, their specifications and their working. The Chapter 3: OFDM PHY Layer Implementation explains the Simulink model of the

OFDM physical layer that is used in this work. The Chapter 4: TICS Software Clock Generation explains how to set up the LMX2594 module settings to generate clock configuration values for any frequency using the TICS software. The Chapter 5: Implementation Methodology explains the steps from simulink model to testing on hardware involving System Generator and HDL coder for IP generation, Vivado IP Integrator to create block design and generating bitstream and finally PYNQ to view the output results. It discusses RF data converter IP settings in vivado in great depths. Chapter 6: Results discusses the results obtained after implementing the design. Finally, Chapter 7: Conclusion and future work summarizes the entire work and discusses briefly the future aspects of this thesis.

Chapter 2

RF Data Converters

The RF Data Converters is one of the most interesting features of RFSoC devices. The integration of data converters on the FPGA itself enables generating very efficient systems in terms of power, area, complexity, etc as has already been discussed. Additionally, the in-built interpolators/decimators allows one to upconvert/downconvert the data rate from lower frquencies, say 300MSPS, to a higher 2.4GSPS without using any additional filter. In this work, the ZU28DRF-FFVG1517, Zynq UltraScale+ RFSoC Gen1 device (ZCU111), is used. It contains a total of 4 12-bit RF analog-to-digital converters which are distributed across 4 banks and 8 14-bit RF digital-to-analog (RF-DAC) converter channels distributed across two banks. The maximum achievable sampling frequency at the RF ADC is 4.096 GSPS and at the RF DAC is 6.554 GSPS. In this chapter, we will briefly discuss the architecture of these data converters and will focus on the RF data converter IP customization as per the requirements. The data listed in Tab. 2.1 specifies the minimum and maximum data sampling rates achievable at the DAC/ADC ends, and the interpolation and decimation factors.

Data Converter	Min. Sampling Rate (in GSPS)	Max. Sampling Rate (in GSPS)	Interpolation or Decimation Factors
ADC	1	4.096	1, 2, 4, or 8 (decimation)
DAC	0.5	6.554	1, 2, 4, or 8 (interpolation)

Table 2.1: RF Data Converters (Gen 1 devices)

The two architectures available for RF Data Converters are dual and quad tile architectures. In the dual architecture, each tile will have two data converters, mixers, decimators/interpolators each. While in quad architecture, each tile will have four of each components. The Gen 1 devices used in this work have dual RF-ADC and quad RF-DAC architecture which are shown in the Fig. 2.1, 2.2 respectively. Considering the dual RF-ADC tile architecture as shown in the Fig. 2.1, the sampling clock provided to both the ADCs is same and the mixer is grouped together with the decimator to form a digital downconverter. The first component after the RF-ADC tile is the digital I/Q mixer. It generates sine/cosine wave at the frequency equal to the carrier frequency of incoming signal using a NCO (Numerically controlled oscillator) which is then multiplied to the incoming signal. This shifts the signal back to the baseband. The other high frequency terms generated at double the carrier frequency are removed by the low pass filter. The three different modes of operation of the I/Q mixer are bypass mode where the mixer is entirely bypassed, fine mode where any arbitrary frequency signal is generated and multiplied to the received signal, and the coarse mode where only a limited set of frequencies can be multiplied to the signal. The second component is the decimator, which can perform rate reduction by factors 1, 2, 4, or 8 as selected by the user. The decimation is done by cascading half band filters, each decimating by a factor of 2, depending on the input decimation factor.

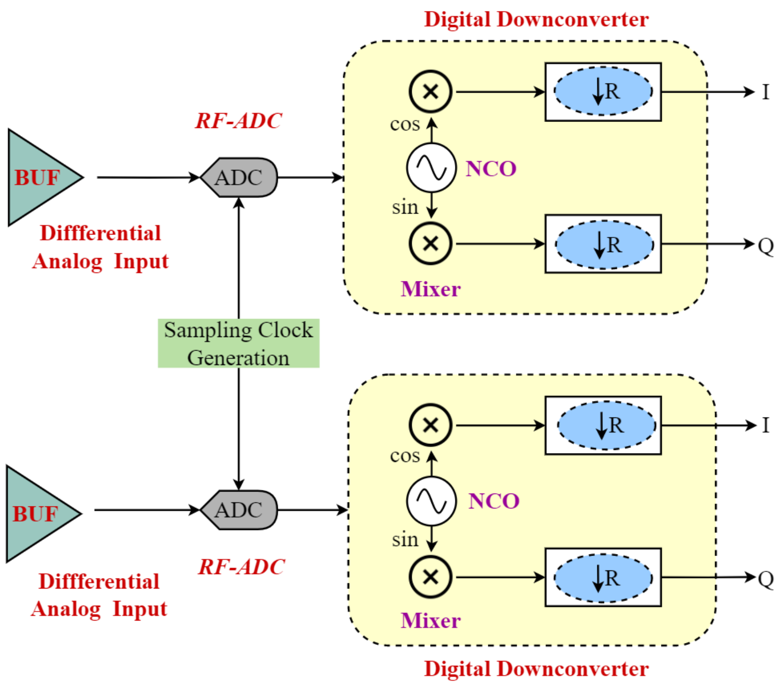


Figure 2.1: Dual RF ADC Architecture

The architecture for quad RF-DACs is very similar to that of ADC but in opposite set of operations and have four DACs in each tile instead of two. In the quad RF-DAC, there are 4 DACs being fed by the same sampling clock and have four I/Q mixers and interpolators. This architecture allows for multi-band operation since here, the output of a single DAC can be fed to all four interpolators where each can be tuned to different frequency thereby recovering different band (maximum of 4 bands). The dual ADC architecture can support only 2 such bands. The input I/Q symbols enter the RF data converters and then pass through a pair of interpolators to increase the sample rate as is set by the user. The interpolation rates available are 1, 2, 4, and 8. Then the I/Q mixer generates the sine/cosine wave to modulate the signal and then passed to the RF-DAC converter. The interpolation is also done by cascading identical half-band filters each increasing the rate by a factor of 2 as per the requirement.

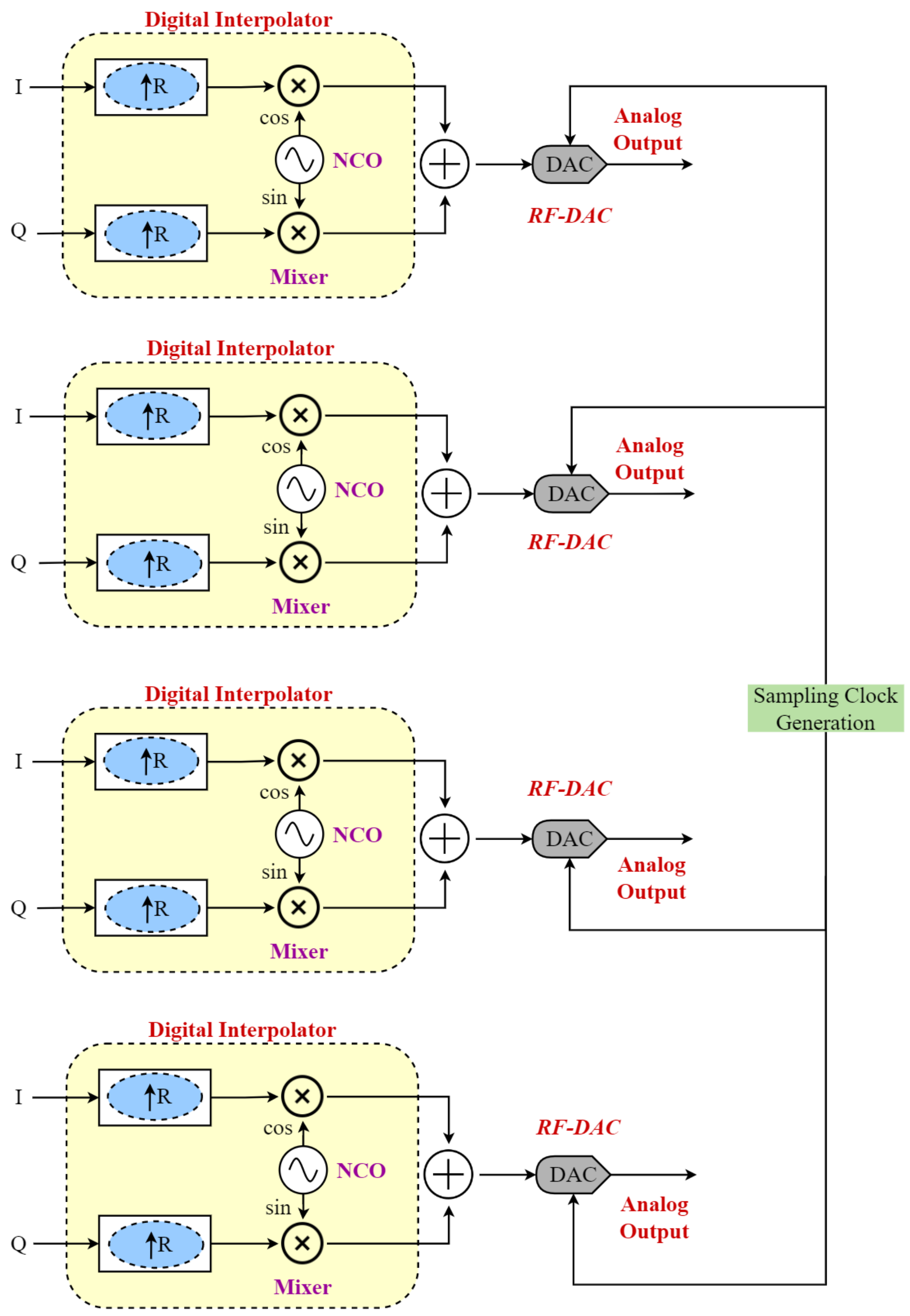


Figure 2.2: Quad RF DAC Architecture

Chapter 3

OFDM PHY Layer Implementation

The OFDM example implementation on RFSoC by University of Strathclyde-Software Defined Radio Research Laboratory is used as a base model in this thesis work. The basic implementation of the OFDM along the with the data transmission rates is shown in the Fig. 3.1 which is discussed below in detail. The entire design is created in the Simulink software by MATLAB. The design is first understood in detail and tested for its working on the ZCU111 hardware. It is then modified at different levels for varying frequencies and tested again on the board to verify the functionality for every modification.

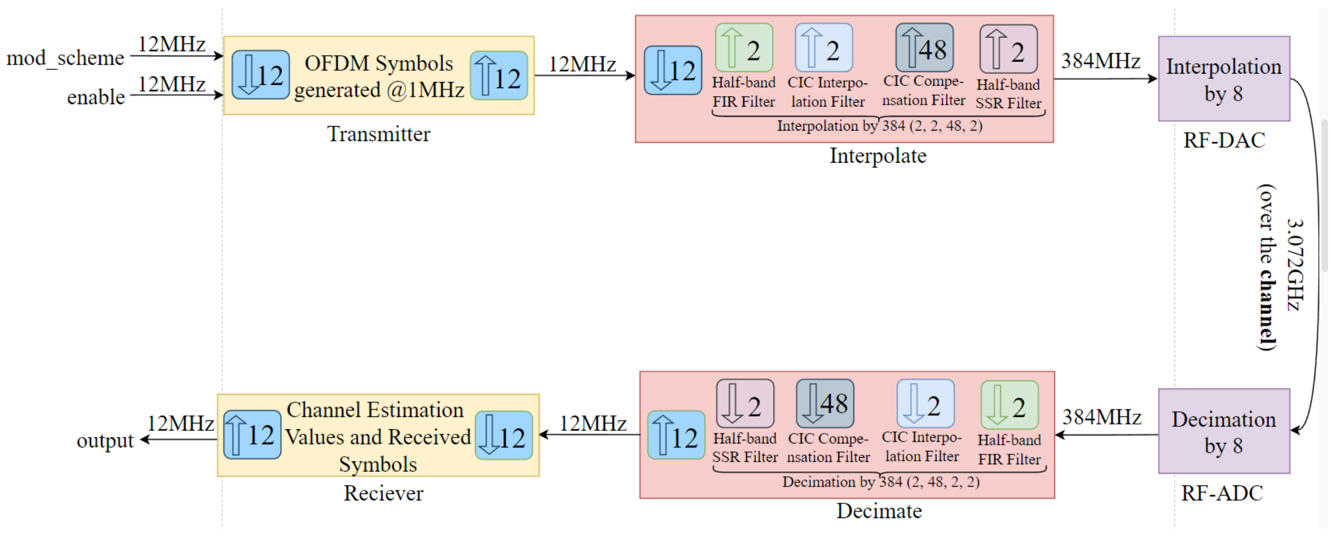


Figure 3.1: OFDM Design Model

The design consists of 4 separate blocks, viz. Transmitter, Interpolate, Decimate, and Receiver. These blocks are mapped to the hardware by HDL coder and System Generator tools. It is to be noted here that the data transmissions from/to PS and from one block to another occur at 12 MHz frequency except at the output of interpolate and the input of decimate block in the PL.

3.1 Transmitter Block

The transmitter block consists of the entire signal processing at the transmitter end. This block receives the modulation scheme and enable signal as input from the PS via AXI4-Lite channel. The execution in this block can be better understood from its sub-blocks as discussed below.

1. Control Signal Generator: This block generates the control signals for the entire OFDM burst. It generates preamble valid, data valid, pilot valid signals based on the IEEE 802.11a. These signals are then passed to the subsequent blocks. The execution of this block starts once the enable signal is turned high.

2. RF Data Generator: The preamble values (short training (STS) and long training symbols (LTS)), information (data) to be passed, and the pilot symbols are generated in this block. The data generated is modulated based on the input modulation scheme is modulated to I and Q complex symbols in the RF signal generator block.

3. Sub-Carrier Mapper: The complex data symbols, preamble and pilot symbols are then passed to this sub carrier mapping block in which the preamble and complex data symbols are rearranged.

4. IFFT: The output is then passed to the IFFT block where IFFT is performed only on the preamble and complex data symbols and not on pilot symbols. It converts frequency-domain subcarriers to produce the OFDM symbols in the time-domain.

5. Cyclic Prefix Addition: Finally, the IFFT output is passed to the Add Cyclic Prefix block (CP). The CP is added corresponing to LTS, STS, and the complex data symbols. This addition of CP is done to recognize the starting point of incoming data at the receiver end.

6. The output is finally up-sampled to 12 MHz for transfer to the next block, interpolator.

3.2 Interpolate and Decimate Blocks

The interpolation block receives its input from the transmitter block. This transmission is done at 12 MHz in the PL using multiple DMAs. This block first down-samples the data to 1 MHz frequency and then up-samples it to a required frequency using combination of certain filters. The interpolated symbols are then passed to the DAC tile.

This block consists of four filters for interpolation. First is half-band filter, second is CIC compensation filter, third is CIC interpolation filter, and fourth is half-band SSR filter. In the original design, the interpolation is done by factor of 2, 2, 48, 2 respectively. The SSR (super sample rate) filter at the last, added to interpolate the sample rate by a factor of 2, allows the data rate of 384 MHz while maintaining clock required for data transmission at 192 MHz. This is achieved by concatenating 2 symbols, adjacent in time frame and sending it as a single data block. For this, the transmission bus width at the output of interpolate is chosen as double the data width of single complex data symbol. This data is passed over a channel formed by a simple loopback connection to be received by the ADCs of the RFSoC and then passed to the decimate block of the OFDM design.

Decimate block receives the data from ADC tile of the RFSoC after down-conversion to baseband frequency. The decimation is done again by using a series of filters, like in the interpolate block but in opposite order, i.e., the SSR filter comes first to decimate by a factor of 2 (as opposed to last in the interpolate block). After decimation to 1 MHz, the data is up-sampled to 12 MHz before passing to the next block (receiver). The filters, delays, gains added in the model for these blocks are selected from the Xilinx library section in the Simulink library. The filter coefficients used for all the filters are generated using a separate MATLAB file depending on the interpolation/decimate rate, output sampling rate and the type of that particular filter.

3.3 Receiver Block

The receiver block gets its input from the decimate block and generates the original data. the data is first down-sampled from 12 MHz to 1 MHz. The sub-blocks in the receiver are discussed as below.

1. Timing and Frequency Synchronization: This block performs, frame detection for CP removal, coarse frequency estimation, and generates control signals like preamble valid or data valid. Schmid and Cox Timing Metric is used for the frame detection in this implementation.

2. FFT: At the receiver, FFT is performed on the OFDM symbols to recover the frequency domain symbols.

3. One Tap Equalizer: This block performs channel estimation and equalization step. The preamble symbols are used for channel estimation and least square estimation methodology is employed. These estimated values are then used by the equalizer where the channel estimated values are multiplied to the data symbols so as to counter the impact of channel on the transferred data bits.

4. Phase Tracking 1: This block tracks linear phase errors which occurs due to sampling phase frequency offsets. The implemented algorithm estimates the gradient of the linear phase error using pilot sub-carriers.

5. Phase Tracking 2: This block tracks common phase error (CPE) from symbol to symbol. CPE is caused by residual frequency offset and phase noise effects.

6. The final output from Phase Tracking 2 is sent to the PS. The output consists of the modulated complex data symbols.

Chapter 4

TICS Software for Clock Generation

The ZCU111 board consists of two evaluation modules (EVMs), viz. LMK04208 and LMX2594 which generates clock for programming the RF data converters. The LMK04208 is a jitter cleaner and a clock generator module whereas the LMX2594 is a high performance clock generator. Of these two, LMX2594 provides the sampling clock for the RF DACs/ADCs. As is highlighted in the Fig.4.1, for this particular case, the incoming data at the RF DAC is clocked at 384MHz. To sample this data, LMX2594 provides this clock (384MHz) to RF DAC. Similarly, to sample the output data from RF ADC, LMX2594 only provides the clock.

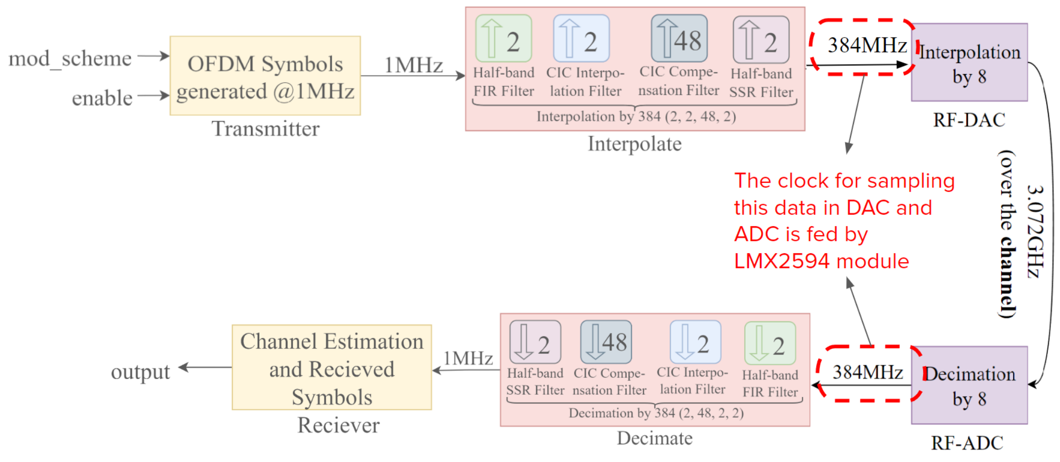
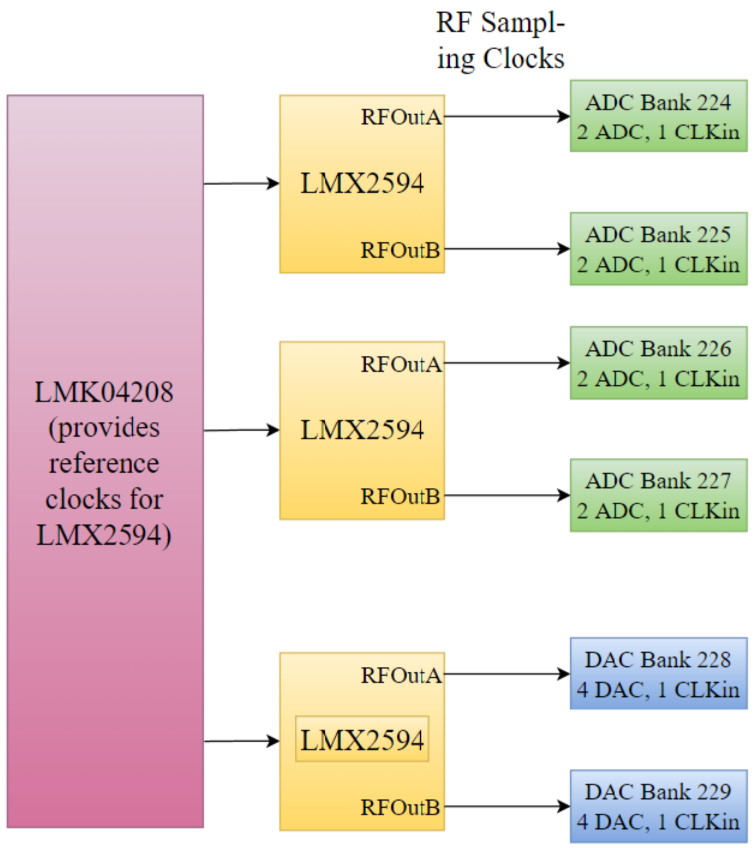


Figure 4.1: OFDM Model

The RF clocking architecture is shown in the Fig. 4.2. Here, the LMK04208 provides the input reference clocks to the LMX2594. The LMX2594 then based on its configuration, generates the clock which is fed directly to the DAC and ADC banks. These chips consists of various blocks like phase synchronizers, VCO mixers, etc that are required for clock generation. The values of these blocks are to be programmed by the user according to the required frequency output which then feeds the RF Data converters. A configuration (.txt) file is required to program these modules. This file is generated by using the TICS software.



RF Sampling Clocks
LMK04208 (provides reference clocks for LMX2594)
LMX2594
RFOutA
RFOutB
ADC Bank 224 2 ADC, 1 CLKin
ADC Bank 225 2 ADC, 1 CLKin
LMX2594
RFOutA
RFOutB
ADC Bank 226 2 ADC, 1 CLKin
ADC Bank 227 2 ADC, 1 CLKin
LMX2594
RFOutA
RFOutB
DAC Bank 228 4 DAC, 1 CLKin
DAC Bank 229 4 DAC, 1 CLKin

Figure 4.2: RF Clocking Architecture

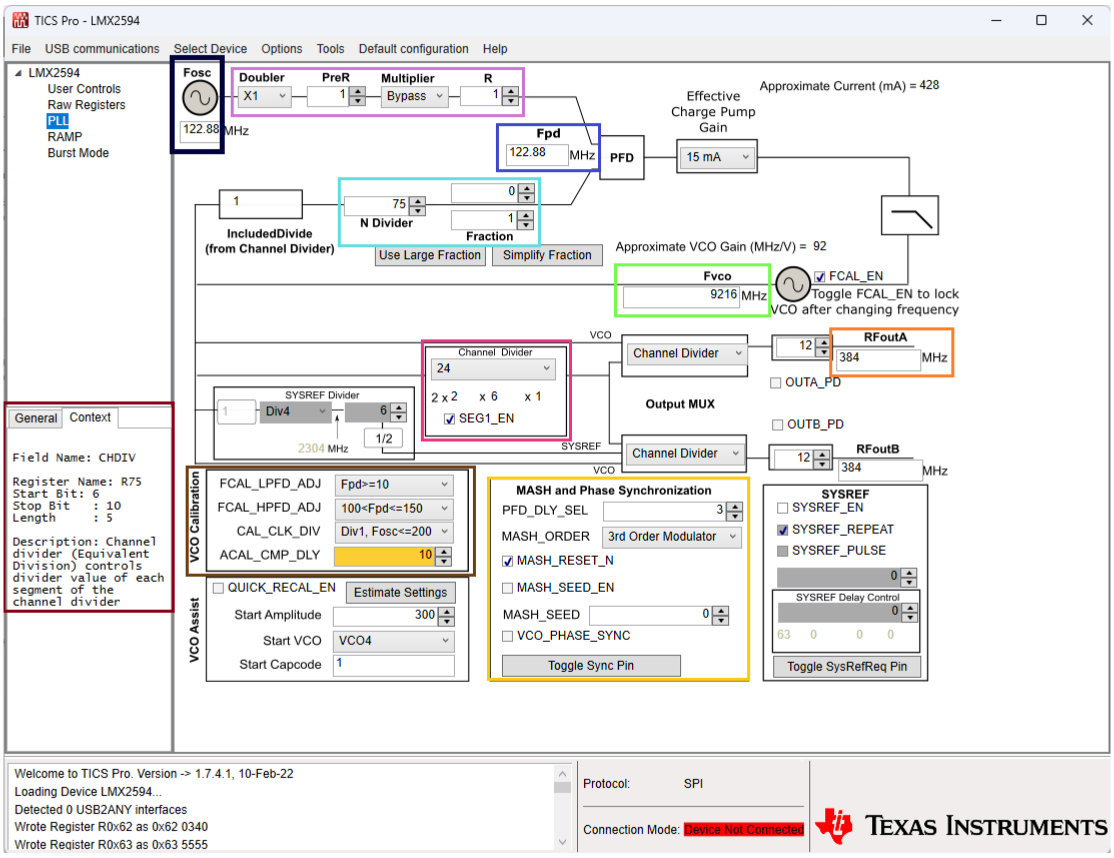
The Texas Instruments Clock Synthesizer (TICS SW Pro) software provides an interactive interface to generate the clock configuration files which are used to program the evaluation modules (EVMs) for a certain frequency. Below shown in the Fig. 4.3 is the snapshot of the setup for generating 384 MHz clock from LMX2594 module in the TICS software. The various highlighted parameters are discussed below.

1. The input to LMX2594 chip Fosc is set to a frequency of 122.88 MHz.

2. The Doubler (OSc_2X) is used to up the input frequency signal (Fosc). The doubler can have two values, X1 (OSc_2X = 0 means that the doubler is disabled) and X2 (OSc_2X = 1 means that the doubler is enabled and the input signal frequency is doubled). The doubler is used to reduce spurs in the noise signal or increase the phase detector frequency.

3. The PreR divider is used to divide the frequency fed to PLL_R (R) divider. The maximum frequency limit to the input of PLL_R divider is 250 MHz and hence PRE_L divider is used to make sure this criteria is met. It is used iff multiplier is used.

4. The PLL_R further divides the frequency for the phase detector (F_{pd}). The


TICS Pro - LMX2594

File USB communications Select Device Options Tools Default configuration Help

LMX2594
User Controls
Raw Registers
PLL
RAMP
Burst Mode

Fosc: 122.88 MHz

Doubler: X1 | PreR: 1 | Multiplier: Bypass | R: 1

Fpd: 122.88 MHz

PFD

Effective Charge Pump Gain: 15 mA

Approximate Current (mA) = 428

IncludedDivide (from Channel Divider): 1

N Divider: 75

Fraction: 0 / 1

Use Large Fraction | Simplify Fraction

Approximate VCO Gain (MHz/V) = 92

Fvco: 9216 MHz

☑ FCAL_EN
Toggle FCAL_EN to lock VCO after changing frequency

Channel Divider: 24
2 x 2 x 6 x 1
☑ SEG1_EN

SYSREF Divider: 1 | Div4 | 6 | 1/2
2304 MHz

Output MUX: Channel Divider (RFoutA), Channel Divider (RFoutB)

RFoutA: 12 | 384 MHz
☐ OUTA_PD

☐ OUTB_PD
RFoutB: 12 | 384 MHz

General | Context

```
Field Name: CHDIV

Register Name: R75
Start Bit: 6
Stop Bit   : 10
Length     : 5

Description: Channel
divider (Equivalent
Division) controls
divider value of each
segment of the
channel divider
```

VCO Calibration
FCAL_LPFD_ADJ: Fpd>=10
FCAL_HPFD_ADJ: 100<Fpd<=150
CAL_CLK_DIV: Div1, Fosc<=200
ACAL_CMP_DLY: 10

VCO Assist
☐ QUICK_RECAL_EN | Estimate Settings
Start Amplitude: 300
Start VCO: VCO4
Start Capcode: 1

MASH and Phase Synchronization
PFD_DLY_SEL: 3
MASH_ORDER: 3rd Order Modulator
☑ MASH_RESET_N
☐ MASH_SEED_EN
MASH_SEED: 0
☐ VCO_PHASE_SYNC
Toggle Sync Pin

SYSREF
☐ SYSREF_EN
☑ SYSREF_REPEAT
☐ SYSREF_PULSE
0
SYSREF Delay Control: 0
63 0 0 0
Toggle SysRefReq Pin

```
Welcome to TICS Pro. Version -> 1.7.4.1, 10-Feb-22
Loading Device LMX2594...
Detected 0 USB2ANY interfaces
Wrote Register R0x62 as 0x62 0340
Wrote Register R0x63 as 0x63 5555
```

Protocol: SPI
Connection Mode: Device Not Connected

Figure 4.3: LMX2594 setup to generate 384 MHz clock in TICS SW Pro

expression for F_{pd} is given as

$$F_{pd} = \frac{F_{osc} * OSC\_2X * MULT}{Pre\_R * PLL\_R}$$

5. N divider and Fraction (NUM and DEN) are used to precisely output the clock of frequencies with decimal part like 245.**76** MHz. The F_{VCO} generated can have min value of 7.5GHz and maximum of 15GHz.

$$F_{VCO} = F_{pd} * (N + \frac{NUM}{DEN})$$

The N-divider has minimum value restrictions based on the modulator order and VCO frequency (F_{VCO}). Also note that, for case when fraction is to be bypassed, i.e., when F_{VCO} is integral multiple (N) of F_{pd}, the denominator is to be set as 1 and not 0 ($\frac{NUM}{DEN}$ with DEN as 0 is a computational error.)

6. The Channel Divider (CHDIV) consists of a series of several dividers and is used to generate frequencies lower than minimum limit of VCO. Also, above 10 GHz, the maximum allowable channel divider value is 6.

   (a) SEG_EN1 is an enable buffer for channel divider. It is disabled only for channel divider value 2 and enabled for channel divider > 2.

   (b) It is to be noted here that the $F_{VCO} \leq 11.5GHz$ for CHDIV$\geq$ 8. This constarint will be used later in computing the parameters' value for any desired frequency.

7. The RFoutA is the final output clock which will be passed to the DAC ADC tiles. It is given by the expression:

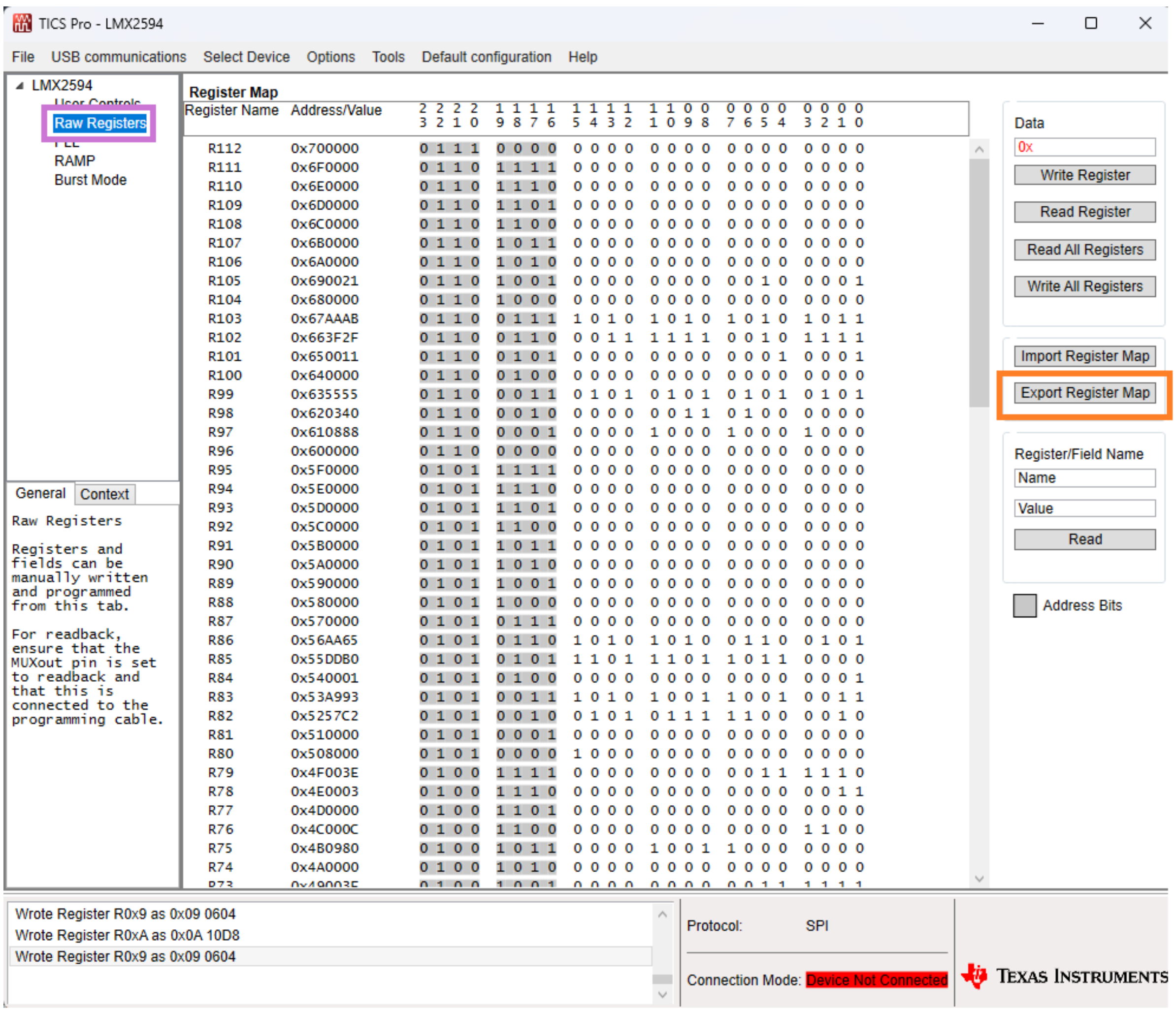
$$RF_{OUTA} = \frac{F_{VCO}}{CHDIV}$$

8. The parameters as highlighted in VCO Calibration are set for faster and more efficient amplitude calibration without compromising on the low phase noise. The FCAL_LPFD_ADJ and FCAL_HPFD_ADJ are used to adjust the calibration speed. ACAL_CMP_DLY is used for delay insertion during VCO amplitude calibration. Lowering this value can speed up VCO calibration, but lowering it too much may degrade VCO phase noise.

9. Phase synchronization: The phase synchronization block is used for synchronizing the delay from the rising edge of the OSCin signal to the output signal.

10. Each parameter corresponds to a register value, some of which have been discussed above. For example, in Fig. 4.3, the CHDIV parameter corresponds to the Register R75 as is highlighted and a brief description is displayed in the bottom left corner of the window itself.

The entire configuration of the LMX2594 module consists of 113 registers whose values are to be set. These register values correspond to the vaues of the parameters that are set, as discussed above. The register values are represented in hexadecimal format and listed in the 'RAW Registers' tab as shown in Fig. 4.4. These values are then exported as a .txt file which is later used to configure the data converters sampling clock.



Raw Registers

Registers and fields can be manually written and programmed from this tab.

For readback, ensure that the MUXout pin is set to readback and that this is connected to the programming cable.

Register Map

Register Name	Address/Value	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
R112	0x700000	0 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R111	0x6F0000	0 1 1 0	1 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R110	0x6E0000	0 1 1 0	1 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R109	0x6D0000	0 1 1 0	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R108	0x6C0000	0 1 1 0	1 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R107	0x6B0000	0 1 1 0	1 0 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R106	0x6A0000	0 1 1 0	1 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R105	0x690021	0 1 1 0	1 0 0 1	0 0 0 0	0 0 0 0	0 0 1 0	0 0 0 1
R104	0x680000	0 1 1 0	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R103	0x67AAAB	0 1 1 0	0 1 1 1	1 0 1 0	1 0 1 0	1 0 1 0	1 0 1 1
R102	0x663F2F	0 1 1 0	0 1 1 0	0 0 1 1	1 1 1 1	0 0 1 0	1 1 1 1
R101	0x650011	0 1 1 0	0 1 0 1	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1
R100	0x640000	0 1 1 0	0 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R99	0x635555	0 1 1 0	0 0 1 1	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1
R98	0x620340	0 1 1 0	0 0 1 0	0 0 0 0	0 0 1 1	0 1 0 0	0 0 0 0
R97	0x610888	0 1 1 0	0 0 0 1	0 0 0 0	1 0 0 0	1 0 0 0	1 0 0 0
R96	0x600000	0 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R95	0x5F0000	0 1 0 1	1 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R94	0x5E0000	0 1 0 1	1 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R93	0x5D0000	0 1 0 1	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R92	0x5C0000	0 1 0 1	1 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R91	0x5B0000	0 1 0 1	1 0 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R90	0x5A0000	0 1 0 1	1 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R89	0x590000	0 1 0 1	1 0 0 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R88	0x580000	0 1 0 1	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R87	0x570000	0 1 0 1	0 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R86	0x56AA65	0 1 0 1	0 1 1 0	1 0 1 0	1 0 1 0	0 1 1 0	0 1 0 1
R85	0x55DDB0	0 1 0 1	0 1 0 1	1 1 0 1	1 1 0 1	1 0 1 1	0 0 0 0
R84	0x540001	0 1 0 1	0 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1
R83	0x53A993	0 1 0 1	0 0 1 1	1 0 1 0	1 0 0 1	1 0 0 1	0 0 1 1
R82	0x5257C2	0 1 0 1	0 0 1 0	0 1 0 1	0 1 1 1	1 1 0 0	0 0 1 0
R81	0x510000	0 1 0 1	0 0 0 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R80	0x508000	0 1 0 1	0 0 0 0	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R79	0x4F003E	0 1 0 0	1 1 1 1	0 0 0 0	0 0 0 0	0 0 1 1	1 1 1 0
R78	0x4E0003	0 1 0 0	1 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1
R77	0x4D0000	0 1 0 0	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R76	0x4C000C	0 1 0 0	1 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	1 1 0 0
R75	0x4B0980	0 1 0 0	1 0 1 1	0 0 0 0	1 0 0 1	1 0 0 0	0 0 0 0
R74	0x4A0000	0 1 0 0	1 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
R73	0x49003E	0 1 0 0	1 0 0 1	0 0 0 0	0 0 0 0	0 0 1 1	1 1 1 1

```
Wrote Register R0x9 as 0x09 0604
Wrote Register R0xA as 0x0A 10D8
Wrote Register R0x9 as 0x09 0604
```

Protocol: SPI

Connection Mode: Device Not Connected

Figure 4.4: The register values for 384 MHz clock

Chapter 5

Implementation Methodology

The OFDM design as explained in the previous chapter is implemented in the Simulink MATLAB. The simulink model consists of additional PS blocks (for modulation scheme input) and channel block for the end-to-end simulation as shown in the Fig. 5.1, and verification of the designs. There are four blocks that are hardware mappable and are exported as vivado IPs to be imported in the Vivado IP Integrator. The entire process from simulink model to testing the design on hardware (ZCU111) board is a four step process. We will understand it with an example design. In this example design, the filters interpolate by factors 2, 2, 48, 2 and the RF DAC further interpolates by factor of 8. The data rates are same as those mentioned in Fig. 3.1

5.1 Step 1: IP generation using System Generator and HDL Coder

The four IPs that are generated from the Simulink Models are Transmitter, Interpolate, Decimate, and Receiver. There are three separate Simulink models provided for generating these IPs.

1. OFDM_TX_HW: This model is used to generate only the interpolate IP.
2. OFDM_RX_HW: This model is used to generate the receiver end IPs which are receiver and the decimate.
3. OFDM_TX_RX_fixed_point: This model is used to generate the transmitter IP.

The interpolate and decimate IPs are generated using the System Generator and hence these subsystems are created using the Xilinx supported hardware blocks from the library browser. The transmitter and receiver IPs are generated using the HDL coder tool by the MATLAB. To test for varying frequencies, the interpolate and decimate blocks and the RF Data Converter IP settings need to be changed as per the requirement.

As has been discussed, the interpolate and decimate blocks use series of FIR Filters for interpolation and decimation purposes. The filter coefficients are generated using a separate MATLAB file based on the interpolation/decimation rate, output frequency, and the type of the particular filter as is highlighted in the Fig. 5.2. In this, first the interpolation factors of all the filters are defined, then the output frequency of all the respective filters is defined. Then finally, the coefficients are generated using inbuilt MATLAB function, as shown for the CIC Compensation filter.

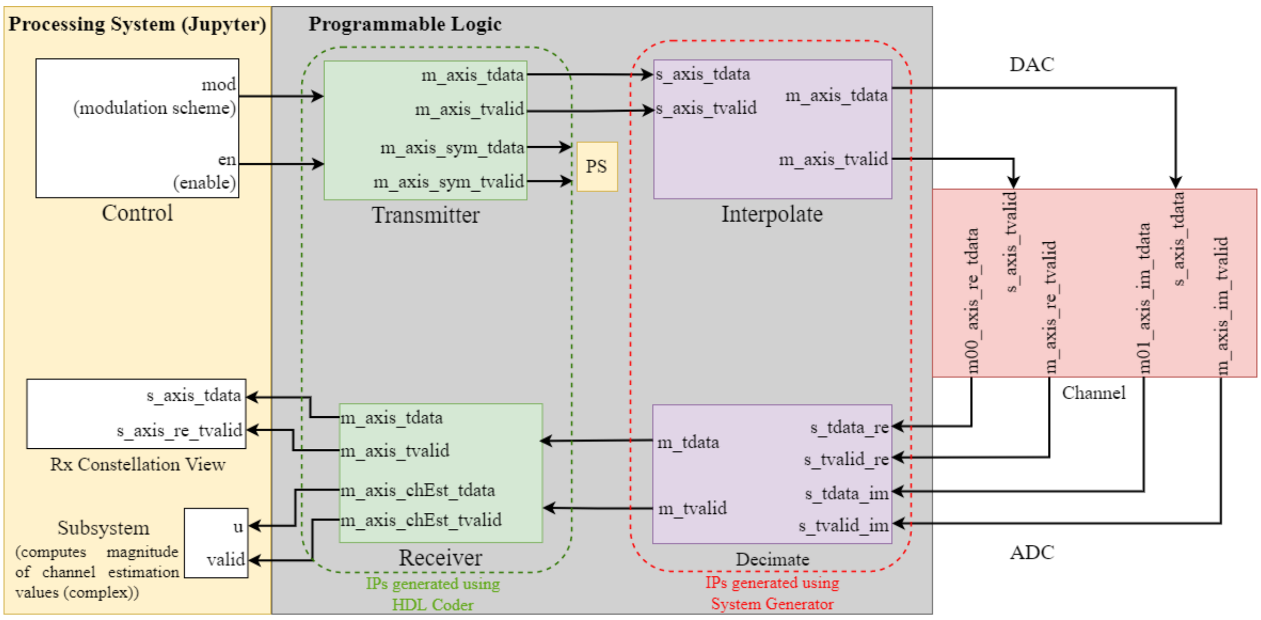


Figure 5.1: OFDM Implementation Simulink Model

Once this is done, the interpolation/decimation factor is set in the Simulink model also. Below shown in the Fig. 5.3, is highlighted the interpolation factor specification of the CIC Compensation filter block in the Simulink Model.

Additionally, the data sample rates in the entire Simulink model can be displayed in a color coded manner to better visualize the sample rates throughtout the design and to verify the changes made in interpolate and decimate blocks. For example, as shown in the Fig 5.4, the incoming data is at 1MHz. After passing through half-band filters, the data is upconverted to 2MHz. Further, the CIC compensation filter upconverts it to 4MHz. The CIC interpolation filter then further upconverts it to 192MHz.

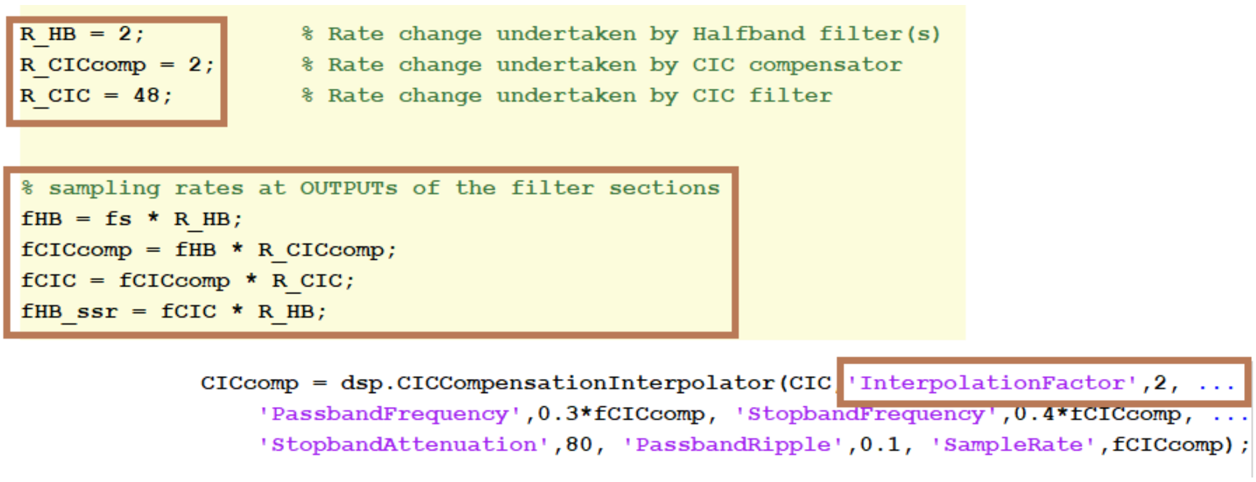
The transmitter and receiver blocks remain the same throughout.

5.2 Step 2: Vivado IP Integrator

After generating the IPs from the Simulink Models, the Xilinx Vivado tool is used to generate the block design that is to be implemented on the hardware. The enitre block design is shown in the Fig. 5.5. In this figure is highlighted the data transmission from transmitter end to DAC input port of RF Data Converter IP and from ADC port to the receiver. The generated IPs are imported and the block design is created using a TCL file which also configures the RF Data Converter settings as per the requirement. From the generated block design, bitstream is created which is used to program the ZCU111.

The parameters of interest in the RF Data Converter IP settings are highlighted in the Fig. 5.6,5.7,5.8.

For the DAC, the DAC1 of tile DAC229 is enabled as highlighted in the Fig. 5.6. The output data symbols generated by the transmitter IP are of complex data type (i.e.



```
R_HB = 2;            % Rate change undertaken by Halfband filter(s)
R_CICcomp = 2;       % Rate change undertaken by CIC compensator
R_CIC = 48;          % Rate change undertaken by CIC filter

% sampling rates at OUTPUTs of the filter sections
fHB = fs * R_HB;
fCICcomp = fHB * R_CICcomp;
fCIC = fCICcomp * R_CIC;
fHB_ssr = fCIC * R_HB;
```

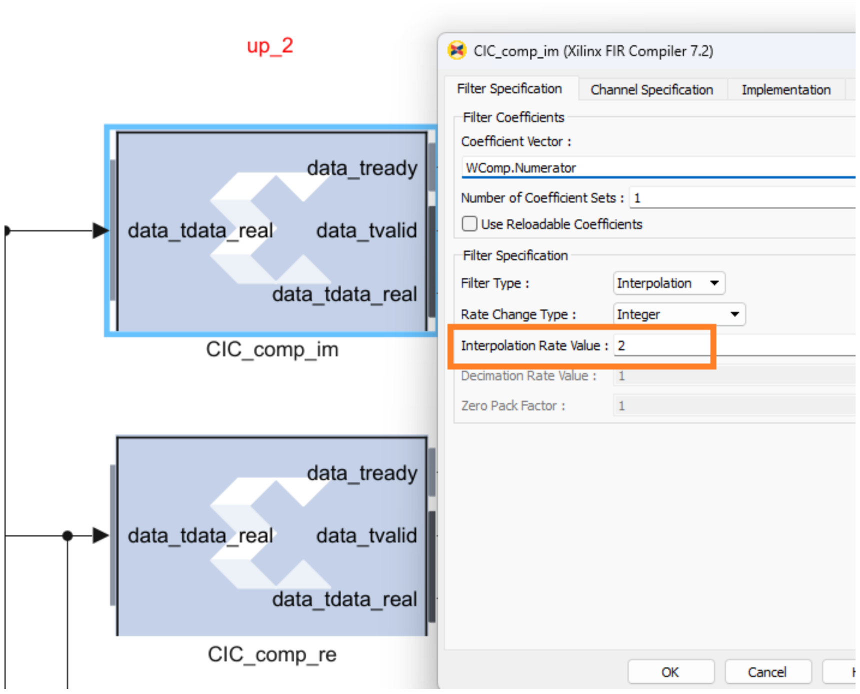
```
CICcomp = dsp.CICCompensationInterpolator(CIC, 'InterpolationFactor',2, ...
    'PassbandFrequency',0.3*fCICcomp, 'StopbandFrequency',0.4*fCICcomp, ...
    'StopbandAttenuation',80, 'PassbandRipple',0.1, 'SampleRate',fCICcomp);
```

Figure 5.2: Filter Coefficients Generation

I/Q symbols). These symbols are received at the input of RF DAC as AXI4 stream values and are further passed as real values (I and Q symbols passed separately). Hence, the analog output type is set as real. The interpolation values available for this specific hardware are 1, 2, 4, and 8 (available interpolation values further depend on the sampling rate set also) and can be selected from the drop down menu. The number of samples per AXI4 stream cycle implies the total number of samples (I and Q symbols counted separately) received in one cycle of AXI stream. This parameter can be understood from the SSR scenario. Since in the original design SSR is employed, two adjacent data symbols each containing its I and Q symbols, are concatenated to form one data packet being transferred to the RF DAC in one clock cycle, i.e., 2I and 2Q symbols. The transmission bus width here is of 64 bits (16 bits of every symbol). Hence, in the settings in Fig.5.6, the number of samples per AXI4 stream cycle value is set as 4. It is to be noted that the analog output data type of RF DAC is real since the I and Q symbols are passed to the channel separately.

Similarly, for RF ADC, ADC0 of tile ADC225 is enabled as shown in the Fig. 5.7. Since the digital output data type is I/Q, the ADC output will have two different data ports, one for I and Q each. It receives input real analog data from the channel and passes 2 I and 2 Q symbols at the two output ports in a single cycle. Hence the number of samples per AXI4 Stream Cycle at the ADC output is set as 2, i.e., samples in a single AXI stream cycle at single output port, 2 I symbols at one and 2 Q symbols at other output port (of the two concatenated data symbols). The transmission bus width here is of 32 bits (16 bits of every symbol). The decimation factor set here is also 8 since the interpolation was also done by 8.

The Fig. 5.8 shows the settings for various input and output clocks in the data converters. The sampling frequency is the sampling rate of data transfer through the channel from DAC to ADC. For RF ADC, it can vary from 1.0 to 4.096 GSPS and for RF DAC, it can vary from 0.5 to 6.554 GSPS. This clock feeds the data converter tiles. The drop down menu shows a list of available frequencies depending on the selected sampling rate. The fabric clock is same as the required AXI4 stream clock and is the frequency of the clock which drives the stream input/output of the DAC/ADC



Filter Coefficients
Coefficient Vector : WComp.Numerator
Number of Coefficient Sets : 1
Use Reloadable Coefficients
Filter Specification
Filter Type : Interpolation
Rate Change Type : Integer
Interpolation Rate Value : 2
Decimation Rate Value : 1
Zero Pack Factor : 1

Figure 5.3: Interpolation Factor in Simulink Model

tiles. Only in the case of SSR, the reference frequency is different from the fabric or AXI4 stream clock. The clock out is the output clock generated within the data converter itself. It is generated from the input reference clock (f_{ref}) itself by using clock dividers. The drop down menu shows the list of available frequencies. The available clock out for RF ADC are $f_{ref}/2$, $f_{ref}/4$, $f_{ref}/8$, $f_{ref}/16$. The available clock out for RF DAC are f_{ref}, $f_{ref}/2$, $f_{ref}/4$, $f_{ref}/8$, $f_{ref}/16$. These output clocks can be used to drive any AXI port. In the current design, these clocks are used to feed the interpolate and decimate IP blocks. The relationship between these frequencies can be illustrated by two simple formulae.

For DAC,

$$F_s = \frac{\textit{AXI4 Stream Clock} * \textit{Interpolation Factor} * \textit{No.of samples per AXI4 Stream Cycle}}{2}$$

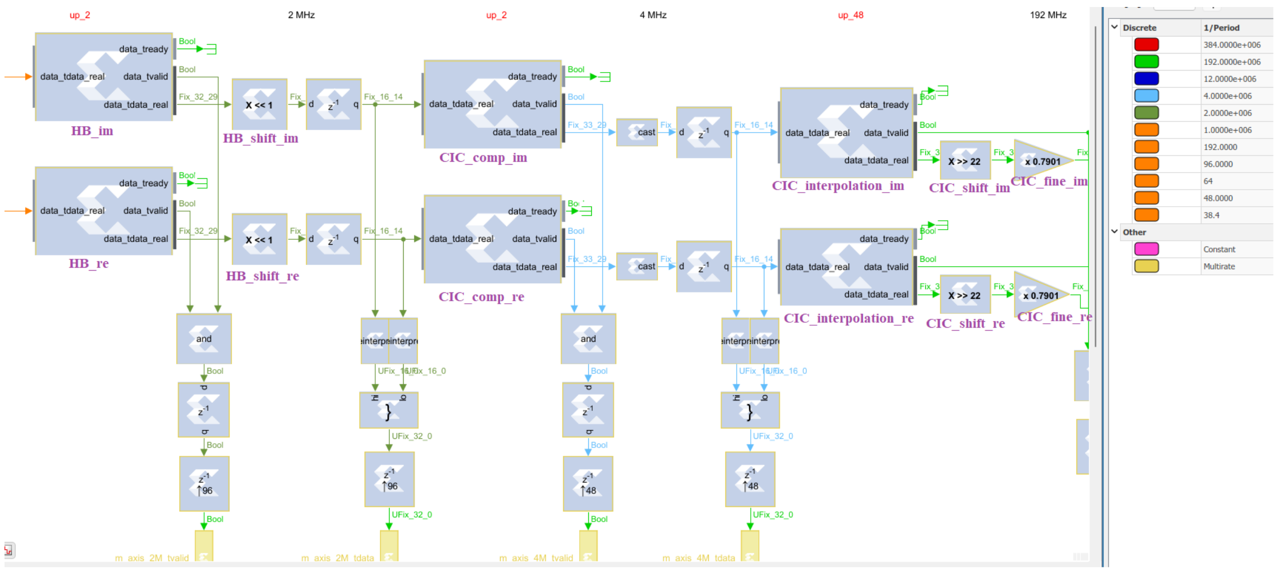
For ADC,

$$F_s = \frac{\textit{AXI4 Stream Clock} * \textit{Decimation Factor} * \textit{No.of samples per AXI4 Stream Cycle}}{1}$$

The 2 in the denominator of F_s for DAC is to compensate for the higher data bus width at the input of DAC (twice as that of ADC).

5.2.1 Architecture

The architecture used for the design is shown in the Fig. 5.9. Here, the Processing System (PS) consisting of Quad ARM Cortex A-53 and Dual ARM Cortex -R6 processors is used to control and configure the IP blocks, viz. Transmitter, Receiver,



Discrete	1/Period
🟥	384.0000e+006
🟩	192.0000e+006
🟦	12.0000e+006
🟦	4.0000e+006
🟩	2.0000e+006
🟧	1.0000e+006
🟧	192.0000
🟧	96.0000
🟧	64
🟧	48.0000
🟧	38.4
Other	
🟪	Constant
🟨	Multirate

Figure 5.4: Sample Time Display

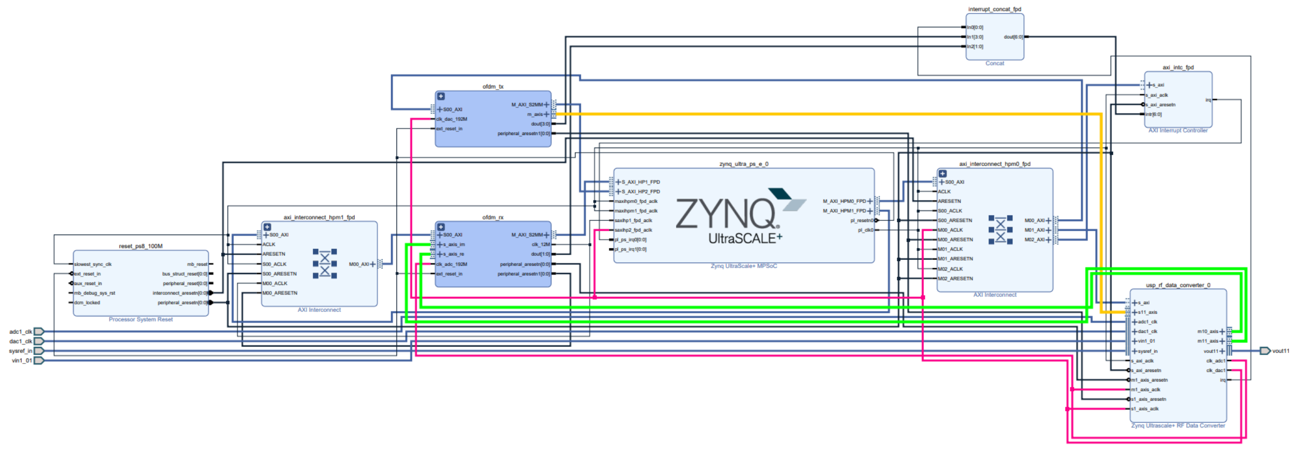
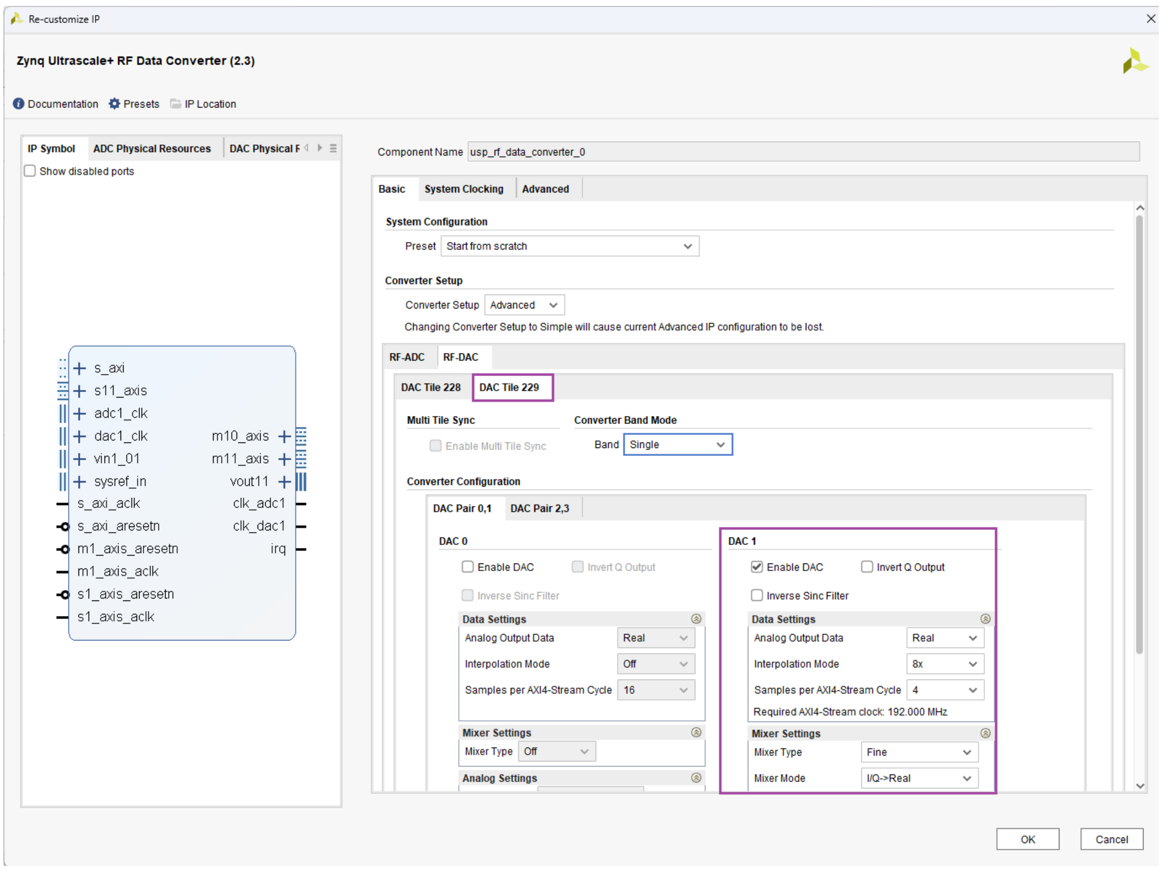


Figure 5.5: IP Integration in Vivado

RF Data Converters IP, AXI DMAs via AXI4-Lite Protocol. Additionally, it reads the input modulation scheme from the user and passes it to the transmitter block using the HP (High Processing port) via AXI Interconnect. On receiving the input modulation scheme from the PS, the transmitter block computes the OFDM symbols, passes them to interpolate block which upconverts to certain frequency as is set by the user and passes them to the RF-DAC converters via AXI DMA as AXI4 Stream protocol. Additionally, the complex data symbols are passed to the PS (for plotting) via AXI DMA which converts the AXI4 Stream protocol to memory mapped.. The RF DAC converters further upconverts to higher data rates and passes to the channel. The RF ADC then receives the data from the channel, downconverts it and then passes to the decimate block via AXI DMA, again as AXI4 Stream protocol. The decimate block, then after down conversion, forwards the data to receiver block which computes the channel estimation values and recovered OFDM symbols. This output is sent to the



Re-customize IP

Zynq Ultrascale+ RF Data Converter (2.3)

Documentation Presets IP Location

IP Symbol | ADC Physical Resources | DAC Physical F

Show disabled ports

Component Name usp_rf_data_converter_0

Basic | System Clocking | Advanced

System Configuration

Preset: Start from scratch

Converter Setup

Converter Setup: Advanced

Changing Converter Setup to Simple will cause current Advanced IP configuration to be lost.

RF-ADC | RF-DAC

DAC Tile 228 | DAC Tile 229

Multi Tile Sync: Enable Multi Tile Sync

Converter Band Mode: Band Single

Converter Configuration

DAC Pair 0,1 | DAC Pair 2,3

DAC 0

- Enable DAC
- Invert Q Output
- Inverse Sinc Filter

Data Settings

Analog Output Data	Real
Interpolation Mode	Off
Samples per AXI4-Stream Cycle	16

Mixer Settings

Mixer Type: Off

Analog Settings

DAC 1

- [✓] Enable DAC
- Invert Q Output
- Inverse Sinc Filter

Data Settings

Analog Output Data	Real
Interpolation Mode	8x
Samples per AXI4-Stream Cycle	4

Required AXI4-Stream clock: 192.000 MHz

Mixer Settings

Mixer Type	Fine
Mixer Mode	I/Q->Real

Figure 5.6: RF DAC settings

PS via AXI DMA which converts the AXI4 Stream protocol to memory mapped.

5.3 Step 3: Clock Configuration File Generation

For the discussed example design, the data sampling at DAC/ADC is done at 384MHz. This clock will be provided by the LMX2594 module on the ZCU111 board itself. To generate this clock, the module needs to be configured to produce this clock frequency at the output and for this, TICS software is used as has already been discussed in Chapter 4. Here we work our way up to achieve the optimal settings values. First step is to find the CHDIV and F_osc values in the Fig. **??**. Given the constraint as mentioned earlier,

$$F_{OSC} \leq 11.5GHz\ for\ CHDIV \geq 8$$

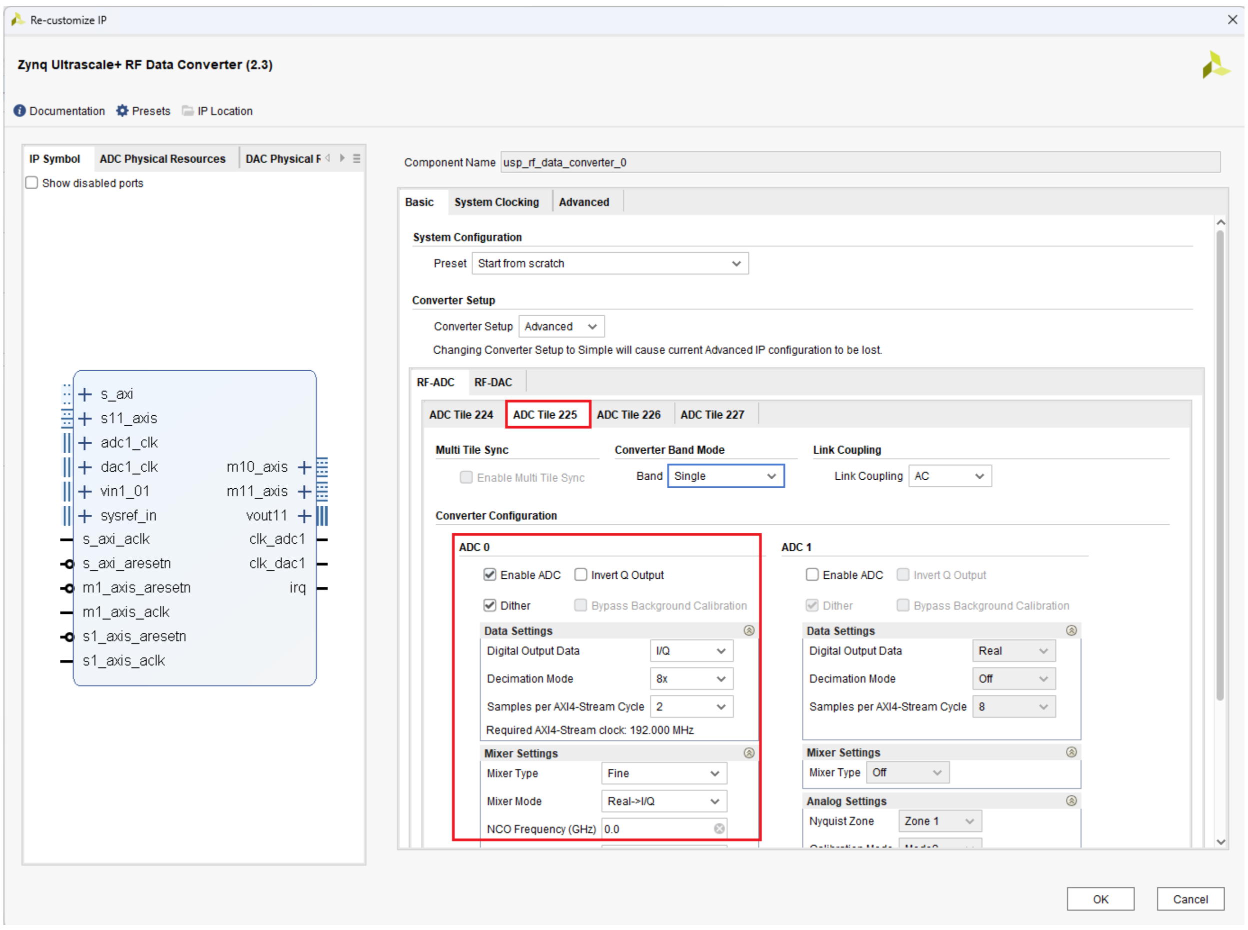
and,

$$F_{OSC} = CHDIV * RF_{OUTA}$$

Assuming,

$$CHDIV \geq 8$$

$$CHDIV * RF_{OUTA} \leq 11.5$$



Zynq Ultrascale+ RF Data Converter (2.3)

Component Name usp_rf_data_converter_0

System Configuration

Preset: Start from scratch

Converter Setup

Converter Setup: Advanced

Changing Converter Setup to Simple will cause current Advanced IP configuration to be lost.

Multi Tile Sync: Enable Multi Tile Sync ☐

Converter Band Mode: Band: Single

Link Coupling: Link Coupling: AC

Converter Configuration

ADC 0

- Enable ADC ✓
- Invert Q Output ☐
- Dither ✓
- Bypass Background Calibration ☐

Data Settings

- Digital Output Data: I/Q
- Decimation Mode: 8x
- Samples per AXI4-Stream Cycle: 2
- Required AXI4-Stream clock: 192.000 MHz

Mixer Settings

- Mixer Type: Fine
- Mixer Mode: Real->I/Q
- NCO Frequency (GHz): 0.0

ADC 1

- Enable ADC ☐
- Invert Q Output ☐
- Dither ✓
- Bypass Background Calibration ☐

Data Settings

- Digital Output Data: Real
- Decimation Mode: Off
- Samples per AXI4-Stream Cycle: 8

Mixer Settings

- Mixer Type: Off

Analog Settings

- Nyquist Zone: Zone 1

Figure 5.7: RF ADC settings

$$CHDIV * 384MHz \leq 11.5GHz$$

$$CHDIV \leq 29.9479$$

The available CHDIV values can be seen from the drop down menu as shown in the Fig. 5.10. The CHDIV$\leq$ 29.947 gives CHDIV value as 24. With this,

$$F_{OSC} = CHDIV * RF_{OUTA}$$

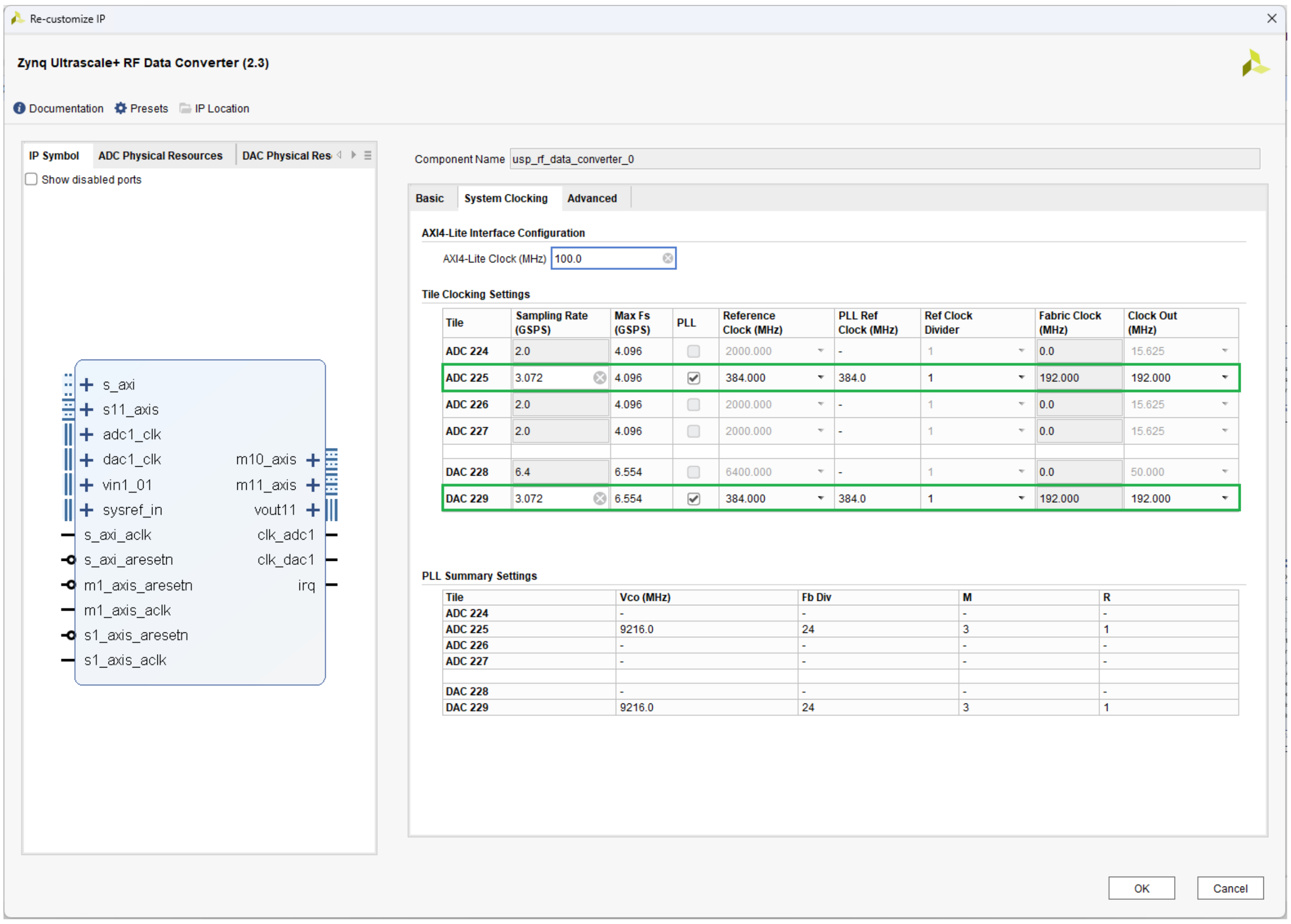
$$F_{OSC} = 9216MHz$$

Next, is to find the N divider, and fraction values. For this, consider the Fig. 5.11 Since,

$$F_{VCO} = F_{pd} * (N + \frac{NUM}{DEN})$$

and F_VCO=9216MHz and F_pd is set as 122.88 MHz from the Fig. 5.12. Here, the doubler can be enabled to double the F_pd (next example).

$$75 = N + \frac{NUM}{DEN}$$


Re-customize IP

Zynq Ultrascale+ RF Data Converter (2.3)

Documentation Presets IP Location

IP Symbol | ADC Physical Resources | DAC Physical Res

☐ Show disabled ports

s_axi
s11_axis
adc1_clk
dac1_clk
vin1_01
sysref_in
s_axi_aclk
s_axi_aresetn
m1_axis_aresetn
m1_axis_aclk
s1_axis_aresetn
s1_axis_aclk
m10_axis
m11_axis
vout11
clk_adc1
clk_dac1
irq

Component Name usp_rf_data_converter_0

Basic | System Clocking | Advanced

AXI4-Lite Interface Configuration

AXI4-Lite Clock (MHz) 100.0

Tile Clocking Settings

Tile	Sampling Rate (GSPS)	Max Fs (GSPS)	PLL	Reference Clock (MHz)	PLL Ref Clock (MHz)	Ref Clock Divider	Fabric Clock (MHz)	Clock Out (MHz)
ADC 224	2.0	4.096	☐	2000.000	-	1	0.0	15.625
ADC 225	3.072	4.096	✓	384.000	384.0	1	192.000	192.000
ADC 226	2.0	4.096	☐	2000.000	-	1	0.0	15.625
ADC 227	2.0	4.096	☐	2000.000	-	1	0.0	15.625
DAC 228	6.4	6.554	☐	6400.000	-	1	0.0	50.000
DAC 229	3.072	6.554	✓	384.000	384.0	1	192.000	192.000

PLL Summary Settings

Tile	Vco (MHz)	Fb Div	M	R
ADC 224	-	-	-	-
ADC 225	9216.0	24	3	1
ADC 226	-	-	-	-
ADC 227	-	-	-	-
DAC 228	-	-	-	-
DAC 229	9216.0	24	3	1

OK Cancel

Figure 5.8: RF System Clocking settings

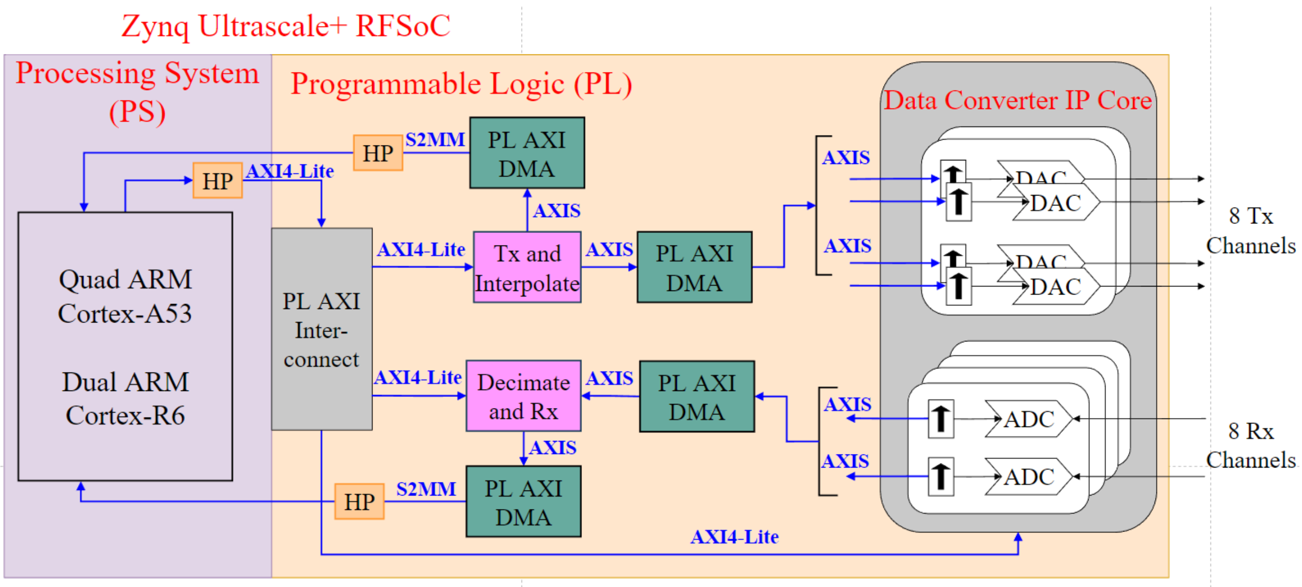


Figure 5.9: RFSoC Architecture for OFDM Model

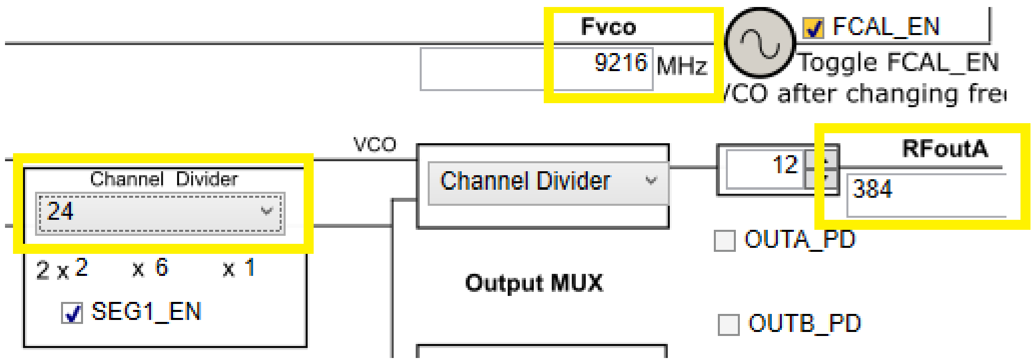


Figure 5.10: TICS Setting Step 1

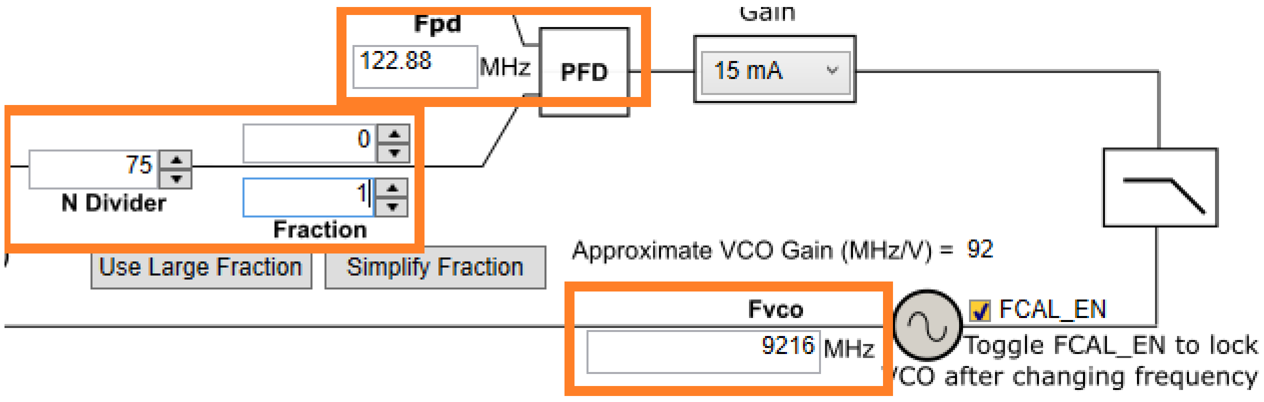


Figure 5.11: TICS Setting Step 2

Since 75 itself is a whole number, N=75, NUM=0, DEN=1. Here, if DEN=0, the denominator will be set as 0 i computing F_VCO which is not possible. Hence, DEN is taken as 1.

Another example can be taken by enabling the doubler. For this, let's consider generating output frequency of 409.6MHz.

$$CHDIV * RF_{OUTA} \leq 11.5$$

$$CHDIV * 409.6MHz \leq 11.5GHz$$

$$CHDIV \leq 28.076$$

The CHDIV$\leq$ 28.076 gives CHDIV value as 24. With this,

$$F_{OSC} = CHDIV * RF_{OUTA}$$

$$F_{OSC} = 9830.4MHz$$

Next, is to find the N divider, and fraction values. Since,

$$F_{VCO} = F_{pd} * (N + \frac{NUM}{DEN})$$

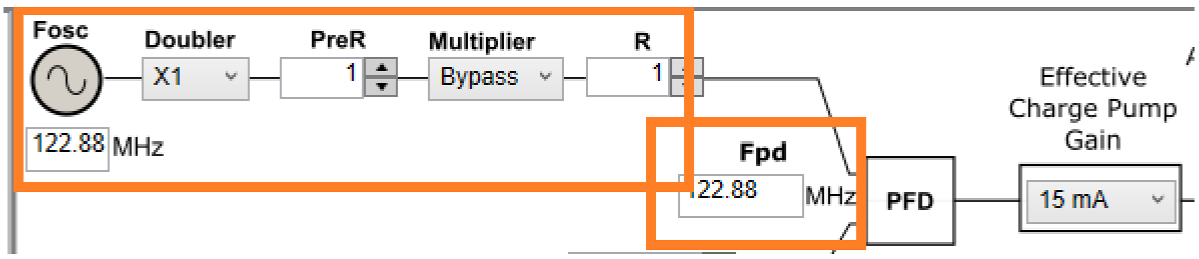

Fosc 122.88 MHz
Doubler X1
PreR 1
Multiplier Bypass
R 1
Fpd 122.88 MHz
Effective Charge Pump Gain 15 mA

Figure 5.12: TICS Setting for Fpd

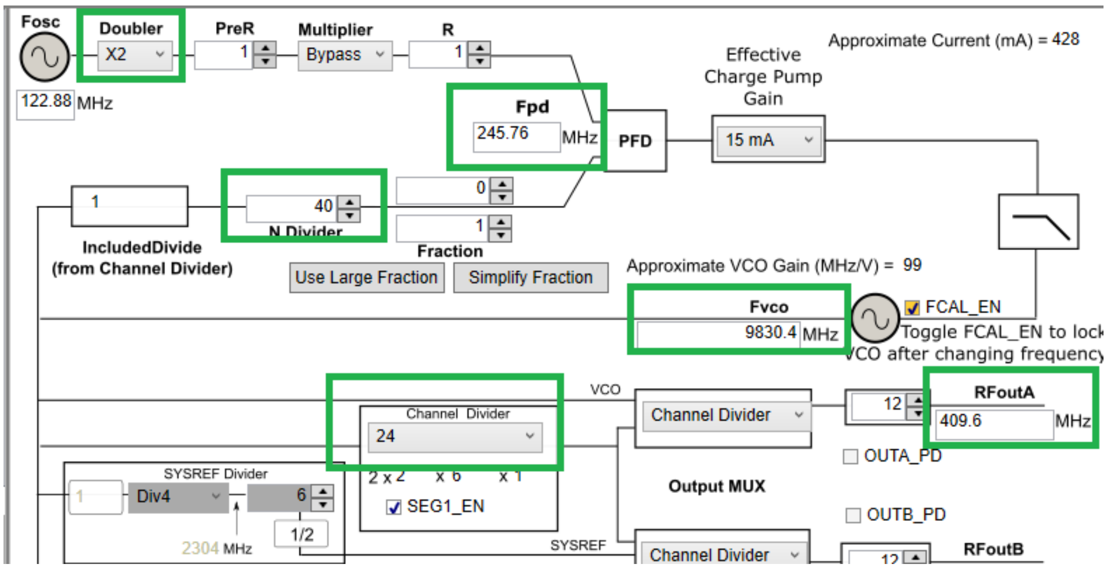

Fosc 122.88 MHz
Doubler X2
PreR 1
Multiplier Bypass
R 1
Fpd 245.76 MHz
Effective Charge Pump Gain 15 mA
Approximate Current (mA) = 428
IncludedDivide (from Channel Divider) 1
N Divider 40
Fraction 0 / 1
Approximate VCO Gain (MHz/V) = 99
Fvco 9830.4 MHz
FCAL_EN ✓
Toggle FCAL_EN to lock VCO after changing frequency
Channel Divider 24
SEG1_EN ✓
SYSREF Divider Div4, 6, 1/2
2304 MHz
RFoutA 12, 409.6 MHz
OUTA_PD ☐
OUTB_PD ☐
RFoutB 12

Figure 5.13: TICS Setting by enabling the doubler

and F_VCO=9830.4MHz. If doubler is enabled, F_pd=245.76MHz,

$$N + \frac{NUM}{DEN} = \frac{9830.4}{245.76} N + \frac{NUM}{DEN} = 40$$

From this, N=40, NUM=0, DEN=1.

5.4 Step 4: PYNQ Setup

For final step of running the design on hardware, PYNQ is used. It provides an interactive python kernel on the Jupyter labs which are integrated with the web browser running directly on the ARM processor. For any design to run on the board, an overlay file (drivers) and the bitstream file is required. Additional .pynq files are required which are simple python scripts running. The bitstream generated from the vivado is programmed on the board by a function defined in overlay itself. The overlay file is additionally used to configure the DACs/ADCs, clock the RF data converters using the clock configuration file generated from the TICS SW Pro, and define various functions required for plotting/displaying the output. For the channel, a simple loopback connection is used to connect the enabled DAC and ADC as

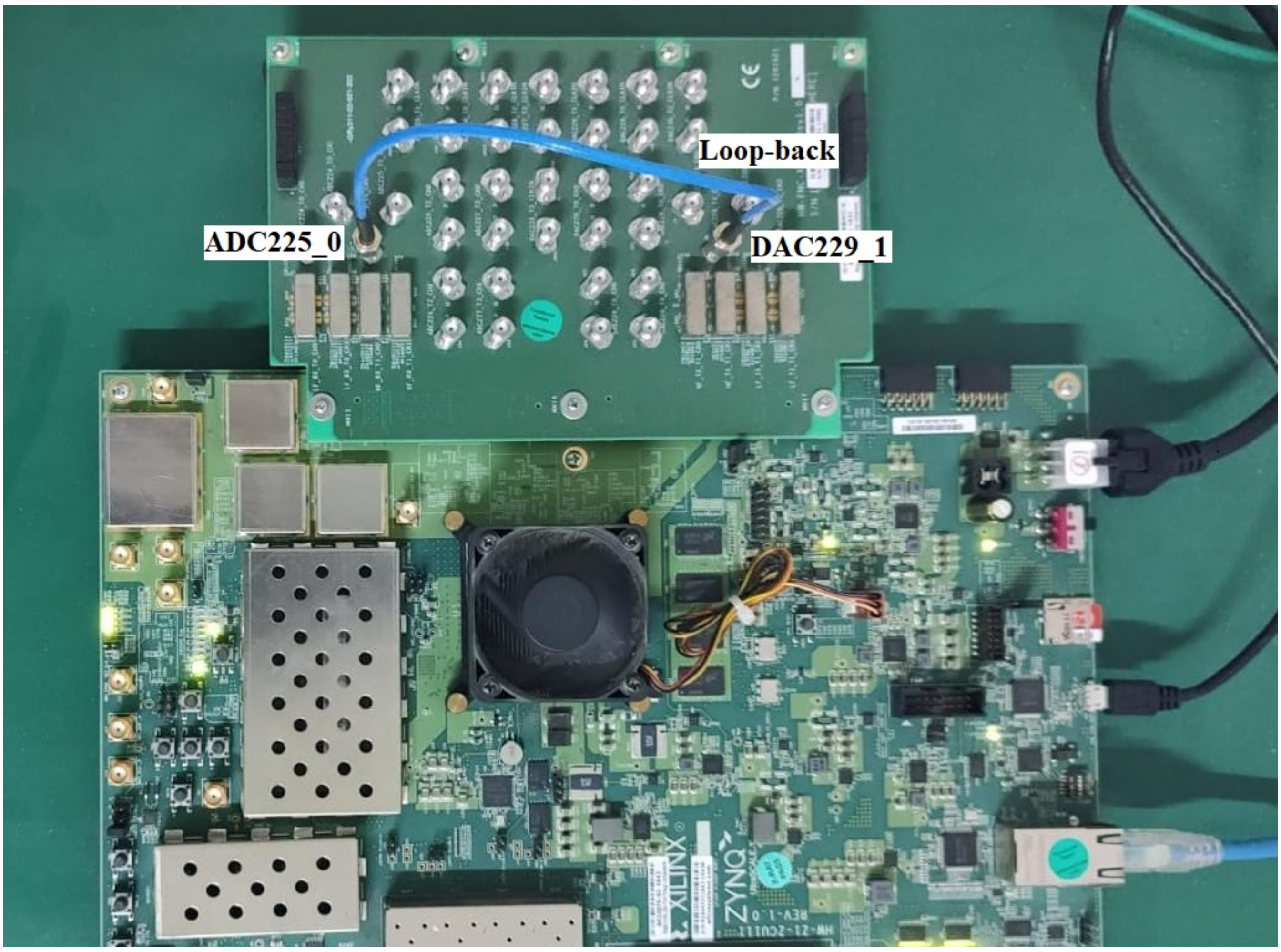


Figure 5.14: ZCU111 Setup

shown in the Fig. 5.14. The reference clock, sampling rate, and the clock configuration generated from TICS are required to be set according to the design. For this, the pre-defined function *set_all_ref_clks(<reference clock value (in MHz)>)* is used which invokes the clock configuration values and programs the LMX2594 module. The PYNQ provides a GUI for visualizing the received data in form of plots. The functionality of the design can be verified by obtaining the constellation plots of the recovered data (output from the receiver block) for various modulation schemes.

Chapter 6

Results

The OFDM base model was modified for varying RF frequencies and tested on the hardware for its functionality. The values for which the design has been tested are listed in the Tab. 6.1. Only the changes at the transmitter end (interpolation and RF DAC) are listed here. Correspondingly, similar changes are done at the receiver end (decimator and RF ADC) too. In the designs original and 1, SSR filter is also used which means the incoming data rate is 192 MHz but the RF DAC ADC can be clocked at 384 MHz. Since adjacent data symbols in time frame are concatenated as a single block in SSR, the incoming data bus width at DAC is twice (64 bits) as compared to the other cases (32 bits). For the remaining designs, the SSR filter is removed entirely. For designs 2, 3 and 4, the third filter interpolation factor is modified. For design 5, second filter interpolation factor is changed from 2 to 6 and in design 6, the second filter is removed.

To verify the functionality of the design, the constellation plot of the demodulated received symbols is observed for all the modulation schemes. The constellation plot for some of the modulation is shown in the Fig. 6.1-6.10 below.

Additionally, the received symbols were demodulated in the PS itself at the receiver end to compute symbol error rate (SER). The SER comprison for designs original and 1 (with SSR) is shown in the Fig. 6.11.

Design	**Interpolation by filters**	**Sampling Frequency of DAC (in MHz)**	**Interpolation by RF DAC**	**Sampling frequency (in GHz)**
Original	2, 2, 48, 2	384	8	3.072
1	2, 2, 48, 2	384	4	1.536
2	2, 2, 48	192	8	1.536
3	2, 2, 96	384	8	3.072
4	2, 2, 60	240	8	1.920
5	2, 6, 20	240	8	1.920
6	2, 120	240	8	1.920

Table 6.1: List of values for which the design has been tested

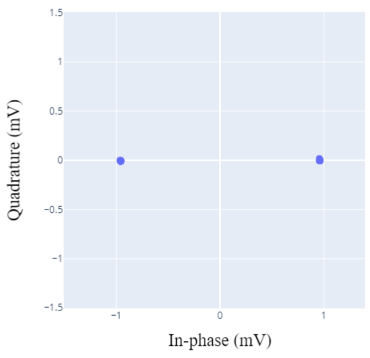


Figure 6.1: BPSK

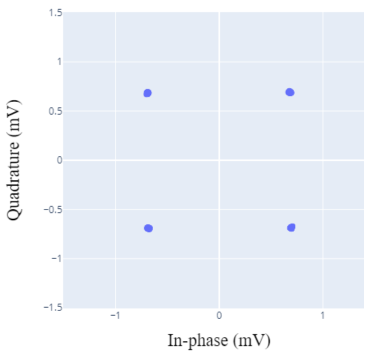


Figure 6.2: QPSK

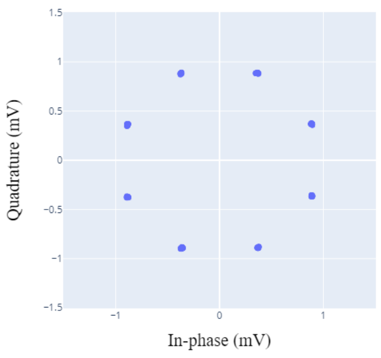


Figure 6.3: 8-PSK

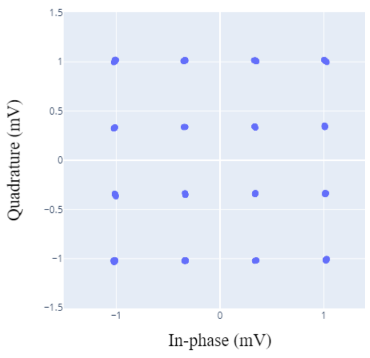


Figure 6.4: 16-QAM

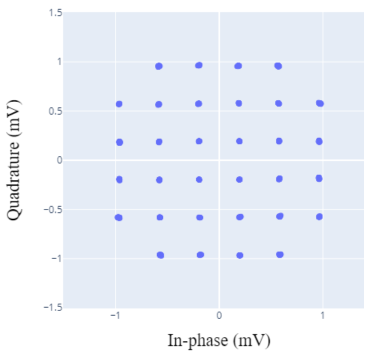


Figure 6.5: 32-QAM

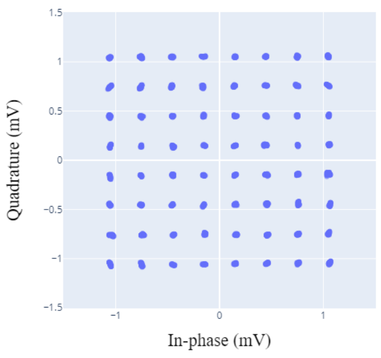


Figure 6.6: 64-QAM

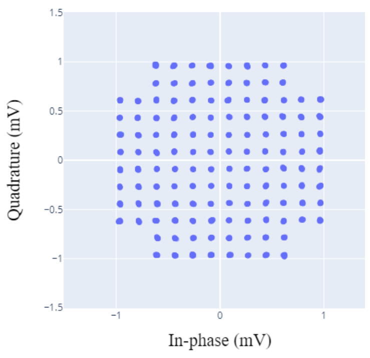


Figure 6.7: 128-QAM

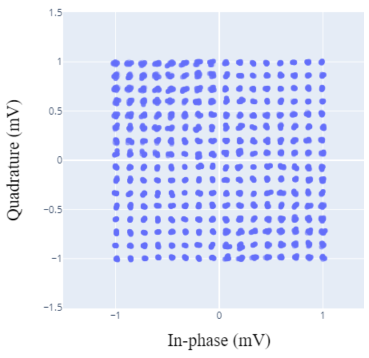


Figure 6.8: 256-QAM

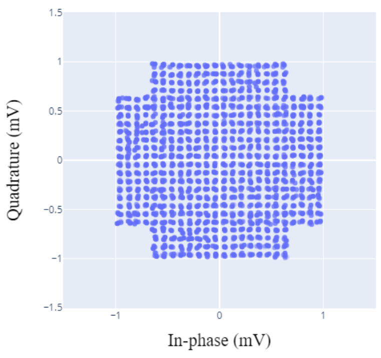


Figure 6.9: 512-QAM

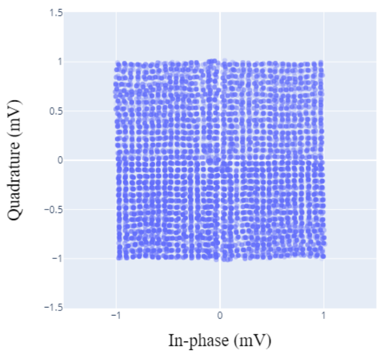


Figure 6.10: 1024-QAM

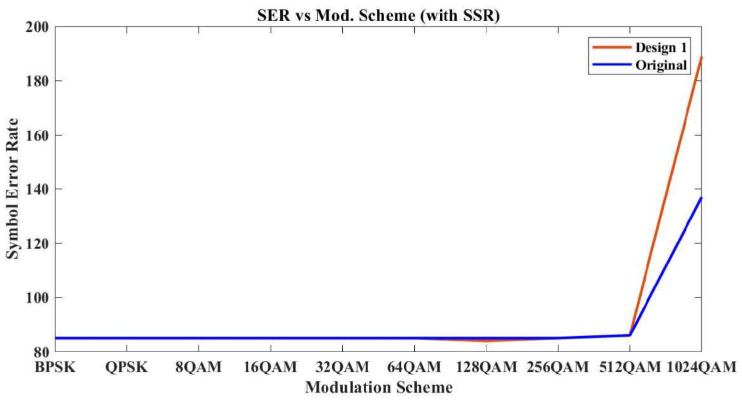


Figure 6.11: SER comparison with SSR

Chapter 7

Future Work and Conclusion

The OFMD PHY Layer example design used in this work has, in addition to upconversion/downconversion by RF data converters, interpolate and decimate blocks which are implemented in the PL. The work done in this thesis focuses on these blocks as well in addition to exploring the RF data converters. These blocks are implmented by using series of filters and their respective interpolate/decimate factors are modified tp genarate differeneg carrier frequencies of the data. Additionally, the parameters associated with RF data converters like their carrier frequency, sampling rates, interpolation/decimation factors, etc. are also explored in detail. Apart from this, the TICS software used to generate the clock configuration files to program the LMX2594 module which feeds the RF DACs/ADCs is also explored in depth. The design is finally tested for its functionality on the PYNQ by observing the received constellation plot corresponding to any modulation scheme. All these understanding would be very helpful to other researchers or engineers to integrate any other baseband design with RF DAC/ADC.

The future work in this thesis would be to create well-detailed handouts and video tutorials on multiple ofdm examples and to create application notes of interpolate and decimate blocks, DAC/ADC and clock configuration.

References

[1] Xilinx (October 2, 2018), *ZCU111 Evaluation Board User Guide*, UG1271 (v1.2)

[2] Xilinx (May 29, 2019), *Zynq UltraScale+ RFSoC RF Data Converter Evaluation Tool (ZCU111)*, UG1287 (v2019.1)

[3] Xilinx (November 30, 2020), *Zynq UltraScale+ RFSoC RF Data Converter v2.4 Gen 1/2/3 LogiCORE IP Product Guide*, PG269 (v2.4)

[4] OFDM Model: https://github.com/strath-sdr/rfsoc_ofdm/. Last Accessed: April 1, 2022

[5] Texas Instruments (April 2019), *LMX2594 15-GHz Wideband PLLATINUM™ RF Synthesizer With Phase Synchronization and JESD204B Support*, SNAS696C

[6] Texas Instruments (September 2016), *LMK04208 Low-Noise Clock Jitter Cleaner with Dual Loop PLLs*, SNAS684

[7] Xilinx (October 26, 2015), *Vivado Design Suite Tutorial Model-Based DSP Design Using System Generator*, UG948 (v2015.3)

[8] Xilinx (March 20, 2013), *Vivado Design Suite Tcl Command Reference Guide*, UG835 (v 2013.1)

[9] Abaco Systms, *Technologies for responding to rapid developments in cognitive RF and EW*