Real-time DSP-enabled digital subcarrier cross-connect (DSXC) for optical communication networks

By

Tong Xu

Submitted to the graduate degree program in the Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Chair: Dr. Rongqing Hui

Dr. Christopher Allen

Dr. Erik Perrins

Dr. Esam Eldin Aly

Dr. Jie Han

Date Defended: August 20th, 2019

The dissertation committee for Tong Xu certifies that this is the approved version of the following dissertation:

Real-time DSP-enabled digital subcarrier cross-connect (DSXC) for optical communication networks

Chair: Dr. Rongqing Hui

Date Approved: August 23rd, 2019

Abstract

Elastic optical networking (EON) is intended to offer flexible channel wavelength granularity to meet the requirement of high spectral efficiency (SE) in today's optical networks. However, optical cross-connects (OXC) and switches based on optical wavelength division multiplexing (WDM) are not flexible enough due to the coarse bandwidth granularity imposed by optical filtering. Thus, OXC may not meet the requirements of many applications which require finer bandwidth granularities than that carried by an entire wavelength channel.

In order to achieve highly flexible and fine enough bandwidth granularities, electrical digital subcarrier cross-connect (DSXC) can be utilized in EON. As presented in this dissertation, my research work focuses on the investigation and implementation of real-time digital signal processing (DSP) enabled DSXC which can dynamically assign both bandwidth and power to each individual sub-wavelength channel, known as subcarrier. This DSXC is based on digital subcarrier multiplexing (DSCM), which is a frequency division multiplexing (FDM) technique that multiplexes a large number of digitally created subcarriers on each optical wavelength. Compared with OXC based on optical WDM, DSXC based on DSCM has much finer bandwidth granularities and flexibilities for dynamic bandwidth allocation.

Based on a field programmable gate array (FPGA) hardware platform, we have designed and implemented a real-time DSP-enabled DSXC which uses Nyquist FDM as the multiplexing scheme. For the first time, we demonstrated real-time DSP enabled real-time DSXC which uses resampling filters for channel selection and frequency translation. This circuit-based DSXC supports flexible and fine data-rate subcarrier channel granularities, offering a low latency data plane, transparency to modulation formats, and the capability of compensating transmission impairments in the digital domain. The experimentally demonstrated $\mathbf{8} \times \mathbf{8}$ DSXC makes use of a Virtex-7 FPGA platform, which supports any-to-any switching of eight subcarrier channels with mixed modulation formats and data rates. Digital resampling filters, which enable frequency selections and translations of multiple subcarrier channels, have much lower DSP complexity and reduced FPGA resources requirements (DSP slices used in FPGA) in comparison to the traditional technique based on I/Q mixing and filtering.

We have also investigated the feasibility of using distributed arithmetic (DA) for real-time DSXC to completely eliminate the usage of DSP slices in FPGA implementation. For the first time, we experimentally demonstrated the implementation of real-time frequency translation and channel selection based on the DA architecture in the same FPGA platform. Compared with resampling filters that leverage multipliers, the DA-based approach eliminates the need of DSP slices in the FPGA implementation and significantly reduces the hardware cost. In addition, with a processing latency that equals to a few clock cycles, a DA-based resampling filter is significantly faster when compared to a conventional direct-structured FIR filter whose overall latency is proportional to the filter order. The DA-based DSXC is, therefore, able to achieve not only the improved spectral efficiency, programmability of multiple orthogonal subcarrier channels, and low hardware resources requirements, but also much reduced cross-connect switching latency when implemented in a real-time DSP hardware platform. This reduced latency can be critically important for time-sensitive applications such as 5G mobile fronthaul, cloud radio access network (C-RAN), cloud-based robot control, tele-surgery and network gaming.

Acknowledgments

First of all, I would like to sincerely thank my supervisor, Dr. Rongqing Hui, for supporting me in conducting research on this challenging and interesting topic. Without his support and patient supervision, I wouldn’t have been able to overcome the difficulties and make progress in my research.

I would also like to thank Dr. Christopher Allen, Dr. Erik Perrins, Dr. Esam Eldin Aly and Dr. Jie Han, who served as my committee members.

I also want to thank Mr. Robert Young, Mr. Daniel DePardo and Ms. Paulette Place for helping me in setting up the hardware platform.

Thanks to my colleagues and friends Govind Vedala, Mustafa Al-Qadi and Kishanram Kaje for the happy times of discussion and collaboration in our research lab.

Finally, I would like to thank my parents and my sister. I wouldn’t have been able to complete my PhD without their support.

Table of Contents

List of Figures viii

List of Tables x

Chapter 1: Introduction 1

1.1 Motivation and background of research 1

1.2 Digital subcarrier cross-connect (DSXC) 6

1.3 Digital subcarrier multiplexing techniques 8

1.4 Overview of proposed work 11

1.4.1 Real-time DSP hardware platform 11

1.4.2 Channel selection and frequency translation in DSXC 11

1.4.3 Implementation of DSXC based on hardware platform 12

Chapter 2: Real-time DSP hardware platform 13

2.1 Hardware Platform 13

2.2 Data Interfaces 16

2.2.1 JESD204B interface 17

2.2.2 Design in Xilinx Vivado 20

2.3 Considerations in FPGA Implementation 21

Chapter 3: Real-time DSP-enabled DSXC based on resampling filters 24

3.1 Digital subcarrier cross-connect based on FPGA 24

3.2 Frequency translation techniques 26

3.2.1 I/Q mixing and filtering 26

3.2.1 Resampling Filters 28

3.3 Resource utilization of frequency translation 36

3.4 Experiments 40
3.5 Conclusion 47
Chapter 4: DA based real-time DSP-enabled DSXC........ 49
4.1 Distributed Arithmetic 49
4.1.1 Principle of DA 51
4.1.2 Polyphase decomposition of DA resampling filter 54
4.2 Experiment........ 58
4.3 Resource requirement and discussion........ 65
4.3.1 Resource Utilization........ 65
4.3.2 Latency........ 70
4.3.3 Power Consumption........ 72
4.4 Conclusion 72
Chapter 5: Conclusion and Future work 74
5.1 Conclusion 74
5.2 Future work........ 74
References........ 76
Acronyms........ 85
Appendix I 88

List of Figures

Figure 1.1 Digital subcarrier cross-connect……………………………………………….…...6

Figure 1.2 (a) Nyquist pulse in time domain (b) spectrum of a single Nyquist pulse (c) spectrum of Nyquist FDM………………………………………………………………..…… 9

Figure 1.3 Principle of OFDM (a) single pulse shape (b) spectrum of a single OFDM symbol (c) spectrum of OFDM……………………………………………………………..…..10

Figure 2.1 (a) block diagram and (b) photo of DSCM testbed………………………………14

Figure 2.2 JESD204B standard…………………………………………………………..…..17

Figure 2.3 Block diagram of real-time Transmitter based on FPGA and DAC…………………18

Figure 2.4 Block diagram of High-Speed Receiver based on FPGA and ADC…………………20

Figure 3.1 Block diagram of digital subcarrier cross-connect (DSXC) based on FPGA……..…24

Figure 3.2 Frequency translation through I/Q mixing and filtering…………………………...…27

Figure 3.3 (a) DDC through BPF and down-sampling, (b) DDC through interpolation BPF, (c) DUC through up-sampling and BPF, (d) DUC through decimation BPF………………………29

Figure 3.4. (a) DFT of $y[n]$, and (b) DFT of $z[n]$ for down sampling.……………………..…29

Figure 3.5. (a) DFT of $[n]$, and (b) DFT of $y[n]$ for up-sampling ……………………….…….30

Figure 3.6 (a) spectrum of a[n], (b) DSP block of DSXC, and (c) spectrum of b[n] ……..….…37

Figure 3.7 FPGA resource cost of a 4×4 DSXC………………………………………….…37

Figure 3.8 Experimental setup………………………………………………………...…40

Figure 3.9 Spectra of output signal: (a) at optical receiver output which is DSXC input (b) after DSP platform but without switching (c) after DSP platform with switching assignment of DSXC1 (d) after DSP platform with switching assignment of DSXC2……………………....43

Figure 3.10. Signal EVM of recovered subcarrier channels……………………………………..44

Figure 3.11 Effects vs FIR filter order …..……………………………………………….……46

Figure 4.1 A super sample rate FIR filter with a degree of parallelism of 4……………………..…56

Figure 4.2 Block diagram of an interpolation FIR filter…………………………………………57

Figure 4.3 Block diagram of a decimation FIR filter…………………………………….………57

Figure 4.4 Experimental setup……………………………………………………………...…58

Figure 4.5 Spectrum of (a) output of optical receiver (b) output of DAC without cross-connect switching (c) output of DA-DSXC1 (d) output of DA-DSXC2…………………………………60

Figure 4.6 Signal EVM of recovered subcarriers……………………………………………..…61

Figure 4.7 (a) EVM vs Passband Ripple, and (b) EVM vs Stopband Attenuation…………...…63

List of Tables

Table 2.1 Available FPGA resources on Virtex 5 and Virtex 7……………………………………15

Table 2.2 Specifications of ADC12J4000 and DAC38RF82……………………………………16

Table 2.3 Specifications of Optical Tx and Rx…………………………………………………..16

Table 3.1 AWG generated input signal to 8 × 8 DSXC…………………………………………42

Table 3.2 FPGA resource cost of DSXC…………………………………………………………..45

Table 4.1 FPGA resource utilization of DA DSXC IP and MULT DSXC IP…………………...68

Table 4.2 FPGA resource utilization of DA DSXC and MULT DSXC…………………………69

Chapter 1: Introduction

1.1 Motivation and background of research

Due to the ever-increasing data traffic in today's optical networks and the demand for high data rates, improving spectral efficiency (SE) in optical communication systems and networks is of the essence. Compared with a traditional wavelength division multiplexing (WDM) system of fixed 50GHz wavelength grid, elastic optical networking (EON) offers more flexibility, with channel wavelength granularity down to 12.5GHz or lower, which may yield tangible SE improvement in optical networks [1, 2]. Both bandwidth and channel allocation are flexible in EON and can be chosen to best accommodate the modulation formats of choice, transmission distance, system capacity, and number of required channels [3-5]. However, due to the limited spectral selectivity of extant optical filters, further reduction of channel wavelength granularity can prove to be challenging in the optical domain. Yet, many applications could benefit from finer channel bandwidth granularities below 10GHz. Optical domain EON alone may not suffice to achieve the fine bandwidth allocation which may be required in some access and metro area network deployments. For this and other reasons conventional solutions combine the use of optical circuits with electronic packet switching technologies (Ethernet, IP, PON), i.e., multiple connections are multiplexed together by interleaving their data packets in time, thus filling up the relatively large bandwidth of the optical circuit.

Subcarrier multiplexing (SCM) can provide much finer granularity by multiplexing a large number of subcarrier channels in the electrical domain [6, 7]. Subcarrier circuits can be flexibly multiplexed and individually switched electronically, offering dedicated circuits to the application down to MHz of bandwidth. Earlier SCM solutions are analog. While the radio frequency (RF) analog filter solutions offer much better spectral selectivity compared to optical filters, the transition between passband and stopband in the transfer function of an RF filter still may not be sharp enough to separate closely spaced subcarrier channels. As a result, analog based SCM usually requires sufficiently large spectral guard-bands between adjacent subcarriers, resulting in a suboptimal solution. In addition, the bandwidth and the central frequency of high order RF filters are usually not dynamically adjustable after they are built, and thus analog SCM systems tend to be static and not suitable for dynamic switching.

Thanks to the rapid development of CMOS-based digital electronics, high speed analog to digital converters (ADC), digital to analog converters (DAC) and digital signal processing (DSP) hardware is widely available nowadays [8]. Processing high data rate signals in the digital domain has become practical and offers many advantages compared to traditional analog techniques [9-12]. For example, high order digital filters can be designed to achieve nearly ideal transfer functions, along with dynamically reconfigurable of roll-off rate, bandwidth, and central frequency. Digitally generated and processed subcarrier channels are referred to as digital subcarrier multiplexing (DSCM). DSCM offers a high degree of flexibility because the applied DSP algorithms can be reconfigurable, and yields high spectral efficiency because minimum spectral guard-band is required between adjacent subcarriers. Real-time generation of DSCM signals based on either high order Nyquist filters, or orthogonal frequency division multiplexing (OFDM) has been demonstrated using FPGA platform [13, 14]. DSP-enabled real-time reconfigurable optical add/drop multiplexing (ROADM) technologies have also been demonstrated using FPGA platforms [15, 16]. In addition to being used as a modulation format for optical signal transmission [17], DSCM can also be used to carry orthogonal channel which can be switched individually by digital subcarrier cross-connect (DSXC) devices as introduced in [18, 20]. A DSXC-based network is a circuit switching solution in which subcarrier channels are individually routed end-to-end to provide dedicated circuits with custom data rates. Compared with optical domain cross-connect (OXC) based on wavelength channels, DSXC in the electronic domain can provide a more flexible and finer data rate granularity, which can help maximize the network spectral efficiency. In comparison with packet based routers, DSXC provides dedicated bandwidth to users without the requirement of packet buffering and forward engine, resulting in a deterministic switching latency [21, 22].

An application example of DSCM is the 5G wireless network fronthaul, which is the network segment between remote radio head (RRH) and central office (CO) [23]. In some solutions common public radio interface (CPRI) is the protocol used in the mobile fronthaul with digital radio over fiber (DRoF) transmitted using on-off keying (OOK). This approach is known to require a relatively high data rate in the fronthaul compared to the radio data rate, due to the sampling of the radio wave and the required high-resolution sampling of ADC and DAC. In DRoF, the received analog wireless waveforms are digitized and encoded into digital bits for transmission. In this analog to digital conversion process, the data rate and thus the bandwidth required for transmission over the fronthaul is scaled roughly by b times higher than the original analog signal bandwidth where b is the bit resolution of the ADC. For example, for 8 channels of 20MHz LTE signals using 40MS/s ADC sampling rate at 15-bit resolution for each I- and Q-component of the complex RF waveform, the digital data rate will be approximately $8 \times 40 \times 15 \times 2 = 12{,}000$ Mb/s (Reference [24] has more detailed data rate estimation taking into account control words and line coding). Further considering multiple antenna elements for MIMO beam forming, the required data rate can easily reach up to 100Gb/s.

In order to improve the spectral efficiency of the fronthaul, analog radio over fiber (ARoF) has been proposed and actively investigated [25, 26]. Compared with DRoF, ARoF can transmit the same radio wave using a narrower bandwidth in the fronthaul. Considering that 5G fronthaul is expected to support numerous RRHs, DSCM would be an efficient technology for aggregating and de-aggregating multiple radio waves while offering spectrum flexibility and efficiency at the same time. The DSCM solution would also offer the ability to compensate for transmission impairments in the digital domain [27, 28]. Recently, an ARoF transmission of a multicarrier signal with a carrier frequency of 60GHz, and an aggregation of hundreds of subcarriers to occupy 152MHz of total bandwidth, has been demonstrated experimentally over 25km of single mode fiber (SMF) [29]. In addition, the deployment of DSXC nodes in the fronthaul makes DSCM channel aggregation, de-aggregation and routing dynamically programmable, which would be desirable in a fronthaul connecting multiple RRHs and COs.

For real-time implementation of the DSXC key functions efficient utilization of DSP resources is a major concern. While application-specific integrated circuit (ASIC) is commonly used in commercial communication equipment, FPGA represents a more flexible platform for prototyping and testing the DSP algorithms that are required in DSXC.

We have demonstrated a real-time DSP-enabled DSXC based on resampling digital filters to achieve frequency translation and channel selection of subcarriers [30], as described in Chapter 3. We demonstrated the first implementation of a real-time DSXC node, realizing one of its basic functionalities (switching individual subcarriers in frequency) using digital resampling filters, and experimentally assessing the DSXC node (deterministic) latency to be less than 1µs. This latency is mainly determined by the order of Finite Impulse Response (FIR) filters and the clock period of the digital circuit. Compared to commodity packet switches, DSXC may therefore provide a simple and cost-effective switching solution that achieves zero-jitter even in the presence of high link utilization.

Although these resampling filters reduce DSP resource utilization compared to the traditional frequency translation scheme based on I/Q mixing and filtering, they still heavily rely on multipliers which are usually implemented as expensive DSP slices in FPGA. To overcome this drawback, we demonstrate a more efficient technique to realize real-time frequency translation and channel selection of DSCM channels based on distributed arithmetic (DA) [31], as described in Chapter 4. No digital multipliers are required when using DA, thus completely eliminating the need for DSP slices in the FPGA. In addition, the DA-based DSXC reduces DSP-induced latency down to only a few clock periods, which is independent of the applied digital filter order. DA has been used to implement digital filters for Nyquist pulse generation in fiber-optic transmitter [9, 14], but has not been used for digital subcarrier frequency translation and channel selection. By applying DA algorithms to implement resampling filters, we show that DSXC key functionalities can be implemented in an FPGA platform without requiring any DSP slice. To our best knowledge, this is the first realization of a DA-based DSXC, which is capable of performing bandwidth flexible switching and routing with improved hardware efficiency, low latency, and transparency to signal modulation formats.

1.2 Digital subcarrier cross-connect (DSXC)

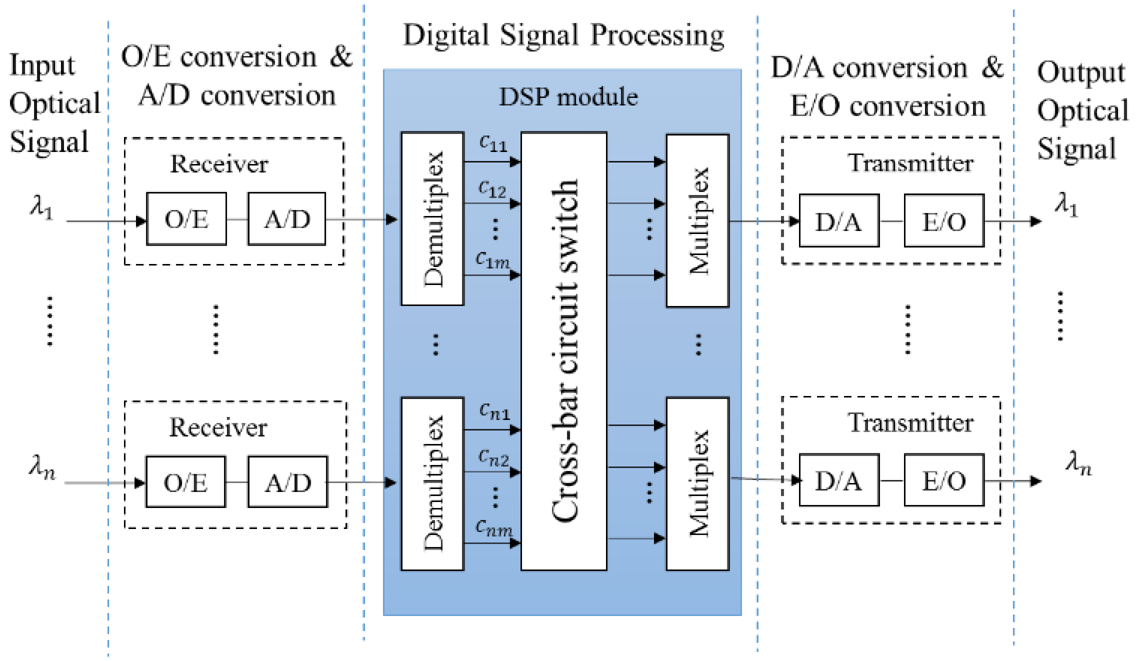


Figure 1.1 Digital subcarrier cross-connect

Figure 1.1 shows the block diagram of a generic DSXC node [30]. Input to the DSXC node are n optical signals, each consisting of m subcarrier channels. Each optical signal is received by a receiver, which performs optical-to-electrical conversion (O/E) through an optical receiver, and analog-to-digital conversion through an ADC. The digitized signal from each receiver is sent to the DSP module for processing. In the DSP module, each multicarrier signal is de-multiplexed into multiple subcarriers and sent into a cross-bar switch to be routed to any output port for multiplexing. The multiplexer aggregates multiple subcarriers and sent them into the targeted transmitter. The transmitter performs digital to analog conversion through a DAC, and electrical to optical (O/E) conversion obtained by an electro-optic modulator. In the shown DSXC node architecture, each digital subcarrier channel c_{ij}, (i = 1, 2....n, and j = 1, 2 ...m) in any input wavelength can be routed to any output wavelength λ_i and subcarrier frequency slot through a cross-bar circuit-switch.

The optical system can be either coherent or direct-detection. The multiplexing method can be through either high order Nyquist filters or OFDM. Digital compensation techniques, such as chromatic dispersion compensation and electronic circuit frequency roll-off compensation, can be performed in the digital domain. In the design of this DSXC, we use high order Nyquist filters for DSCM, which provide the flexibility of using unequal spectral bandwidth and distinct modulation formats to be assigned to each subcarrier channel.

In order to be able to route any subcarrier channel of any input wavelength to any subcarrier channel of any output wavelength, frequency translation and channel selection of individual subcarrier are two critical functions in a DSXC. Frequency translation includes frequency down conversion and up conversion of each subcarrier channel. The frequency down conversion can be achieved through decimation filter, in which the decimation factor is the ratio of the input rate to the output rate. The frequency up conversion can be achieved through an interpolation filter, in which the interpolation factor is the ratio of the output rate to the input rate. Since resampling filter includes decimation filter and interpolation filter, both decimation factor and interpolation factor are named as resampling factor. According to Nyquist criterion, the available analog bandwidth of each wavelength channel is limited to half of the ADC's sampling rate. This total bandwidth can be subdivided among many frequency slots (FS). The bandwidth of each FS is given by the total available bandwidth divided by the resampling factor when digital resampling filters are used. In this process, any subcarrier in a FS is first down-converted to the lowest frequency FS through a decimation filter, and then up-converted to any targeted FS through an interpolation filter [23]. During the down-conversion process, a decimation filter, whose frequency response has a passband targeted at a particular FS, selects the subcarrier in this FS and down converts it to the first FS. During the up-conversion process, an interpolation filter, whose frequency response has a passband targeted at a particular FS, selects the up-sampled copy of subcarrier in this FS and rejects all copies in other FSs. Theoretically, decimation is equivalent to a cascaded process of filtering and down sampling, whereas interpolation is equivalent to a cascaded process of up sampling and filtering. Further information about these two procedures can be found in [30]. Both decimation filter and interpolation filter can be categorized as resampling filters, which are essentially finite impulse response (FIR) filters with the capability of changing sampling rate of its input signal. FIR filter characteristics such as passband ripple, width of transition band, and stopband attenuation are determined by the filter order and coefficients. For a given sampling rate, a low passband ripple, sharp transition band, and large stopband attenuation are desirable features, which usually require a high filter order and take significant DSP resources in hardware. In addition, when supporting high capacity DSXC with fine spectral granularity of subcarrier channels the number of digital filters can be quite high. In summary, an efficient digital filter design is critically important in order to minimize the DSP resource requirement.

1.3 Digital subcarrier multiplexing techniques

DSXC is circuit cross-connect switch that is based on DSCM. Nyquist-FDM and OFDM are the are two major techniques for implementing DSCM.

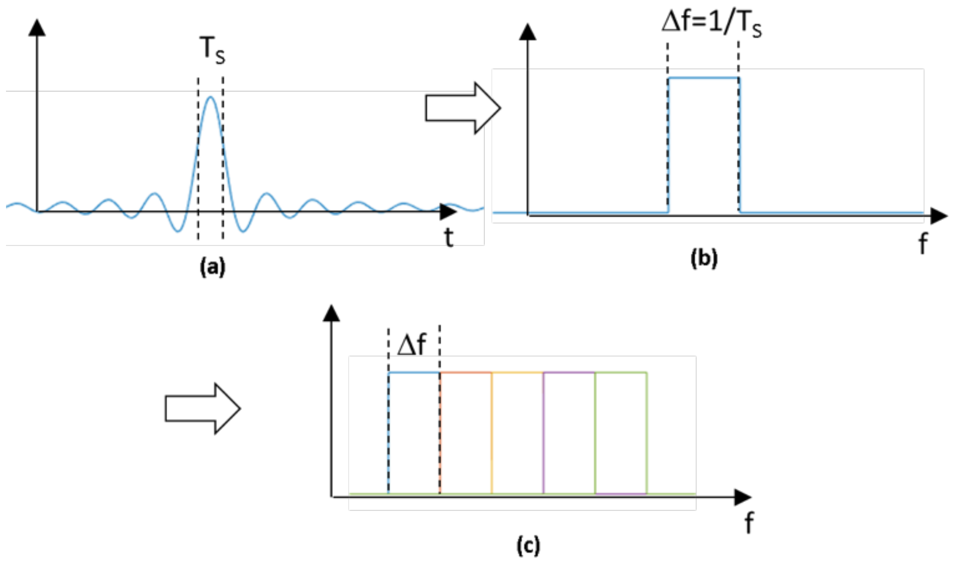


Figure 1.2 (a) Nyquist pulse in time domain (b) spectrum of a single Nyquist pulse (c) spectrum of Nyquist FDM

The basic principle of Nyquist pulse modulation is shown in Figure 1.2. For simplicity, we use OOK modulation format. Modulation formats with high spectral efficiency can also be employed [32]. As it shown in Figure 1.2 (a), a Nyquist pulse in time domain is a sinc-shaped waveform which spreads into adjacent time slots. By taking the Fourier transform of the Nyquist pulse in Figure 1.2 (a), we can get the spectrum of this pulse. As shown in Figure 1.2 (b), the spectrum of Nyquist pulse has a rectangular shape. By shifting the spectrum of Nyquist pulse in Figure 1.2 (b) by $k\Delta f$ ($k = 0, 1, \ldots, N-1$), we can get N sub-spectra of Nyquist pulse signal. The superposition of the N sub-spectra results in the total Nyquist-FDM spectrum as it illustrated in Figure 1.2 (c). Ideally, there is no guard-band between every two adjacent channels if perfect filtering is performed. However, ideal filtering cannot be realized in practical implementation since the order of digital filter is limited, so a guard-band between every two adjacent channels is required to prevent inter-channel crosstalk. In Nyquist-FDM based DSCM, phase synchronization between different channels is not needed, since every channel is independent from other channels.

The basic principle of OFDM is shown in Figure 1.3, suppose OOK is utilized as the modulation format. A single carrier OFDM signal in time domain is shown in Figure 1.3 (a), it has a rectangular pulse shape and a symbol period of T_S. By taking the Fourier transform of the signal in Figure 1.3 (a), we can get the spectrum of a single carrier OFDM signal whose bandwidth is Δf. The spectrum of this single carrier OFDM signal spreads to adjacent frequency slots. Shifting the spectrum of OFDM symbol in Figure 1.3 (b) by $k\Delta f$ (k = 0, 1,…, N-1), we can get N sub-spectra of OFDM signal. The superposition of the N sub-spectra results in the total OFDM spectrum as illustrated in Figure 1.3 (c).

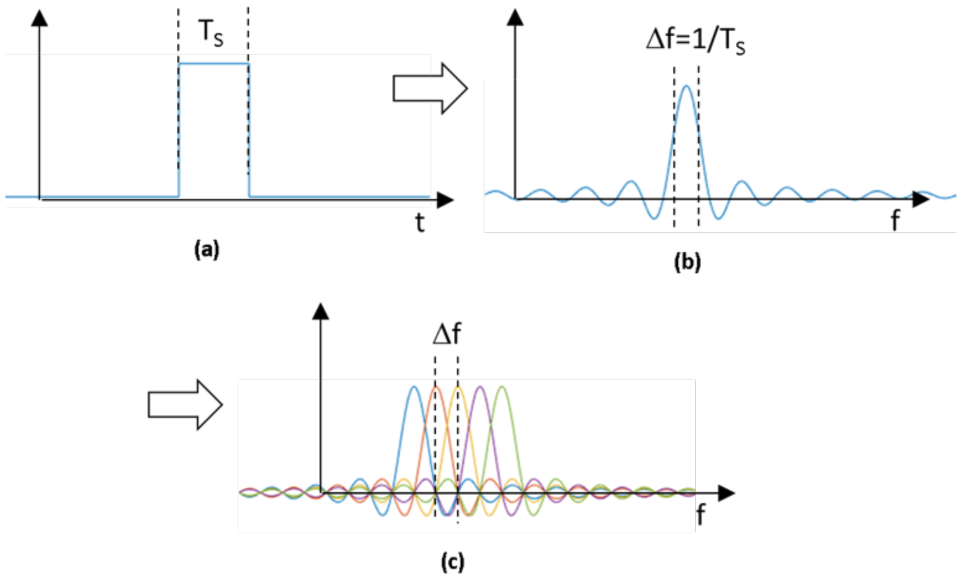


Figure 1.3 Principle of OFDM (a) single pulse shape (b) spectrum of a single OFDM symbol (c) spectrum of OFDM

As it shown in Figure 1.3, in OFDM based DSCM, the spectral of each channel spreads to adjacent channels and there is no guard-band between adjacent channels.

The research of real-time DSCM has been actively conducted. The real-time generation and reception of N-FDM have been reported and experimentally demonstrated [9, 14, 33]. Recent years, OFDM has also received a lot of attention in the field of optical communication [34]. The real-time generation of OFDM [9, 14, 35], real-time reception of OFDM [36-38] and the implementation of real-time OFDM transceivers [39, 40] have been demonstrated. Optical networks based on real-time OFDM with flexible power loading and bandwidth allocation have also been demonstrated [41-43].

1.4 Overview of proposed work

In this dissertation, we aim to investigate and implement real-time DSP-enabled DSXC for optical communication networks. In Chapter 2, we introduced the principle of DSXC and a hardware platform to implement real-time DSP for this DSXC. In Chapter 2 and Chapter 3, we investigate the cost and performance of different techniques, such as resampling filters and DA architecture, to efficiently implement this DSXC on the hardware platform. The implemented real-time DSP-enabled DSXC is demonstrated experimentally.

1.4.1 Real-time DSP hardware platform

We built a hardware platform to implement real-time DSP-enabled DSXC. This platform is based on Virtex7 FPGA, which allows the test of various real-time DSP algorithms for cross-connect switching in optical communication systems and networks. It consists of two ADCs boards, a DAC board, a Virtex 7 FPGA board and a data processing computer. In order to achieve very high data transfers between different parts of this platform, we developed the high-speed interface, such as JESD204B interface between FPGA and converters.

1.4.2 Channel selection and frequency translation in DSXC

In this work, we focus on the study of DSXC based on Nyquist FDM. The essential operation of a DSXC is the switching and routing of subcarrier channels, which is achieved through channel selection and frequency translation. There are multiple techniques to achieve channel selection and frequency translation. We discussed and compared the performance and resource cost of two techniques: (1) mixing and filtering, (2) resampling filters. In the design of real-time DSP algorithms, considerations must be given to the performance and resource cost because the DSP resource of FPGA is limited.

1.4.3 Implementation of DSXC based on hardware platform

After the development of real-time DSP algorithms and the hardware platform, we implemented the real-time DSP algorithm on this hardware platform. The implemented DSXC can be either based on resampling filters that consumes multipliers or resampling filters that based on DA architecture. In Chapter 3, we discuss and demonstrate the DSXC based on resampling filters that consume multipliers and compared it with traditional technique that is based on I/Q mixing and filtering. In Chapter 4, we describe the principle of DA architecture and introduce a DA-based DSXC, and we compare it with the DSXC based on resampling filters that utilize multipliers. We built the project for FPGA and generated the bitstream to be downloaded onto FPGA board, which results in a real-time DSP-enabled DSXC. We incorporated this cross-connect into optical fiber transmission system and analyzed its performance in experiment.

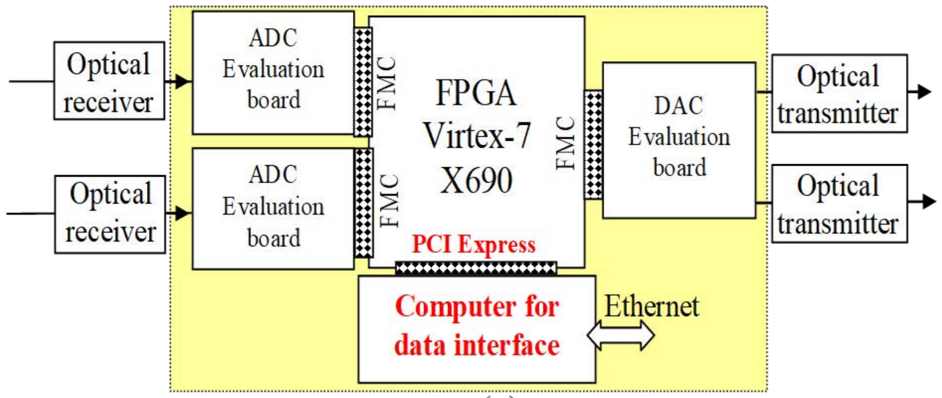
Chapter 2: Real-time DSP hardware platform

In this chapter, we introduce the architecture of a real-time DSP hardware platform for implementing DSXC. This platform consists of high-speed ADCs, high-speed DACs, FPGA, and optical transceivers. The development of data interfaces and considerations for FPGA implementation are also discussed in this chapter.

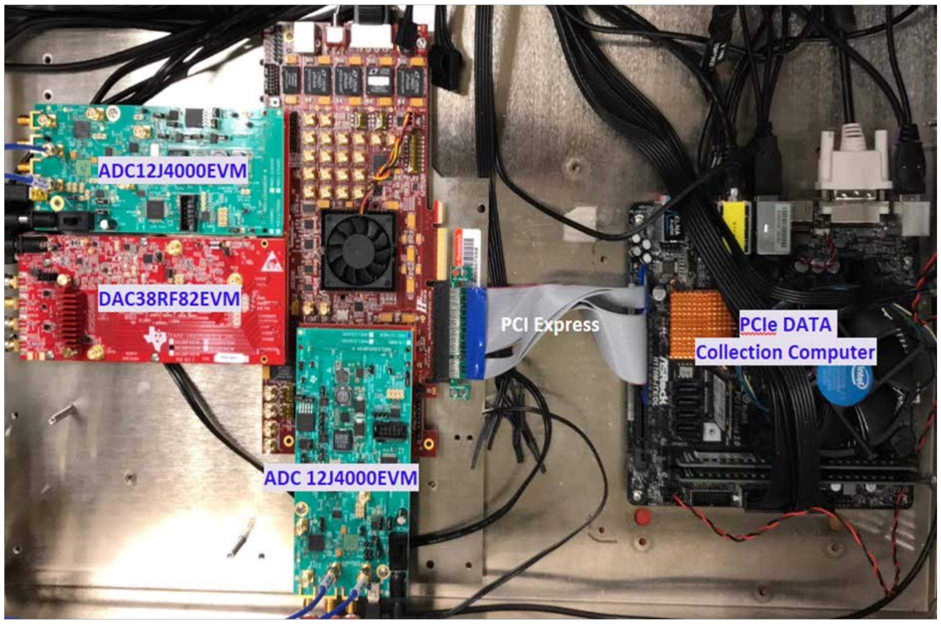
2.1 Hardware Platform

In order to meet the requirements of processing time in a practical application, a real-time DSP is needed. A general-purpose computer lacks the needed resources and is not suitable for real-time DSP. In contrast, an FPGA has the advantage of fast, parallel processing and is programmable, which makes it a good choice for building a platform for the prototype research of a real-time DSP for optical communication.

Our purpose is to establish an optical system testbed capable of generating, detecting, and processing advanced multiplexing techniques such as N-FDM and OFDM, which allows us to investigate various modulation formats and DSP algorithms in real-time optical systems and networks. In order to demonstrate algorithms and capabilities of DSCM, we have developed a flexible FPGA platform that consists of three major parts: an FPGA board (HTG700), two ADC boards (ADC12J4000EVM), and a DAC board (DAC38RF82EVM). The interface between the converters and the FPGA is a JESD204B, and the interface between the FPGA and the computer is a peripheral component interconnect express (PCIe).



(a)



(b)

Figure 2.1 (a) block diagram and (b) photo of DSCM testbed

As shown in Figure 2.1 (a), the testbed consists of two ADC evaluation boards, a DAC evaluation board, an FPGA board, and a connection with the data collection computer through the PCIe. Two ADCs are used to convert the received electrical signal from the optical receivers into the digital domain and send it to the FPGA board through the FPGA mezzanine card (FMC) connectors. The FPGA has three major tasks: digital down-conversion (DDC)/digital up-conversion (DUC), digital filtering, and cross-connect switching of digital subcarrier channels. Digital compensation of transmission impairments and waveform distortion can also be implemented with DSP algorithms in the FPGA platform. The DAC is used to convert the processed digital signals in the FPGA back to analog waveforms. Figure 2.1 (b) shows the cross-connect platform. This platform uses a Texas Instruments (TI) DAC evaluation board (DAC38RF82EVM), which supports two DAC output channels each with a 2.5GS/s un-interpolated input sampling rate with 16-bit resolution, and two ADC evaluation boards (TI: ADC12J400EVM), each with a 4GS/s sampling rate with 12-bit resolution. The FPGA evaluation board is a HTG700, which is mounted with a Xilinx Virtex7-XC7VX690T FPGA chip and equipped with three FMC connectors. The FPGA board also supports a PCIe-X8-Gen3 with eight lanes operating in parallel, and the maximum data rate is approximately 63 Gb/s.`

Table 2.1 Available FPGA resources on Virtex 5 and Virtex 7 [44, 45]

Part Number	**Slices**	**Logic Cells**	**CLB Flip-Flops**	**BRAM (Kbits)**	**DSP Slices**
XC5VFX200T	30,720	196,608	122,880	16,416	384
XC7VX690T	108,300	693,120	866,400	52,920	3,600

Table 2.1 shows the available FPGA resources on a Virtex 5 XC5VFX200T and a Virtex 7 XC7VX690T. A Virtex 7 has many more resources than a Virtex 5. The DSP slices on a Virtex 7 are almost 10 times those of a Virtex 5. With more FPGA resources, a DSP platform with higher performance can be developed and more complex DSP algorithms can be investigated.

Table 2.2 Specifications of ADC12J4000 and DAC38RF82

Part Number	Bit Resolution	Maximum Sampling Rate	Number of Channels	Interface
ADC12J400	12	4 GSPS	1	JESD204B
DAC38RF82	16	3.33 GSPS	2	JESD204B

Table 2.2 shows the specifications of the ADC and DAC used in the design of this DSXC. The ADC operates in single channel mode with 12-bit resolution, and the DAC operates in dual-channel mode with 16-bit resolution. Both of them have a JESD204B interface. Considering the available resources and timing issues in the FPGA design, we used 1.6 GHz as the sampling rate of both the ADC and DAC.

Table 2.3 Specifications of Optical Tx and Rx

Part Number	Bandwidth	Wavelength	Type	Modulation
OZ510	30 MHz ~ 3GHz	1330nm	Linear	Intensity modulation

Table 2.3 shows the specifications of the optical Tx and Rx used in this testbed. Both are linear, have intensity modulation, and have a RF bandwidth from 30 MHz to 3 GHz.

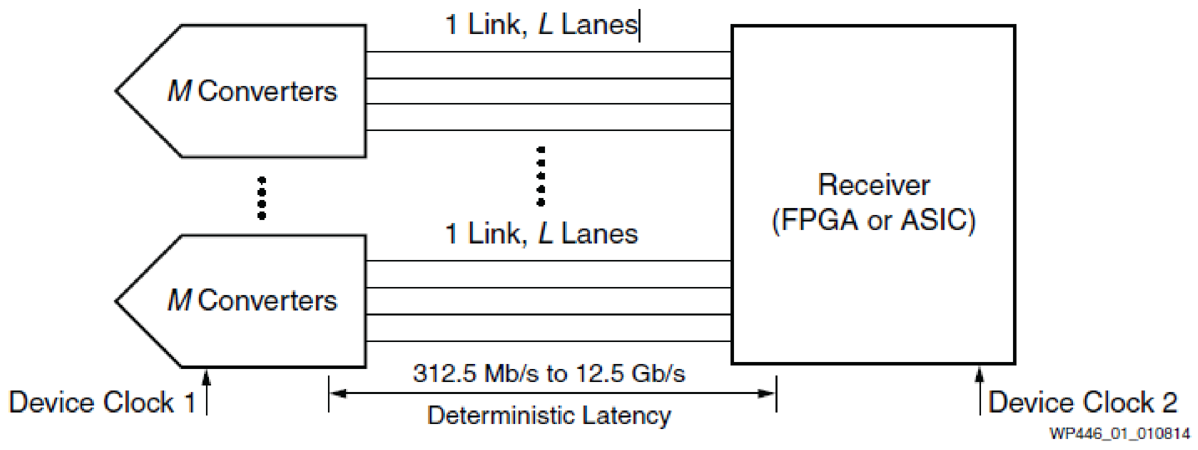
2.2 Data Interfaces

There are a few different data interfaces in our design. Low speed data interfaces include a JTAG (named after the Joint Test Action Group) interface and a universal asynchronous receiver-transmitter (UART) interface. The JTAG is the interface between the FPGA and the PC, through which the bitstream can be downloaded to the FPGA. The UART is also a low speed interface between the FPGA and the PC, which can be used to transfer low rate data between the FPGA board and the computer, such as during debugging and monitoring the status of modules in the

FPGA. The high-speed interfaces include the JESD204B and the PCI Express. The JESD204B is a high-speed serial interface between the data converters and the computer, and the PCI Express is a high-speed serial interface between the FPGA board and the computer. The JESD204B and PCI Express are complicated and are discussed in the following sections.

2.2.1 JESD204B interface

The high throughput rates of these gigabit ADCs and DACs push the limits and timing constraints of the current standard high-speed interface, a serial low-voltage differential signaling (LVDS) interface. In order to address this problem, the Joint Electron Device Engineering Council (JEDEC) committee created a robust wide data converter interface known as the JESD204 interface [46][48]. The latest version of the JESD204 interface is the JESD204B.



[Xilinx WP 446]

Figure 2.2 JESD204B standard

As shown in Figure 2.2, the JESD204B provides deterministic latency with a data rate in each serial lane up to 12.5 Gb/s[46, 47]. Since the data rate of each serial lane is much higher than in the LVDS, the required pin count in the JESD204B can be greatly reduced, which makes the design of a printed circuit board much simpler and more compact. With improvement comes complexity; the JESD204B interface is much more complex than the LVDS interface, which makes designing with the JESD204B more challenging. The JESD204B has an OSI-type protocol stack, which is very complicated. In order to reduce the complexity of design, Xilinx has provided a configurable FPGA object for implementing the JESD interface, called the JESD204B IP Core. In this design, a JESD204B subclass 1 is used.

A real-time high-speed transmitter is based on the FPGA board and DAC board. This design is able to generate arbitrary waveforms at the output of the DAC.

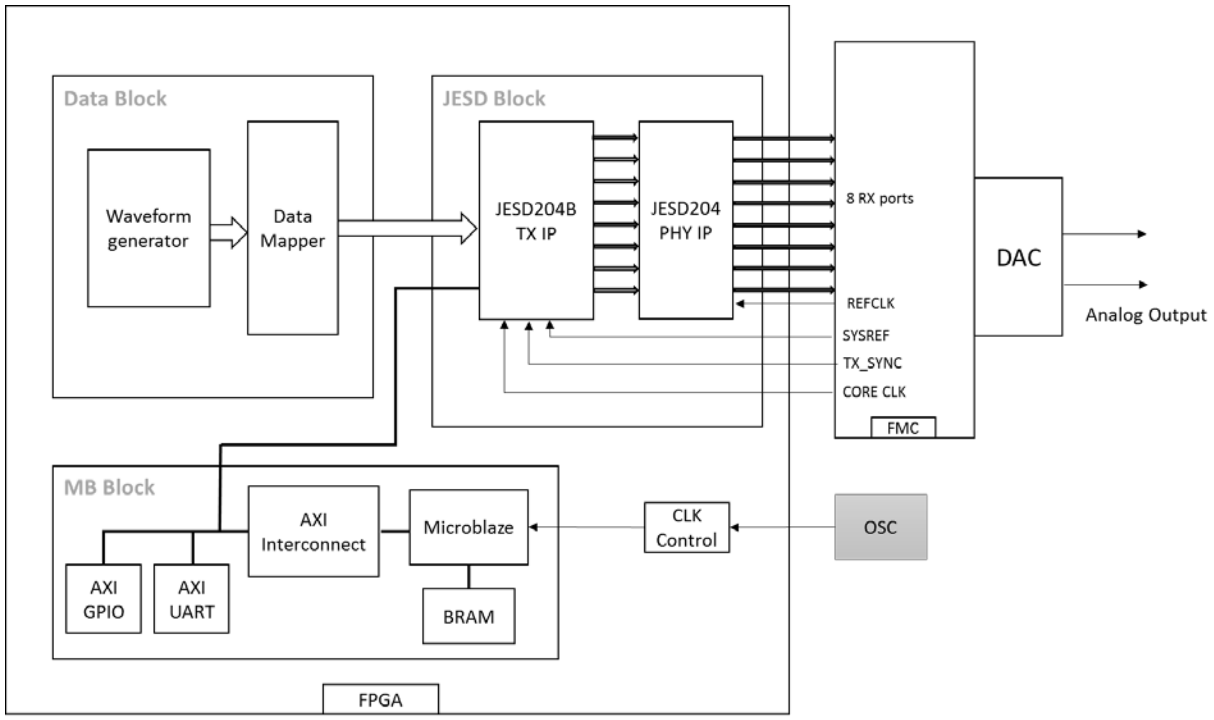


Figure 2.3 Block diagram of a real-time transmitter based on FPGA and DAC

Since the data is transferred from the FPGA to the DAC through the JESD interface, the FPGA board serves as a JESD transmitter, and the DAC board serves as a JESD receiver accordingly. Digital signals are transferred from the FPGA board to the DAC board through eight JESD serial lanes, and the maximum data rate of each JESD lane is 12.5 Gb/s. As shown in Figure 2.3, the block design inside the FPGA mainly consists of three blocks: the data block, the JESD block, and the Microblaze block (MB block).

In the MB block, the Microblaze is a soft-core processor implemented on the FPGA, which can be programmed by C language. With the Microblaze, IP cores can be easily programmed and configured through the advanced extensible interface (AXI). The block random access memory (BRAM) provides local memory for the Microblaze processor. The AXI interconnect serves as the bridge between the Microblaze and other AXI IPs. In the data block, the waveform generator is a user-packaged IP that generates waveforms by using a look-up table (LUT). The data mapper maps the generated waveform into the AXI data stream that fits the data format of the JESD block. In the JESD block, the JESD204B TX block transmits data to the JESD204 PHY once it receives the request from the DAC through the FMC connector. In the JESD block, the JESD TX IP is configured and monitored through an AXI4-Lite management interface. The JESD PHY IP implements the Xilinx GTX transceiver logic and control interface. After receiving the AXI stream data from the data block, the JESD TX IP sends the data to the JESD PHY IP. The source clock for the Microblaze and AXI interfaces is provided by an oscillator (OSC) on the HTG700. The JESD clock, which includes the reference clock (REFCLK) and core clock (CORE CLK) for JESD, are generated by the clock chip (LMK04828) on the DAC board.

A real-time high-speed receiver is based on the FPGA board and ADC board. This allows the ADC to sample and digitize analog signals and send digital signals to the FPGA for real-time processing. The receiver project is intended to build the interface and synchronize between the ADC board and the FPGA board. This is an essential part for a high-speed digital receiver, which enables the DSXC to convert analog signals received from the photodetector to the digital domain and transfer them to the FPGA board for real-time processing.

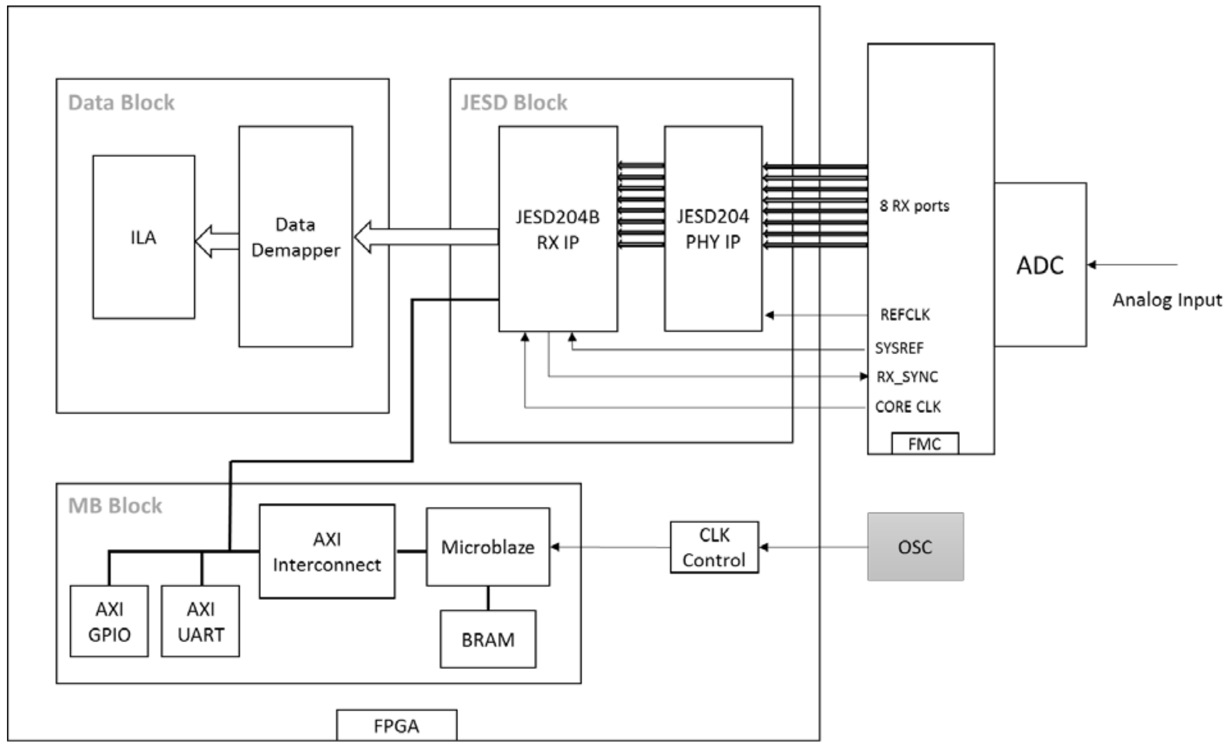


Figure 2.4 Block diagram of high-speed receiver based on FPGA and ADC

As shown in Figure 2.4, the block design of the real-time receiver consists of an MB block, data block, and JESD block. The design of the receiver is similar to the design of the real-time transmitter, except that in this design the FPGA board serves as the JESD receiver while the ADC board serves as the JESD transmitter. The ADC converts analog input signals into digital data and transmits them to the FPGA board.

As shown in Figure 2.3 and 2.4, projects for interfacing ADC and DAC with FPGA have been built in Xilinx Vivado separately. The ADC and DAC have been tested separately to verify if the JESD204B interfaces are working properly.

2.2.2 Design in Xilinx Vivado

In the FPGA design of the DSXC, Xilinx Vivado 2015.4 is used as the design tool. The design of the resampling filters is accomplished using the Xilinx System Generator and Filter Design HDL Coder. A more detailed description of the DSXC design process is given in Appendix I.

2.3 Considerations in FPGA Implementation

In implementing an FPGA, there are a few issues that need to be addressed, such as clocking and bit resolution, as they may have a significant impact on the signal quality. Timing is critical during the setup of a real-time DSP hardware platform. In this setup, the ADC and DAC use external clocks generated by signal generators. If the signal generators are running freely, the frequencies of their generated clocks may slightly differ, and this can cause a frequency offset in the digital signal. This frequency offset may cause the degradation of the signal quality at the output of the DAC. In order to avoid frequency offset when transferring a digital signal from the ADC clock domain to the DAC clock domain, it is necessary to lock the clock frequencies. We used the 10 MHz reference signal to lock the clock frequencies of all the signal generators and first in first out (FIFO) IP to transfer large amounts of data between the different clock domains.

Bit growth is also a very common issue in real-time DSP implementation, because in FPGA arithmetic, the bit width of a digital signal can rapidly grow, making it impossible to keep full precision during the processing. For example, adding two N bit numbers results in a $N + 1$ bit number, and multiplying two N bit numbers results in a $2N - 1$ bit number. If full precision is kept, the bit growth can quickly exceed the computation capability of the FPGA. Thus, it is very important to reduce bit width. The simplest way of reducing bit width is truncation, which can be achieved through dropping the least significant bits (LSB). However, truncation results in undesirable DC bias at the output. The round to even technique can solve the problem of DC bias by rounding up to the nearest even number. Compared with truncation, which does not consume logic resources, this technique costs logic resources but only contributes a negligible resource cost of the DSXC. In the implementation of the DSXC, we use the round to even technique to reduce the bit width while keeping sufficient resolution in the signal quality.

Another issue is that the FPGA has limited maximum clock frequency, on the order of a few hundred MHz. Current GSPS ADCs and DACs can achieve a sampling rate on the order of a few GHz, which is much higher than the maximum clock frequency in FPGA processing. The solutions for this issue are parallel processing and pipelining. Parallel processing means processing the data in parallel channels, so data with high sampling rates can be processed with a relatively low clock frequency. The FPGA architecture makes it very suitable for parallel processing. For example, if the FPGA has a clock frequency of 200 MHz and the data are processed in 20 parallel channels, then logically the total sampling rate is 4 GHz.

Timing is a critical issue in FPGA implementation, because it takes time for a signal to propagate from one flip-flop, through a combinational logic, to the next flip-flop. The more complicated the combinational logic, the longer takes for the signal to propagate. Timing can be a very difficult issue when the combinational logic is large and the FPGA clock frequency is high (above 50 MHz). The simplest technique to fix a timing issue is pipelining. By adding flip-flops into a large combinational logic, pipelining breaks the combinational logic into multiple stages, where the propagation delay in each stage is shorter than the original propagation delay. It is also important to note that pipelining does not decrease the total throughput of this digital design. Pipelining only increases the latency by a few clock periods, which is acceptable in many FPGA designs.

DSP units such as finite impulse response (FIR) filters, mixers, and local oscillators can also be easily implemented in parallel and be pipelined. Fast Fourier transform (FFT) can be implemented in FPGA by using a third-party IP named SpiralFFT, which can be generated in the format of a hardware description language (HDL) as described in [49]. Thus, all the DSP algorithms described in this thesis can be implemented in FPGA to achieve a very high processing rate.

Chapter 3: Real-time DSP-enabled DSXC based on resampling filters

In this chapter, we introduce the design and experimental demonstration of the first real-time DSP-enabled DSXC based on resampling filters. This DSXC is implemented on a Xilinx Virtex-7 FPGA board, and the results have been presented in [30].

3.1 Digital subcarrier cross-connect based on FPGA

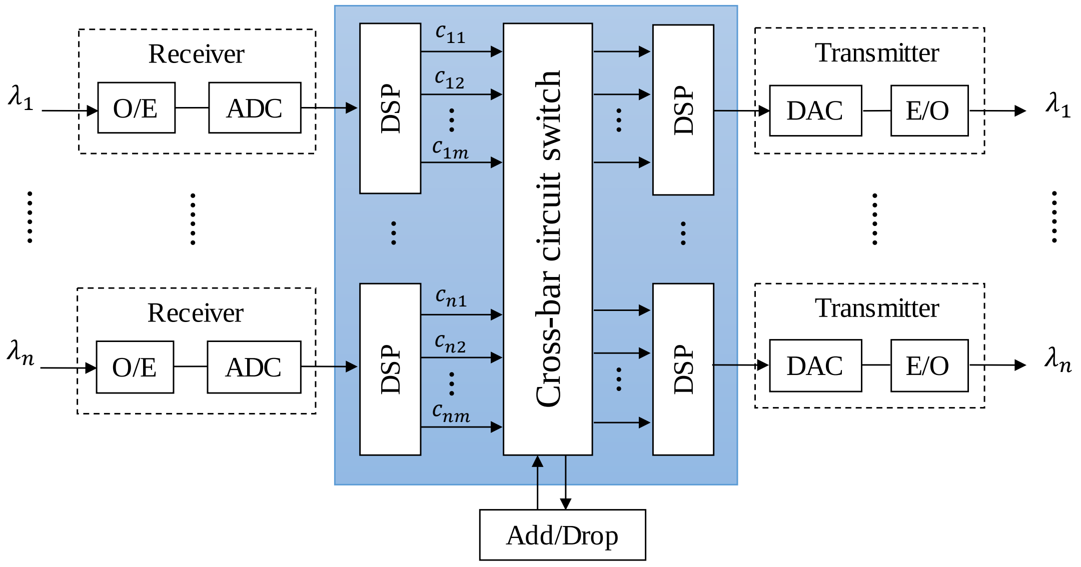


Figure 3.1 Block diagram of digital subcarrier cross-connect (DSXC) based on FPGA

Figure 3.1 shows a basic DSXC block diagram based on FPGA. The input signal to the DSXC switch fabric comprises *n* wavelength channels, and each wavelength carries *m* radio frequency (RF) subcarrier channels. The RF subcarriers can support different modulation formats and make use of different spectral bandwidths. Each wavelength channel is detected by an optical receiver performing optical to electrical (O/E) conversion, which is followed by an ADC digitizing the analog waveform delivered by the optical carrier. Then the digitized waveform is sent to a DSP block for subcarrier de-multiplexing. The de-multiplexed subcarriers from all wavelength channels are sent to a cross-bar circuit switch, in which any input subcarrier can be routed to any output port. Subcarrier channel local add/drop can also be performed in this cross-bar switch unit. The subcarriers switched through the cross-connect are then re-grouped and multiplexed by *n* DSP units corresponding to *n* output wavelength channels. Each digitally multiplexed composite signal forms a wavelength channel that is converted from electrical to optical (E/O) through a DAC and an optical transmitter. With this basic DSXC architecture, any subcarrier channel of any input optical carrier can be routed to any subcarrier of any output wavelength channel.

For demonstration purpose, the DSP units and the cross-bar circuit switch can be implemented using a single FPGA module which provides real-time processing. The cost of this implementation choice is proportional to the amount of FPGA resources that is required to implement the DSXC. FPGA resources mainly consist of memory resources and DSP slices. Memory resources include look-up table (LUT), LUT random access memory (LUTRAM), flip-flop (FF) and block RAM (BRAM). DSP slices are used to carry out digital multiplications, which are usually the most expensive operation in a real-time DSP hardware platform. For convenience, in this chapter we use the term *DSP cost* to represent the number of required DSP slices.

Since subcarrier channels in each wavelength are multiplexed in frequency domain, frequency translation (also known as spectral translation) is a critical operation in DSXC. A straightforward and conventional frequency translation technique is based on signal mixing and filtering operations whereby multiple local oscillators (LOs), mixers, and low pass filters are combined to achieve the intended goal. Digital filtering, which is the convolution between the input data sequence and the filter coefficients, is achieved through FIR filters that involve a large number of multiplications and represent the major DSP cost. Because the number of FIR filters increases linearly with the number of subcarriers in the cross-connect switch, the DSP cost for a DSXC based on this frequency translation technique also increases accordingly, and may become a major limiting factor.

Since DSXC is performed in the digital domain, some inherent properties of digital sequence and DSP algorithms can be utilized to achieve frequency translation at a reduced DSP cost. More specifically, interpolation and decimation are techniques that have been widely used in digital systems to change the sampling rate of a signal [50]. By applying some modifications as described in Section 3.2, interpolation and decimation techniques can be used to perform frequency translation with significantly less DSP cost in comparison to the frequency translation obtained through conventional I/Q/ mixing and filtering. For reader's convenience, we first briefly describe frequency translation through I/Q mixing and filtering. Then, we describe frequency translation through resampling filters.

3.2 Frequency translation techniques

In this section, we briefly introduce two techniques for frequency translation: 1) I/Q mixing and filtering; 2) Resampling filters.

3.2.1 I/Q mixing and filtering

A traditional technique for frequency translation is through I/Q mixing and filtering. As shown in Figure 3.2, traditional frequency translation includes down-conversion and up-conversion, in which LOs are implemented from direct digital synthesizer (DDS). In order to maintain phase synchronization between LO and the RF subcarrier whose frequency needs to be translated, I/Q mixing is usually required. Figure 3.2 shows a standard two-step digital frequency translation process which consists of both digital down-conversion (DDC) and digital up-conversion (DUC).

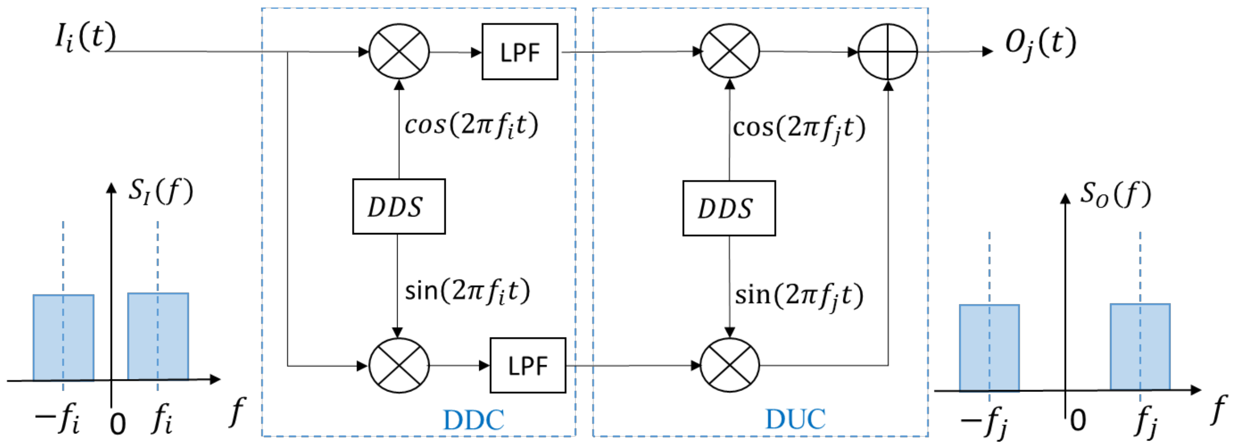


Figure 3.2 Frequency translation through I/Q mixing and filtering

As shown in Figure 3.2, in the DDC process, a DDS simultaneously generating sine and cosine waveforms is used to provide a pair of LOs. Assume the incoming signal data sequence on the i^{th} subcarrier channel is $I_i(t) = A_i(t)\cos(2\pi f_i t + \varphi_i)$, where $A_i(t)$ is modulated amplitude, f_i is the carrier frequency, and φ_i is the carrier phase; and the in-phase (*I*) and quadrature (*Q*) components of the LO are $\cos(2\pi f_i t)$ and $\sin(2\pi f_i t)$, respectively. After down-conversion mixing and low pass filtering (LPF), the *I* and *Q* components of the baseband signal are $\frac{1}{2}A_i(t)\cos(\varphi_i)$ and $-\frac{1}{2}A_i(t)\sin(\varphi_i)$, respectively. If the subcarrier channel needs to be dropped at this node, the *I* and *Q* components are combined together to recover the original baseband signal. Otherwise, the *I* and *Q* components are mixed with another pair of LOs, $\cos\left(2\pi f_j t\right)$ and $\sin\left(2\pi f_j t\right)$, in the DUC module. The frequency up-conversion generates $\frac{1}{2}A_i(t)\cos(\varphi_i)\cos\left(2\pi f_j t\right)$ and $-\frac{1}{2}A_i(t)\sin(\varphi_i)\sin\left(2\pi f_j t\right)$, and they are combined to form the DUC module output as,

$$O_m(t) = \frac{1}{2}A_i(t)\cos[2\pi f_j t + \varphi_i] \tag{3.1}$$

Throughout this frequency translation process, the carrier frequency is changed from f_i to f_j, while the original carrier phase, φ_i, is automatically maintained.

Note that I/Q mixing requires two separate filters for the I and the Q channels. Alternatively, one can use a single stage frequency translation from f_i to f_j using an LO frequency $|f_j - f_i|$. However, to avoid spectral overlap with other subcarrier channels, this approach requires the selection of the subcarrier channel at f_i by a bandpass filter before mixing, and the selection of subcarrier frequency at f_j by another bandpass filter after mixing, and thus the number of digital filters remains unchanged.

Alternatively, frequency translation may be applied to complex field modulated subcarriers in which the upper and lower sidebands of each subcarrier channel are not redundant. In that case Hilbert transform must be applied to avoid spectral aliasing, which would further increase the DSP cost.

As previously noted, when implementing this frequency translation in FPGA platform, the major DSP cost comes from digital filters, which increases linearly with the number of subcarrier channels.

3.2.1 Resampling Filters

Alternatively, frequency translation may be achieved through resampling and filtering of each subcarrier channel. Similar to mixing and filtering, the resampling and filtering technique also consists of both DDC and DUC processes. As shown in Figure 3.3(a) and 3.3(c), DDC is usually achieved by cascading a bandpass filter (BPF) with a down sampling unit, while DUC can be achieved with an up-sampling unit followed by a BPF. For simplicity, we refer both down-sampling and up-sampling as *resampling* [50].

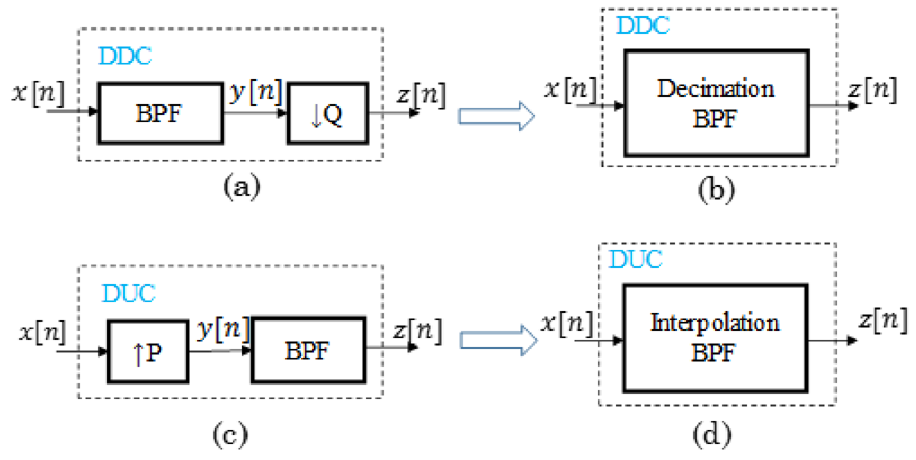


Figure 3.3 (a) DDC through BPF and down-sampling, (b) DDC through interpolation BPF, (c) DUC through up-sampling and BPF, (d) DUC through decimation BPF

In Figure 3.3(a), a BPF is used to select a specific subcarrier channel, the data sequence after BPF is down sampled by a factor of Q through a down sampling unit.

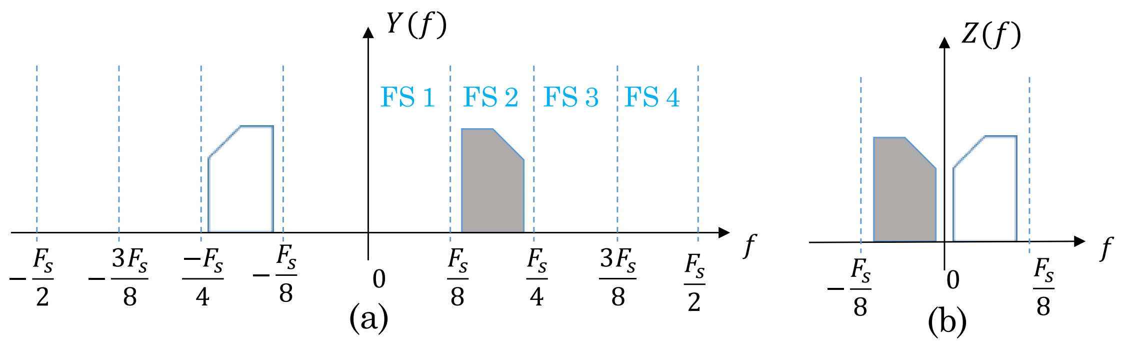


Figure 3.4. (a) DFT of $y[n]$, and (b) DFT of $z[n]$ for down sampling

Figure 3.4 shows an example of discrete Fourier transform (DFT) spectra of the input data sequence $y[n]$ and the output data sequence $z[n]$ of the down sampling unit with an input sampling frequency F_s and a down-sampling factor $Q = 4$. Through down sampling, the frequency range is scaled down by a factor of 4 from $(-F_s/2, F_s/2)$ to $(-F_s/8, F_s/8)$. As shown in Figure 3.4(a), the selected subcarrier channel originally located in frequency slot 2 (FS2) is automatically down shifted to the frequency slot $(0, F_s/8)$. In general, if the selected subcarrier channel is originally located within an even frequency slot, such as FS2 and FS4 in Figure 3.4(a), it will be spectrally flipped after down sampling. While if it is located in an odd frequency slot, such as FS1 and FS3, its spectrum will not flip.

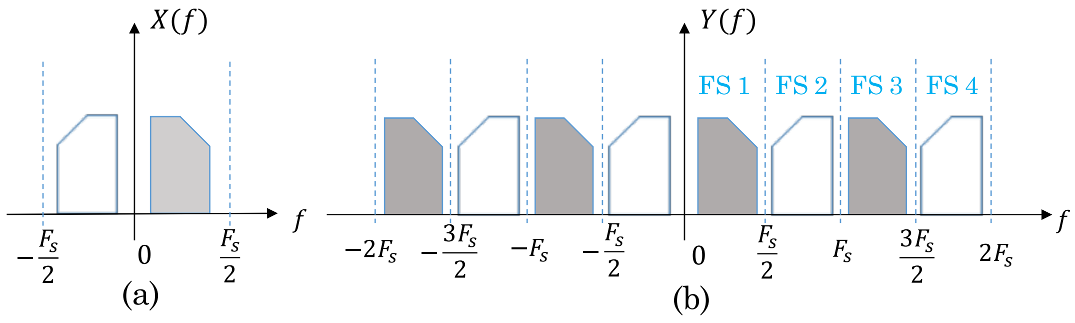


Figure 3.5. (a) DFT of $[n]$, and (b) DFT of $y[n]$ for up-sampling

For the process of DUC shown in Figure 3.3(c), a subcarrier channel at the lowest frequency slot needs to be translated to a higher frequency slot. Figure 3.5 shows an example of DFT spectra of the input data sequence $x[n]$ and the output data sequence $y[n]$ of the up-sampling unit with a sampling rate F_s and an up-sampling factor of $P = 4$. Through up-sampling, the frequency range is expanded 4 times from $(-F_s/2, F_s/2)$ to $(-2F_s, 2F_s)$. The up-sampled DFT spectrum in this expanded frequency range consists of multiple copies of the original spectrum, and each of them falls into a different frequency slot. By applying a band pass filter on the up-sampled spectrum, a particular copy of spectrum at the desired frequency slot can be selected, which is equivalent to a frequency translation. Again, similar to the down-sampling process the frequency-translated spectra in even frequency slots such as FS2 and FS4 shown in Figure 3.5(b), are flipped in comparison to the original spectrum in FS1. The flipped spectrum, although contains the full information, is a frequency conjugated version of the original signal, and thus another conjugate operation has to be performed when the baseband waveform needs to be recovered.

In comparison to I/Q mixing and filtering, the resampling and filtering technique shown in Figure 3.3(a) and 3.3(c) does not need LOs and mixers, and there is no need for carrier phase synchronization. Since the actual bandwidth of each frequency slot is determined by the sampling frequency and the resampling factor, it can be flexible to accommodate different data rates carried by different subcarriers. Suppose the sampling frequency is f_s and the resampling factors are L_1, L_2 and L_3, then the bandwidths of the frequency slot after resampling is $f_s/(2L_1)$, $f_s/(2L_2)$ and $f_s/(2L_3)$, respectively. The channel data rate granularity of DSXC can be made fine enough to address network efficiency requirements through the change of resampling factor.

However, the major drawback of both DDC and DUC shown in Figure 3.3(a) and 3.3(c) is that the BPF still requires significant DSP resources of FPGA, similar to that based on I/Q mixing and filtering. A novel technique to solve this problem is to combine the resampling and BPF into a single resampling BPF as showing in Figure 3.3(b) and 3.3(d). Resampling BPF is a general term which includes decimation BPF for DDC and interpolation BPF for DUC.

Resampling filters can be implemented as polyphase decimation or interpolation filters on FPGA hardware [51], which was proposed primarily for resampling of data sequences while avoiding spectral aliasing and rejecting spectral images. Although polyphase resampling filters have been previously used in wireless transceivers [52], they have not been used for DSXC switches which require the capability of handling asynchronous subcarrier channels with non-equal bandwidth and independent modulation formats. While the required DSP resources linearly increases with the number of subcarrier channels for both I/Q-mixing-and-filtering and resampling-and-filtering, DSP resources required for resampling BPF is independent of the number of subcarrier channels. This significantly reduces the DSP resources requirement for FPGA implementation.

Digital frequency translation can be accomplished with resampling and filtering. As shown in Figure 3.3(a), DDC can be achieved by applying a band-pass filter (BPF) before a down-sampling unit, so that the subcarrier channel selected by the BPF is down-converted to a lower frequency slot. Let Q be the down-sampling factor indicating that one of every Q output samples from the BPF is retained, while the other $Q-1$ samples are discarded. The processing of these $Q-1$ discarded samples is in fact unnecessary and could be avoided to reduce the DSP cost. By combining down-sampling and filtering into a single decimation filter as shown in Figure 3.3(b), only one of every Q sampling operations is actually performed for BPF. Therefore, the total computation is effectively reduced by a factor of Q, and the DSP cost of a decimation BPF is only $1/Q$ of a conventional BPF with the same number of coefficients.

Similarly, DUC can be achieved by using an up-sampling unit followed by a BPF, as shown in Figure 3.3(c), so that a subcarrier channel at a lower frequency slot is up-converted to a higher frequency slot and be selected by the BPF. For an up-sampling factor of P, $P-1$ zeros are inserted between every two samples of the input digital sequence in the up-sampling process. As the multiplication of these inserted zeros with filter coefficients always results in zeros in the subsequent digital filtering process, these operations are not necessary. By combining up-sampling and filtering into a single interpolation filter as shown in Figure 3.3(d), unnecessary operations performed on the inserted zeros can be avoided by using only $1/P$ of the BPF coefficients during each convolution. Thus, the total computation is effectively reduced by a factor of P and the DSP cost of the interpolation BPF is only $1/P$ that of a traditional BPF with the same number of coefficients.

The frequency translation based on resampling filters and its DSP cost can be analyzed mathematically. In Figure 3(a) the input sequence $x[n]$ represents a digital multi-carrier signal.

Suppose that the BPF is an N-tap FIR filter with coefficients $h_0, h_1, \ldots, h_{N-1}$, then the filter output is

$$y[n] = \sum_{i=0}^{N-1} h_i \cdot x[n-i] \tag{3.2}$$

which is the digital sequence of a selected subcarrier channel that needs to be down-converted. After down sampling by a factor of Q, the output is

$$z[n] = y[nQ] \tag{3.3}$$

The Z-transform of $z[n]$ can be calculated as

$$\begin{aligned} \mathrm{Z(z)} &= \sum_n z[n] \cdot z^{-n} \\ &= \sum_n y[nQ] \cdot z^{-n} \\ &= \sum_m y[m] \cdot \left(\frac{1}{Q} \sum_{p=0}^{Q-1} e^{j\frac{2\pi}{Q}pm} \right) \cdot z^{-\frac{m}{Q}} \\ &= \frac{1}{Q} \sum_{p=0}^{Q-1} \sum_m y[m] \cdot (e^{-j\frac{2\pi}{Q}pm} \cdot z^{1/Q})^{-m} \\ &= \frac{1}{Q} \sum_{p=0}^{Q-1} Y(e^{-j\frac{2\pi}{Q}p} \cdot z^{1/Q}) \end{aligned} \tag{3.4}$$

By letting $\mathrm{z} = e^{j\Omega}$ in Equation (3.4), the discrete-time Fourier transform (DTFT) of the down-sampled digital sequence $z[n]$ can be calculated as

$$\mathrm{Z}\left(e^{j\Omega}\right) = \frac{1}{Q} \sum_{p=0}^{Q-1} Y(e^{j\frac{\Omega - 2\pi p}{Q}}) \tag{3.5}$$

where $\mathrm{Y}\left(e^{j\Omega}\right)$ is the DTFT of $y[n]$.

According to Equation (3.5), $\mathrm{Z}\left(e^{j\Omega}\right)$ is an expanded and shifted version of $\mathrm{Y}\left(e^{j\Omega}\right)$ with an expansion factor Q. Since the DTFT of a digital sequence is periodical with a period of 2π, every spectral component with an original bandwidth π/Q will be expanded to π. As a result, the original spectrum which occupies a frequency slot (FS) with a bandwidth of π/Q will be automatically expanded to π, and frequency shifted by $2p\pi$ into Q copies with $p = 0, 1, 2 \dots Q-1$.

Although $\mathrm{Y}\left(e^{j\Omega}\right)$ is continuous and periodical, the DFT of $y[n]$, is one sampled period of its DTFT, which is sampled at discrete points $\Omega = 2\pi k/M$, where Ω is a normalized angular frequency, M is the length of DFT, and $k = 0, 1, 2 \dots M-1$ is the index of sampling in the frequency domain. Thus, $\Omega \in (0, 2\pi)$ for $M >> 1$. If the sampling rate of $y[n]$ is F_s, the actual frequency range of its DFT, denoted by $Y(f)$, is $(-F_s/2, F_s/2)$.

As described previously, it is unnecessary to calculate the values of the samples in $y[n]$ that are not used in the subsequent down-sampling process. The BPF and the down-sampling unit could be more efficiently implemented together as a decimation filter as shown in Figure 3.3(b). Consider Equation (3.2) and Equation (3.3), the output of the decimation BPF is

$$z[n] = \sum_{i=0}^{N-1} h_i \cdot x[nQ - i] \tag{3.6}$$

Compared with Equation (3.2), the amount of calculations in Equation (3.6) has been reduced by a factor of Q. This is because the number of output samples of the decimation filter is Q times less than that of a traditional digital filter, and thus the DSP cost is reduced by a factor of Q.

Similarly, for the conventional up-sampling process shown in Figure 3.3(c), suppose the input digital sequence is $x[n]$, after up-sampling by a factor of P, the output is,

$$y[n] = \begin{cases} x[n/P] & \text{if } n/p \text{ is an integer} \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

The Z transform of $y[n]$ can be calculated as

$$\mathrm{Y(z)} = \sum_n y[n] \cdot z^{-n} \tag{3.8}$$

$$= \sum_{n:\frac{n}{p}\in Z} x\left[\frac{n}{P}\right] \cdot z^{-n}$$

$$= \sum_{k} x[k] \cdot z^{-kP}$$

$$= X(z^P)$$

Based on the result of Equation (3.8), the DTFT of the up-sampled digital sequence $y[n]$ can be calculated by letting $\mathrm{z} = e^{j\Omega}$, we get

$$Y\left(e^{j\Omega}\right) = X\left(e^{j\Omega \mathrm{P}}\right) \tag{3.9}$$

Where $X\left(e^{j\Omega \mathrm{P}}\right)$ is the DTFT of $x[\mathrm{n/P}]$. In this up-sampling process, $\mathrm{Y}\left(e^{j\Omega}\right)$ is a compressed version of $\mathrm{X}\left(e^{j\Omega}\right)$, and the compression factor is equal to the up-sampling factor P.

This up-sampling process can be explained by the similar scaling rule between DFT and DTFT as described above for down-sampling. An up-sampling by a factor P is equivalent to creating P equally spaced copies of DFT of $x[n]$, denoted by $Y(f)$, in the expanded frequency range of $(-\mathrm{P}F_s/2, \mathrm{P}F_s/2)$.

As shown in Figure 3.3(c), $y[n]$ and $z[n]$ are the input and output of the BPF, respectively. Suppose this BPF is a N-tap FIR filter with coefficients $h_0, h_1, \ldots, h_{N-1}$, the output of this BPF is

$$z[n] = \sum_{i=0}^{N-1} h_i \cdot y[n-i] \tag{3.10}$$

As described previously, the up-sampling unit and BPF can be combined into an interpolation BPF to reduce the DSP cost. Consider Equation (3.7) and Equation (3.10), the output of this combined interpolation BPF is

$$z[n] = \sum_{k=0}^{N/P-1} h_{n-kP} \cdot x[k] \tag{3.11}$$

Compared with Equation (3.10), Equation (3.11) only requires N/P, instead of N, multiplications, so that the DSP cost is reduced accordingly by a factor of P.

3.3 Resource utilization of frequency translation

In order to estimate the resource utilization of different frequency translation techniques, a 4×4 DSXC is designed in Xilinx System Generator. In this design, the FPGA platform is based on Xilinx Virtex-7 690t [45].

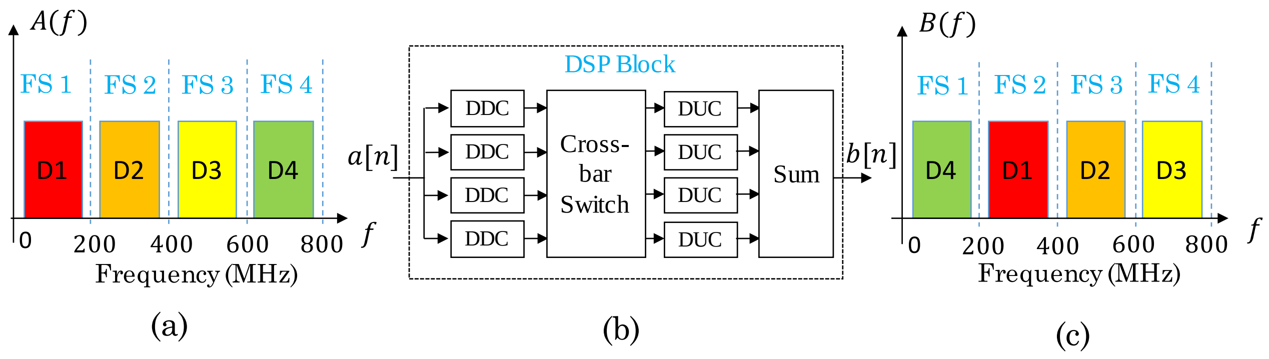


Figure 3.6 (a) spectrum of a[n], (b) DSP block of DSXC, and (c) spectrum of b[n]

Figure 3.6(a) shows an example of input electrical signal spectrum which has 4 subcarrier channels each carrying a different data sequence ($D1 \sim D4$). For simplicity, in this example each subcarrier channel has the same bandwidth. Figure 3.6(b) shows the block diagram of DSP used for this 4×4 DSXC. In this DSP block, the composite digital sequence including all 4 subcarrier channels at the input is first made into 4 equal copies. Each of the 4 DDC blocks down converts a channel from its subcarrier frequency to the baseband. The 4×4 cross-bar switch routes each down-converted baseband data sequence to a DUC block for frequency up-conversion. Channel add/drop is also possible at this stage before DUC. After up-conversion with each channel assigned a new subcarrier frequency, these subcarrier channels are combined at the output and sent to a DAC. The spectrum of the output electrical signal is illustrated in Figure 3.6(c) with the frequencies of subcarriers switched in comparison to the input spectrum.

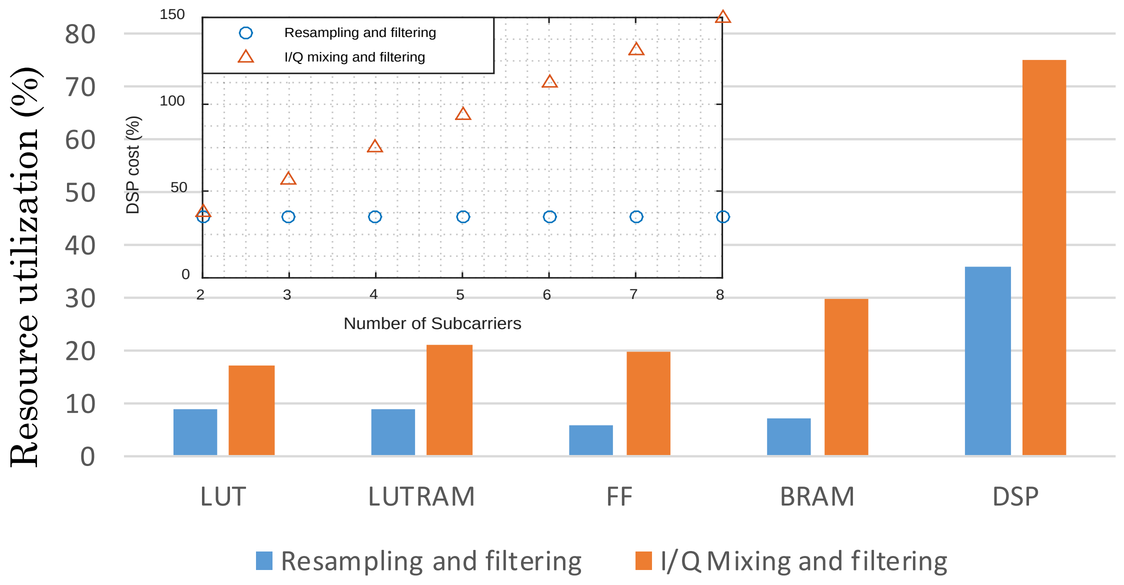


Figure 3.7 FPGA resource cost of a 4×4 DSXC

Two frequency translation techniques, one based on resampling BPF, and the other one based on I/Q mixing and filtering, are compared for this example. Both of them use 800MHz total analog bandwidth which is equally divided into 4 frequency slots with 200MHz bandwidth in each slot. 40MHz is reserved as the guard band between adjacent subcarrier channels. 79^{th} order finite impulse response (FIR) filters are used for both techniques with 60dB stopband attenuation. For the resampling-based frequency translation technique, the resampling factor is 4, and bandpass-filtering in each DDC block is accomplished by a decimation BPF. Similarly, each DUC block also performs bandpass-filtering which is implemented as an interpolation BPF. For the frequency translation based on mixing with LO, each DDC is performed by I/Q mixing and filtering by two low-pass FIR filters for the *I* and the *Q* channels, and each DUC also uses a DDS and two mixers, as shown in Figure 3.2. In this configuration, LOs are implemented through DDS by using look-up table (LUT), and mixers are implemented as digital multipliers, they both cost FPGA resources in memories and DSP slices. Figure 3.7 shows the comparison of FPGA resource cost to build this 4×4 DSXC based on the two different frequency translation techniques. In order to achieve the same performance, DSXC based on I/Q mixing and filtering has more than twice DSP cost than that based on resampling filters.

Here we used Xilinx Virtex-7 690t FPGA chip as the DSP hardware platform, and the total number of available DSP slices on this chip is 3600. As indicated by Figure 3.7, the bottleneck of the FPGA available resources in the design of DSXC is the DSP slices, and thus it is very important to minimize the cost of DSP slices in the design of DSP algorithms. The usage of DSP slices is mainly consumed by FIR filters, and the design of FIR filters is a tradeoff between the performance and resources cost. A higher order FIR filter has smaller passband ripple, sharper cutting edges, and higher stopband attenuation, but has higher resources cost. Passband ripple of a FIR filters causes frequency-dependent attenuation of the signal in the passband, which introduces signal waveform distortion.

For a traditional FIR filter to be implemented on a Virtex-7 FPGA, and suppose the length of its coefficients is S and the coefficients are symmetric, if the degree of parallelism is R, then the DSP cost of this FIR filter is $R \times (S/2)$. Since a mixer is just a multiplier that supports parallel processing, it simply uses R DSP slices. For the I/Q mixing and filtering technique shown in Figure 3.2, the frequency translation of each subcarrier channel needs 2 filters and 4 mixers, so that it requires $(S+4)R$ DSP slices. If the number of subcarrier channels is L, the total DSP cost of a DSXC based on I/Q mixing and filtering is

$$C_1 \approx (S+4)LR \tag{3.12}$$

Since usually $S >> 4$, the resources cost of this DSXC mainly comes from FIR filters. According to Equation (3.12), the total DSP cost increases linearly with the number of subcarriers.

Resampling BPF will cost less resource compared to a traditional FIR filter of the same coefficients. In order to compare with the frequency translation based on I/Q mixing and filtering with the number of subcarrier channels of L, we assume the total available bandwidth is B, and the bandwidths of subcarrier channels are $B_1, B_2, \dots, B_L$, so that $\sum_{i=1}^{L} B_i = B$. The resampling factor M_i of subcarrier channel i is inversely proportional to the bandwidth of that channel, $M_i = B/B_i$, thus an $L \times L$ DSXC could be built by using resampling filters with resampling factors $M_1, M_2, \dots, M_L$. For a resampling FIR filter with a resampling factor M_i and a length of coefficients S, if the degree of parallelism in the DSP system is R, the number of required DSP slices to build this resampling FIR filter is $R \times S/M_i$. Since a DDC block needs a decimation BPF and a DUC block needs an interpolation BPF for each subcarrier channel, all together the DSXC needs L decimation BPF and L interpolation BPF. Therefore, the total DSP cost for building these resampling filters is

$$C_2 \approx \sum_{i=1}^{L} 2 \times R \times S/M_i = 2S \times R \tag{3.13}$$

According to Equation (3.13), the total DSP cost of DSXC based on resampling filters for frequency translation is independent of the number of subcarrier channels. Basically, a higher channel count requires a larger resampling factor for resampling filters which reduces the DSP cost of each filter, and thus the total DSP cost does not increase with the number of channels. In comparison, for frequency translation based on I/Q mixing and filtering technique, each digital filter requires the same amount of DSP slices and thus the overall DSP cost increases linearly with the number of subcarrier channels.

The inset of Figure 3.7, obtained through Equation (3.12) and Equation (3.13), shows the DSP cost of DSXC based on two different methods. Here we assume length of filter coefficients is $S = 80$ and the degree of parallelism is $R = 4$. The DSP cost of the DSXC based on resampling filters remains unchanged when the number of subcarriers increases. Whereas the DSP cost increases linearly with the number of subcarriers for I/Q mixing and filtering, and there will not be enough DSP slices available on a Xilinx Virtex-7 690T FPGA if the number of subcarriers exceeds 5.

Although we used 4×4 DSXC as the example with equal subcarrier channel spacing and equal data rate for each channel, unequal channel spacing and different bandwidth for subcarrier channels can also be used because the resampling factor for each channel can be independently set. This has been experimentally demonstrated and will be discussed in the next section.

3.4 Experiments

In order to demonstrate DSXC and test its performance experimentally, an optical system based on digital subcarrier multiplexing has been setup using an FPGA platform for real-time DSP and cross-connect switching.

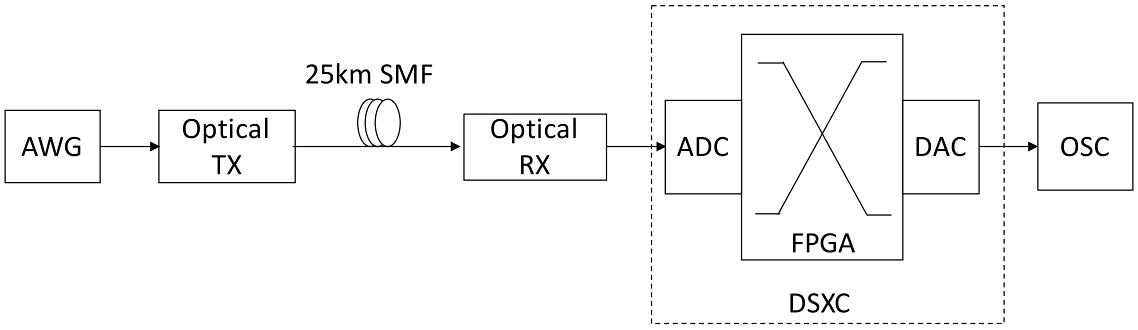


Figure 3.8 Experimental setup

The experimental setup is shown in Figure 3.8, where an AWG generates an electrical waveform which has multiple subcarriers. A linear optical transmitter converts this multicarrier electrical waveform into optical domain through direct intensity modulation. The optical signal is transmitted through 25 km standard single mode fiber (SMF), and detected by an optical receiver which linearly converts the received optical signal into electrical waveform. This detected electrical waveform is then sent into the DSXC for subcarrier level cross-connect switch. The waveform at the output of the DSXC is sampled by a real-time digital oscilloscope (OSC, DPO72304DX) for analysis.

This DSXC platform consists of three major parts: FPGA board (Hitech-global HTG700), analog-to-digital converter (ADC) (TI ADC12J4000EVM) and digital-to-analog converter (DAC) (TI DAC38RF82EVM). The resolutions of the ADC and the DAC are 12-bit and 16-bit respectively. Because a common clock is required, both the ADC and the DAC are running at an input sampling rate of 1.6 GSPS, so that the available analog bandwidth is 800MHz. The FPGA board is mounted with a Xilinx Virtex-7 690t FPGA chip. The FPGA clock frequency is 200MHz, and thus the sampled data in FPGA is processed in eight parallel channels.

Based on this DSP platform, an 8×8 DSXC has been implemented, which switches subcarrier channels with three different data rates. Only resampling filters have been used for frequency translation in the experiment. The available analog bandwidth of 800MHz is divided into 8 frequency slots with three different widths: 200MHz, 100MHz and 50MHz, and 20MHz is reserved for the guard band between adjacent subcarrier channels. Thus, the bandwidth of the corresponding subcarrier channels are 180MHz, 80MHz and 30MHz, respectively, and in principle each can have an independent modulation format. Equiripple 108th order FIR filters are used in this experiment for DDC and DUC. The ripple in the filter passband is 0.5dB and the stopband attenuation is 30dB.

Table 3.1 shows an example of the input signal to the 8×8 DSXC, in which 8 subcarrier channels, SC1, SC2, …SC8, are generated by the AWG. Each subcarrier is filtered to have rectangular spectral shape by using the method described in [53]. To demonstrate the capability of working with mixed modulation formats and data rates, Table I shows the bandwidth and modulation format assignment for the 8 subcarrier channels.

Table 3.1 AWG generated input signal to 8×8 DSXC

Subcarrier #	**Bandwidth (MHz)**	**Modulation Format**
1	30	QPSK
2	80	QPSK
3	180	16QAM
4	80	16QAM
5	80	QPSK
6	80	QPSK
7	30	QPSK
8	30	QPSK

Figure 3.9 (a) shows the spectrum of the signal at the output of the optical receiver, where each subcarrier channel has almost equal amplitude. In order to characterize the effects of this DSP platform imposed on the signal, we measured the output of DSP platform without cross-connect switching, and the spectrum is shown in Figure 3.9 (b). Although most channels from SC1 to SC6 have nearly the same amplitude at the output for the frequency range of <700MHz, high frequency channels SC7 and SC8 experience large role-off of more than 5dB for the frequencies beyond 700MHz. This roll-off is mainly caused by ADC and DAC.

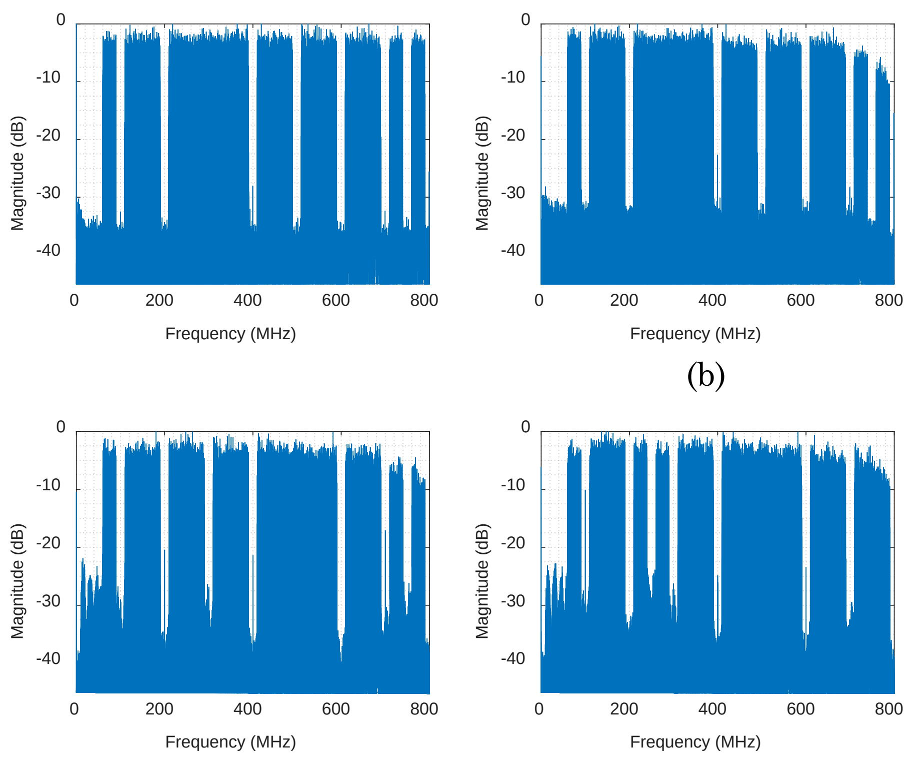


Figure 3.9 Spectra of output signal: (a) at optical receiver output which is DSXC input (b) after DSP platform but without switching (c) after DSP platform with switching assignment of DSXC1 (d) after DSP platform with switching assignment of DSXC2.

Figure 3.9(c) and (d) show the spectra after subcarrier switching for two different output channel assignments. In Figure 3.9 (c) denoted as DSXC1, the original subcarrier channels [1 2 3 4 5 6 7 8] have been switched to [7 4 6 5 3 2 8 1] at the output. While for the spectrum shown in Figure 3.9 (d) denoted as DSXC2, the original subcarrier channels [1 2 3 4 5 6 7 8] are switched to [8 6 1 7 2 3 4 5].

To evaluate the impact of DSXC on the signal quality, the waveforms at DSXC input and output are processed to find the error vector magnitude (EVM) for each subcarrier channel. The EVM is calculated according to the method described in [54]. As the frequency response of the DSP platform including ADC and DAC is deterministic as shown in Figure 3.9 (b, c, d), its impact can be digitally compensated in frequency domain at the transmitter and/or the receiver.

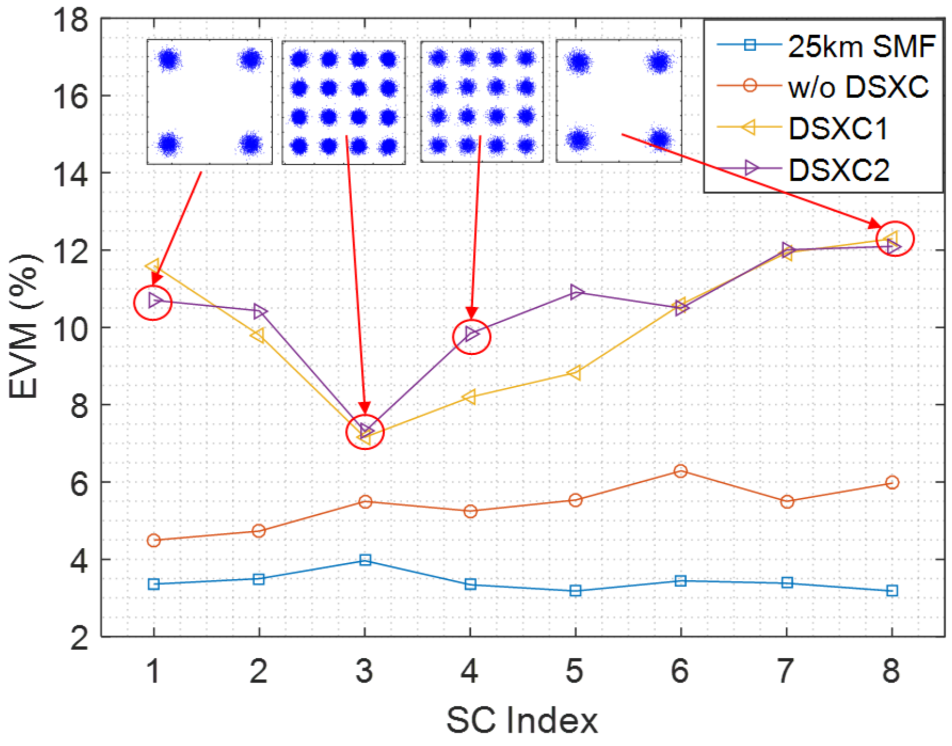


Figure 3.10. Signal EVM of recovered subcarrier channels

Figure 3.10 shows the EVM of the 8 subcarrier channels in 4 different scenarios. Open squares show the EVM measured at the input of DSXC, which is after 25km fiber transmission and detected by the optical receiver. Open circles show the EVM after passing through the DSXC platform but without cross-connect switching, and thus no digital filters are applied for each subcarrier channel. The EVM degradation compared to those shown by open squares is primarily due to the frequency-dependent transfer functions and high frequency roll-offs of ADC and DAC. Although we have applied slope compensation at the receiver, small amount of EVM degradation still exists, especially for high frequency channels. Open triangles in Figure 3.10 show EVM values of all channels after cross-connect switching with two different output channel assignments corresponding to the spectra shown in Figure 3.9(c) and (d). The additional

EVM degradation compared to those without switching is mainly attributed to resampling filters. This includes the impact of passband ripple which directly contributes to the increase of EVM, and the inter-subcarrier crosstalk because of the insufficient stopband attenuation. Several representative constellation diagrams are shown in the inset of Figure 3.10, including channels with both QPSK and 16QAM modulation and at different frequency slots.

Table 3.2 FPGA resource cost of DSXC

Resource	**Utilization**	**Available**	**Utilization %**
LUT	38909	433200	8.98
LUTRAM	18929	174200	10.87
FF	60475	866400	6.98
BRAM	96	1470	6.53
DSP slice	2051	3600	56.97

FPGA resources used to build this DSXC are summarized in Table 3.2. This includes 56.97% of the DSP slices usage which is often the bottleneck for this application. There would not be enough DSP slices available with a Xilinx Virtex-7 690t to build this DSXC if the I/Q mixing and filtering method was used.

As mentioned in Chapter 3.3, higher order digital filters help reducing passband ripple and increasing stopband attenuation. FIR filters with lower ripple in the passband and higher suppression in the stopband would result in less EVM degradation. In fact, the EVM percentage of a signal constellation diagram is monotonically increases with the passband ripple. However, increasing the order of digital filters would increase the FPGA resources cost, especially the DSP cost. In addition, higher order digital filters would also introduce longer processing delays for the signal.

In terms of signal processing latency, the latency of this DSXC is mainly introduced by the FIR filters. In fact, the latency of a FIR filter is $t_L = T_C(S-1)/2$, where $S-1$ is the filter order with S the length of filter coefficients, and T_C is the clock period. Latency caused by other utility logic such as cross-bar switch and data type converter is only a few clock periods which is negligible compared to t_L. With a clock period of $T_C = 5ns$, the latency of each FIR filter as the function of the FIR filter order is shown in Figure 3.11. By sending an impulse to the DSXC in the simulation based on Xilinx system generator, the overall latency of DSXC in this system, including two Equiripple 108th order FIR filters and other utility logics, was found to be less than 0.65 μs.

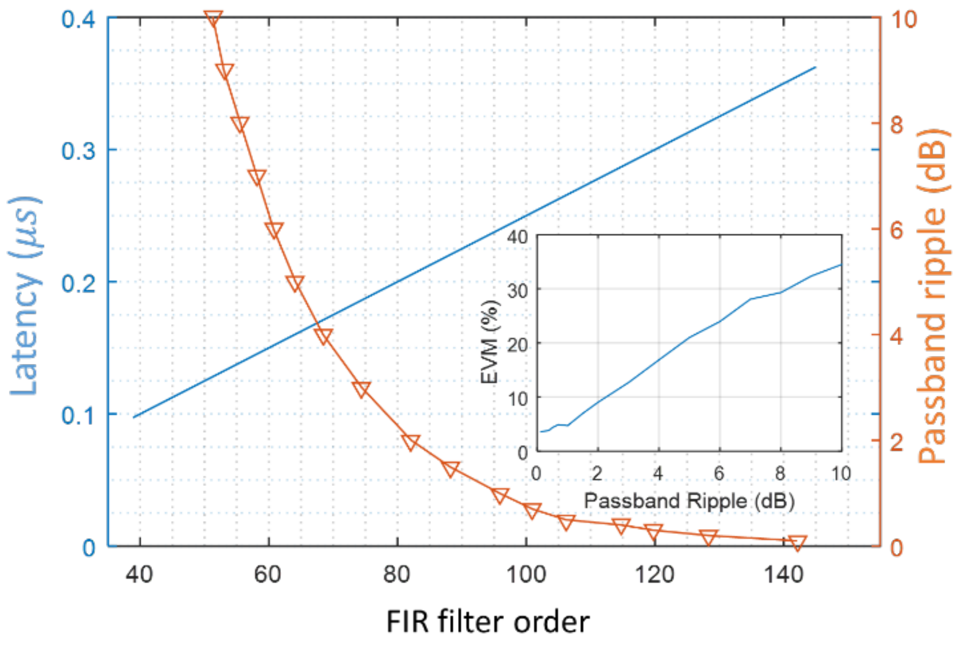


Figure 3.11 Effects vs FIR filter order

The simulation result also shows the impact of filter order on the EVM of the received signal. In the simulation, a subcarrier channel with QPSK signal is selected by an Equiripple FIR filter, and EVM is calculated as the function as filter order, as shown in Figure 3.11. With the increase of filter order, the passband ripple decreases and thus EVM improves. The inset of Figure 3.11

shows the calculated relation between passband ripple and the signal EVM, which indicates that the EVM increases monotonically with the increase of passband ripple.

3.5 Conclusion

We described a real-time DSP-enabled 8×8 digital subcarrier cross-connect (DSXC) test-bed implemented on a Virtex-7 FPGA platform. The functionality and performance of the 8×8 DSXC test-bed are assessed in terms of signal quality, required FPGA resources, and cross-connect data plane latency. Frequency translation of individual subcarrier channels while being routed through the DSXC is achieved through digital resampling filters, implemented on the FPGA. This solution reduces the required FPGA resources when compared to the more conventional I/Q mixing and filtering. To implement I/Q mixing and filtering the amount of required FPGA resources increases linearly with the number of subcarrier channels, while it remains constant when using digital resampling filters.

The experimental results show that the 8×8 DSXC test-bed successfully switches the spectral location of each individual subcarrier channel while it is routed through the DSXC. Each subcarrier channel can be independently assigned a specific bandwidth, modulation format and position in the spectrum. In addition, the circuit-switching DSXC introduces a deterministic and relatively small delay (<1μs) in the data plane compared to the hard-to-predict delay and jitter of commercial packet switches, which depend on the link utilization and packet size.

Due to its fine bandwidth granularity and high spectral efficiency (which cannot be achieve by today’s optical cross-connects), DSXCs are suited for access and metro area networks that support applications with stringent network round trip time requirements, like 5G, cloud assisted robotics, tele-surgery, and real-time gaming. For example, with its capability to mitigate transmission impairments in the digital domain, offer bandwidth flexibility, and support multiple modulation formats, DSXC represents a valid solution to concurrently support and switch a variety of RoF channels in the mobile network fronthaul. Applications of such capabilities have also been discussed for DSP-based analog RoF systems [55-57].

Chapter 4: DA based real-time DSP-enabled DSXC

In this chapter, a real-time DSXC based on distributed arithmetic (DA) architecture is presented. The experimental demonstration of this DSXC is presented in [31]. Compared with the DSXC presented in Chapter 3, the DA-based real-time DSXC has lower latency, lower resource cost and potential lower energy consumption.

4.1 Distributed Arithmetic

First proposed in the early 1970s [58], DA has been used to efficiently implement sum-of-products without using any multipliers [59-61]. In DA architecture, multiplication and accumulation are jointly achieved by using adders, look-up tables (LUTs), and shifters, so that conventional multipliers are not needed. Considering that multipliers are usually the most expensive type of resource in real-time DSP platform, DSP design based on DA architecture can be an advantageous alternative. As a kind of multiply-accumulate circuitry (MAC), FIR filter can be implemented using DA by pre-computing and storing all of the possible results in a LUT. As a consequence, the major drawback of DA-based FIR filter is that the size of its LUT, which must contain the number of possible outcomes, increases exponentially with the number of filter taps. For a FIR with large a number of taps, the LUT size may be too large to be practical. LUT partitioning can significantly reduce the total size of LUT, but at the cost of increased adder complexity and signal latency [61]. With that said, the design of DA FIR filter usually results to be a tradeoff between memory size on the one hand, adder complexity and processing latency on the other [61]. Techniques such as antisymmetric product coding (APC) and odd-multiple-storage (OMS) have been proposed to reduce the LUT size by a factor of two. Another approach, which combines APC and OMS, can further reduce the size of LUT by a factor of four [62]. FIR filter based on DA can be efficiently implemented on hardware such as FPGA or ASIC to support real-time processing [63]. DA-based reconfigurable FIR filters can also be efficiently implemented in FPGA or ASIC [64]. DA-based FIR filters have been implemented to save DSP resources of FPGA for real-time Nyquist pulse generation in the transmitter of an optical communication system [9]. The comparison between real-time Nyquist pulse generation based on DA and real-time OFDM waveform generation based on multipliers showed that DA can greatly reduce the FPGA required resources [9, 14].

In principle, DSP functions such as FIR filters, discrete cosine transform (DCT), FFT, discrete wavelet transform (DWT), image and video processing functions can be implemented using DA architectures [58], and DA-architectures have been used to build traditional filters such as pulse shaping, low-pass and bandpass filters [9, 14]. However, to our best knowledge, DA based resampling filters have not been reported as a technique to simultaneously achieve channel frequency translation and channel selection which are two key functions required in DSXC. We have previously demonstrated digital filtering and frequency translation and channel selection based on resampling filters to reduce DSP resources requirement for DSXC compared to I/Q mixing and filtering [30]. Here we show that DA architecture can further reduce DSP resource consumption and significantly reduce DSXC latency.

In the remainder of this chapter, we demonstrate the implementation of DA-based bandpass resampling filters to achieve simultaneous digital filtering (for channel selection) and frequency translation of a DSXC. In order to support the relatively high data rate optical system applications with GS/s sampling rates provided by ADC and DAC, parallel processing must be applied in the relatively low rate FPGA platform. Processing is achieved through polyphase decomposition, in which a super-sample rate FIR filter is composed of multiple low sample rate sub-filters. LUT partitioning is then applied to each sub-filter implemented in DA to further reduce the LUT size. The major contributions are: 1) the efficient implementation of resampling filters on FPGA hardware through DA architecture to eliminate the need of DSP slices; 2) the use of DA-based resampling sub-filters to support parallel processing of high-speed signals and reduce the LUT size; and 3) the use of DA-based resampling filter algorithm to achieve simultaneous bandpass filtering and frequency translation. Both system performance and hardware resource cost of a DSXC making use of DA-based resampling filters are investigated. For the reader's convenience the principle of DA-based FIR filter design is reviewed in Chapter 4.1.1, and the principle of polyphase decomposition to realize super-sample rate FIR filter is reviewed in Chapter 4.1.2. By utilizing resampling filters, which combine DA architecture and polyphase decomposition, the DA-based DSXC is able to support subcarrier level switching of high-speed signals through parallel processing, subcarrier channel selection, and frequency translation.

4.1.1 Principle of DA

The DA principles are discussed in [59, 63]. For the reader's convenience, we briefly describe the main principle of DA and its application to DA-based FIR filters.

Let A and B be two N-element vectors. Let R be the bit width of each element in vector B. The elements in A can have any bit width. The elements in A are constant values while the elements in B change over time. Equation (4.1) shows the inner-product computation of A and B, which can be obtained using DA as described next.

$$C = \sum_{k=0}^{N-1} A_k \cdot B_k \tag{4.1}$$

Suppose each value in B is represented in the format of 2's complement and is scaled to be $|B| < 1$, then B can be decomposed as shown in Equation (4.2).

$$B_k = -b_{k0} + \sum_{r=1}^{R-1} b_{kr} \cdot 2^{-r} \tag{4.2}$$

Substituting Equation (4.2) into Equation (4.1), the inner-product of A and B can be expanded as in Equation (4.3).

$$C = -\sum_{k=0}^{N-1} A_k \cdot b_{k0} + \sum_{k=0}^{N-1} A_k \cdot [\sum_{r=1}^{R-1} b_{kr} \cdot 2^{-r}] \tag{4.3}$$

Taking the 2^{-r} component out of the bracket in Equation (4.3), we get

$$C = -\sum_{k=0}^{N-1} A_k \cdot b_{k0} + \sum_{r=1}^{R-1} 2^{-r} \cdot [\sum_{k=0}^{N-1} A_k \cdot b_{kr}] \tag{4.4}$$

Signed 2's complement and unsigned offset binary format have the same resource cost if they have the same word size. Without loss of generality, the samples in vector B can be assumed to be unsigned words of size R. Equation (4.4) can be re-written as in Equation (4.5), where the expression of C_r is shown in (4.6).

$$C = \sum_{r=0}^{R-1} 2^{-r} \cdot C_r \tag{4.5}$$

$$C_r = \sum_{k=0}^{N-1} A_k \cdot b_{kr} \tag{4.6}$$

As shown in Equation (4.6), every C_r of $r = 0, 1, \ldots, R-1$, can only be assigned one of 2^N possible values obtained from all possible permutations of the b_{kr} values. The 2^N possible values for C_r can be pre-computed and stored in a LUT.

A DA based FIR filter has a structure similar to the previously depicted inner-product computation between a constant vector A (the filter impulse response) and a time-varying vector B (the input signal). Suppose the impulse response vector of the FIR filter is $\{h(k),\ k = 0, 1, \ldots, N-1\}$ and its input vector is $\{s_n(k),\ k = 0,1, \ldots, N-1\}$, then the output of this FIR filter can be given by Equation (4.7).

$$y(n) = \sum_{k=0}^{N-1} h(k) \cdot s_n(k) \tag{4.7}$$

Here we assume the input sample of the filter is $x(n)$, and $s_n(k) = x(n-k)$. Comparing with Equation (4.6), Equation (4.7) can be rewritten as

$$y(n) = \sum_{r=0}^{R-1} 2^{-r} \cdot C_r \tag{4.8}$$

where

$$C_r = \sum_{k=0}^{N-1} h(k) \cdot (s_n(k))_r \tag{4.9}$$

with $(s_n(k))_r$ being the rth bit of $s_n(k)$.

Equation (4.8) and (4.9) can be directly implemented by pre-computing all of the possible multiplication results and storing them into a LUT. However, since the number of possible values in the LUT (its size) is 2^N, which increases exponentially with the filter length N, this approach is impractical when N is large. In order to reduce the LUT size, the filter length N can be partitioned to form a set of P shorter vectors of coefficients, which require only $2^{N/P}$ values to be stored in the LUT.

Let $N = PM$ (P and M are positive integers), the index k can be mapped into $(m + pM)$ for $m = 0,1,\dots,M-1$ and $p = 0,1,\dots,P-1$. In this case, Equation (4.8) can be rewritten as

$$y(n) = \sum_{r=0}^{R-1} 2^{-r} \cdot \left(\sum_{p=0}^{P-1} (S_n)_{r,p}\right) \tag{4.10}$$

where

$$(S_n)_{r,p} = \sum_{m=0}^{M-1} h(m+pM) \cdot (s_n(m+pM))_r \tag{4.11}$$

for $\mathrm{r} = 0,1,\dots,\mathrm{R}-1$ and $\mathrm{p} = 0,1,\dots,\mathrm{P}-1$.

Since each $(S_n)_{r,p}$ has 2^M possible values this approach requires P relatively small LUTs. Equation (4.10) can be re-written using the memory-read operation of LUT as

$$y(n) = \sum_{r=0}^{R-1} 2^{-r} \cdot \left(\sum_{p=0}^{P-1} \mathcal{F}(\boldsymbol{b}_n)_{r,p}\right) \tag{4.12}$$

where $\mathcal{F}$ is the memory-read operator and $\mathcal{F}(\boldsymbol{b}_n)_{r,p} = (S_n)_{r,p}$. Bit vector $(\boldsymbol{b}_n)_{r,p}$ is used as address word and $(\boldsymbol{b}_n)_{r,p} = [\left(s_n(pM)\right)_r, \left(s_n(1+pM)\right)_r, \cdots, (s_n(M-1+pM))_r]$ for $0 \leq \mathrm{r} \leq \mathrm{R}-1$ and $0 \leq \mathrm{p} \leq \mathrm{P}-1$.

4.1.2 Polyphase decomposition of DA resampling filter

Since the sampling rate of high-speed data converters is much higher than the FPGA clock rate, it is necessary to process data in parallel using polyphase decomposition.

A super sample rate FIR filter is a filter whose sampling rate is higher than its clock rate. The purpose of polyphase decomposition is to support parallel processing, in which a high sampling rate data can be processed with a relatively low speed clock [65]. Here we briefly describe the steps of constructing a super sample rate FIR filter and the steps of constructing super sample rate resampling filters based on the former.

For a FIR filter with N taps, its output can be expressed as in Equation (4.13).

$$y_k = \sum_{i=0}^{N-1} x_{k-i} * h_i \tag{4.13}$$

where $h = [h_0, h_1, h_2, \dots, h_{N-1}]$ is the impulse response of the FIR filter, x is the input vector and y is the output vector. Suppose the degree of parallelism is 4, then the parallel filter structure can be derived as in (4.14).

$$\begin{aligned}
y_0 &= x_0 * h_0 + x_{-1} * h_1 + x_{-2} * h_2 + x_{-3} * h_3 + x_{-4} * h_4 + x_{-5} * h_5 + x_{-6} * h_6 + x_{-7} * h_7 + \cdots \\
y_1 &= x_1 * h_0 + x_0 * h_1 + x_{-1} * h_2 + x_{-2} * h_3 + x_{-3} * h_4 + x_{-4} * h_5 + x_{-5} * h_6 + x_{-6} * h_7 + \cdots \\
y_2 &= x_2 * h_0 + x_1 * h_1 + x_0 * h_2 + x_{-1} * h_3 + x_{-2} * h_4 + x_{-3} * h_5 + x_{-4} * h_6 + x_{-5} * h_7 + \cdots \\
y_3 &= x_3 * h_0 + x_2 * h_1 + x_1 * h_2 + x_0 * h_3 + x_{-1} * h_4 + x_{-2} * h_5 + x_{-3} * h_6 + x_{-4} * h_7 + \cdots
\end{aligned} \tag{4.14}$$

Equation (4.14) can be re-written as Equation (4.15) by re-arranging and re-grouping the multiplications.

$$\begin{aligned}
y_0 = {} & [x_0 * h_0 + x_{-4} * h_4 + x_{-8} * h_8 + \cdots] + [x_{-1} * h_1 + x_{-5} * h_5 + x_{-9} * h_9 + \cdots] \\
& + [x_{-2} * h_2 + x_{-6} * h_6 + x_{-10} * h_{10} + \cdots] + [x_{-3} * h_3 + x_{-7} * h_7 + x_{-11} * h_{11} \\
& + \cdots]
\end{aligned}$$

$$y_1 = [x_1 * h_0 + x_{-3} * h_4 + x_{-7} * h_8 + \cdots] + [x_0 * h_1 + x_{-4} * h_5 + x_{-8} * h_9 + \cdots] + [x_{-1} * h_2 + x_{-5} * h_6 + x_{-9} * h_{10} + \cdots] + [x_{-2} * h_3 + x_{-6} * h_7 + x_{-10} * h_{11} + \cdots]$$

$$y_2 = [x_2 * h_0 + x_{-2} * h_4 + x_{-6} * h_8 + \cdots] + [x_1 * h_1 + x_{-3} * h_5 + x_{-7} * h_9 + \cdots] + [x_0 * h_2 + x_{-4} * h_6 + x_{-8} * h_{10} + \cdots] + [x_{-1} * h_3 + x_{-5} * h_7 + x_{-9} * h_{11} + \cdots]$$

$$y_3 = [x_3 * h_0 + x_{-1} * h_4 + x_{-5} * h_8 + \cdots] + [x_2 * h_1 + x_{-2} * h_5 + x_{-6} * h_9 + \cdots] + [x_1 * h_2 + x_{-3} * h_6 + x_{-7} * h_{10} + \cdots] + [x_0 * h_3 + x_{-4} * h_7 + x_{-8} * h_{11} + \cdots]$$

(4.15)

According to Equation (4.15), the coefficients of the original FIR filter can be polyphase decomposed into four sub-filters, whose coefficients are in Equation (4.16).

$$H_0 = [h_0, h_4, h_8, h_{12}, \cdots]$$

$$H_1 = [h_1, h_5, h_9, h_{13}, \cdots]$$

$$H_2 = [h_2, h_6, h_{10}, h_{14}, \cdots]$$

$$H_3 = [h_3, h_7, h_{11}, h_{15}, \cdots] \quad (4.16)$$

As shown in Equation (4.16), the original FIR filter has been decomposed into 4 sub-filters and the length of each sub-filter is only $1/4$ of the original filter.

Note that it is very important to make sure that all sub-filters (H0, H1, H2 and H3) have the same latency. According to Equation (4.15) and (4.16), the block diagram of this super sample rate FIR filter with a degree of parallelism of 4 is depicted in Figure 4.1.

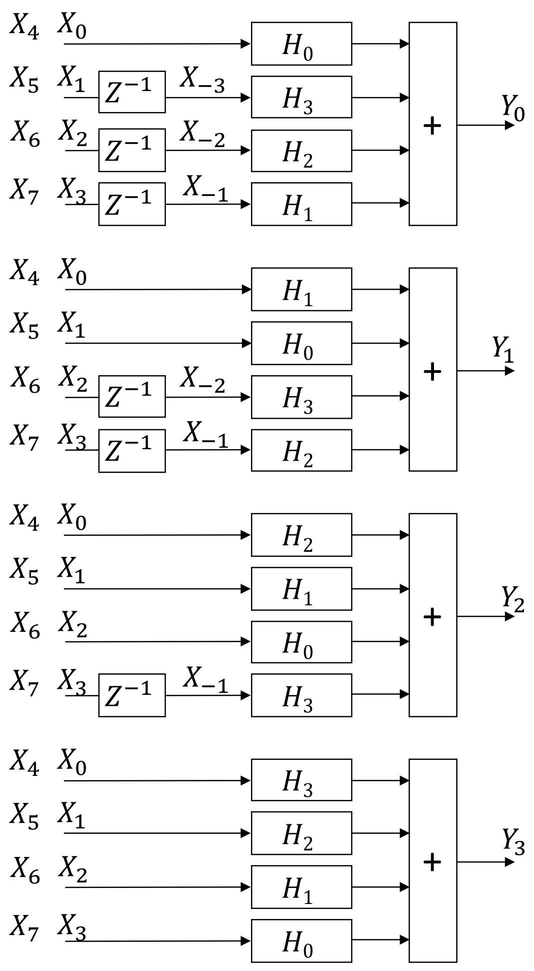


Figure 4.1 A super sample rate FIR filter with a degree of parallelism of 4.

As shown in Figure 4.1, both the input and output of this super sample rate FIR filter have a degree of parallelism of 4, for which the data rate is 4 times the FPGA clock rate. A resampling filter which supports super sample rate can also be derived from the principle of super sample rate FIR filter as shown above. In principle, an interpolation FIR is equivalent to a cascaded process of up-sampling and filtering, while a decimation FIR is equivalent to a cascaded process of down-sampling and filtering. As described in [30], in a resampling filter, the number of operations can be greatly reduced by avoiding the calculation of unnecessary output and the multiplication of a number with zero. For simplicity, we assume the resampling factor L to be equal to the degree of parallelism as depicted in Figure 4.1. For an interpolation FIR filter with the same coefficients as the FIR filter in Fig. 6, its block diagram can be modified as in Figure 4.2.

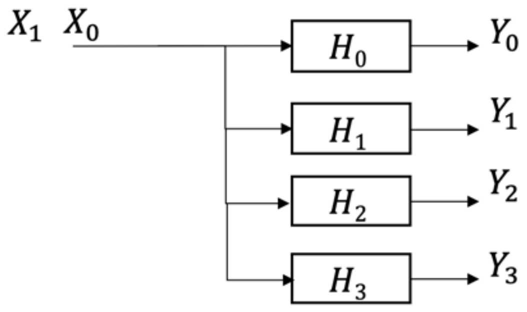


Figure 4.2 Block diagram of an interpolation FIR filter

As shown in Figure 4.2, the output rate of this FIR filter is four times of its input data rate, and the number of sub-filters is 4, which is only $1/4$ of the FIR filter depicted in Figure 4.1. Similarly, for a decimation FIR filter with the same coefficients as the FIR filter in Figure 4.1, its block diagram can be modified into the block diagram of a decimation FIR filter as in Figure 4.2. As shown in Figure 4.3, the output data rate of this decimation FIR filter is one fourth of its input data rate, and its number of required sub-filters is 4, which is one fourth of the FIR filter depicted in Figure 4.3, so the LUT size is also reduced by a factor of 4. In conclusion, compared with super sample rate FIR filters, the resource cost of a resampling FIR filter decreases by a factor equal to the resampling factor.

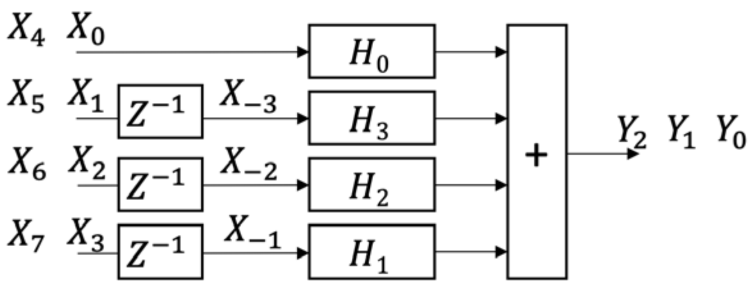


Figure 4.3 Block diagram of a decimation FIR filter

4.2 Experiment

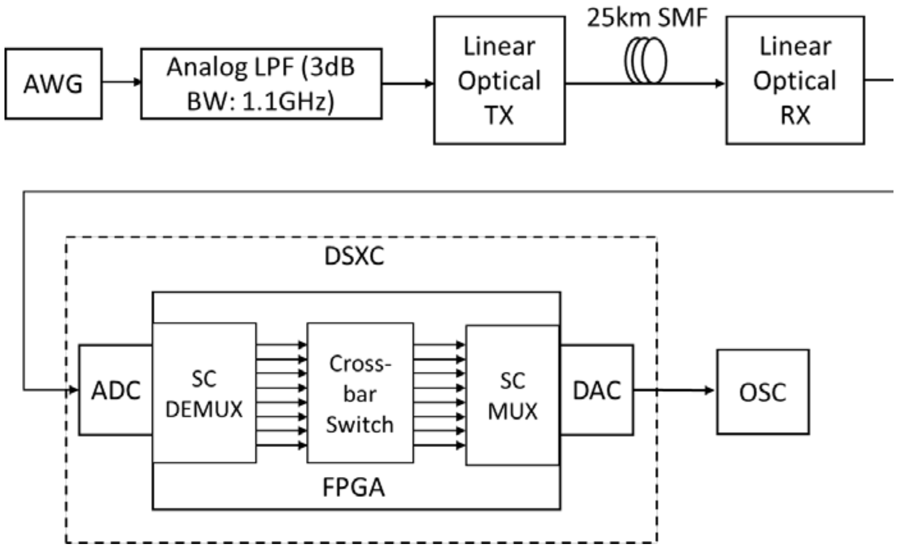


Figure 4.4 Experimental setup

The DA architecture ability to perform subcarrier channel frequency down-conversion, up-conversion and digital filtering is tested out using the DSXC node and fiber-optic system shown in Figure 4.4. The composite DSCM signal used as input to the optical transmitter consists of multiple digital subcarriers generated by an arbitrary waveform generator (AWG). The AWG-generated composite DSCM signal is filtered by an analog low-pass filter (LPF) whose 3dB bandwidth is 1.1GHz. The filtered DSCM signal is then converted into an optical signal at 1310nm wavelength by intensity modulation of an optical transmitter with approximately 1mW average optical power. After propagation over 25km of single mode fiber (SMF), the optical signal is detected by an optical receiver with direct-detection, and converted back in to a RF signal before being sent to the DSXC. In the DSXC, the electrical signal is digitized by an ADC with a sampling rate of 1.6GS/s and a resolution of 12 bits per sample. Then the digitized signal is transferred to a FPGA for subcarrier level cross-connect switching, which includes subcarrier de-multiplexing, cross-bar circuit switching, and subcarrier multiplexing. At the DSXC output the processed data is sent to a DAC where it is converted to form an analog waveform. The DAC has an input sampling rate of 1.6GS/s and a resolution of 16 bits per sample. The waveform at the DAC output is recorded by a real-time digital oscilloscope (OSC) for offline processing and evaluation of the signal quality after the DSXC.

In this experiment, the DSCM signal generated by the AWG consists of eight subcarriers (SCs), and each SC carries a 16QAM signal that occupies a bandwidth of 80MHz. 20MHz is reserved as the guard band between adjacent SCs. Since the available bandwidth of the ADC is 800MHz (according to the Nyquist theorem), up to 8 DSCM channels can be supported. In the design of the resampling filters, a resampling factor of 8 is used, which equally divides the total available bandwidth of 800MHz into 8 frequency slots (FSs) each with 100MHz bandwidth. More in general, the resampling factor may vary from SC channel to SC channel depending on the bandwidth that is assigned to each SC channel to match its individual data rate and modulation format.

Channel selection, frequency translation, and switching of all 8 subcarrier channels are performed using 8 pairs of FIR filters implemented at the input and output of the DA-DSXC. Each pair of FIR filters consists of one decimation (input) and one interpolation (output) filter to perform down-conversion and up-conversion, respectively. In this experiment, equiripple FIR filters are used at 1.6 GS/s sampling rate, with 80MHz width of passband and 20MHz width of transition band. In order to achieve desirable performance, the FIR filter is designed to have a passband ripple A_{pass} = 0.5dB, and a stopband attenuation A_{stop} = 40dB. With the above filter specifications, the FIR filter order is 134 (unless otherwise specified) as determined by a filter design tool available in Matlab. Coefficients of the FIR filters are obtained using the FIRPM function in Matlab.

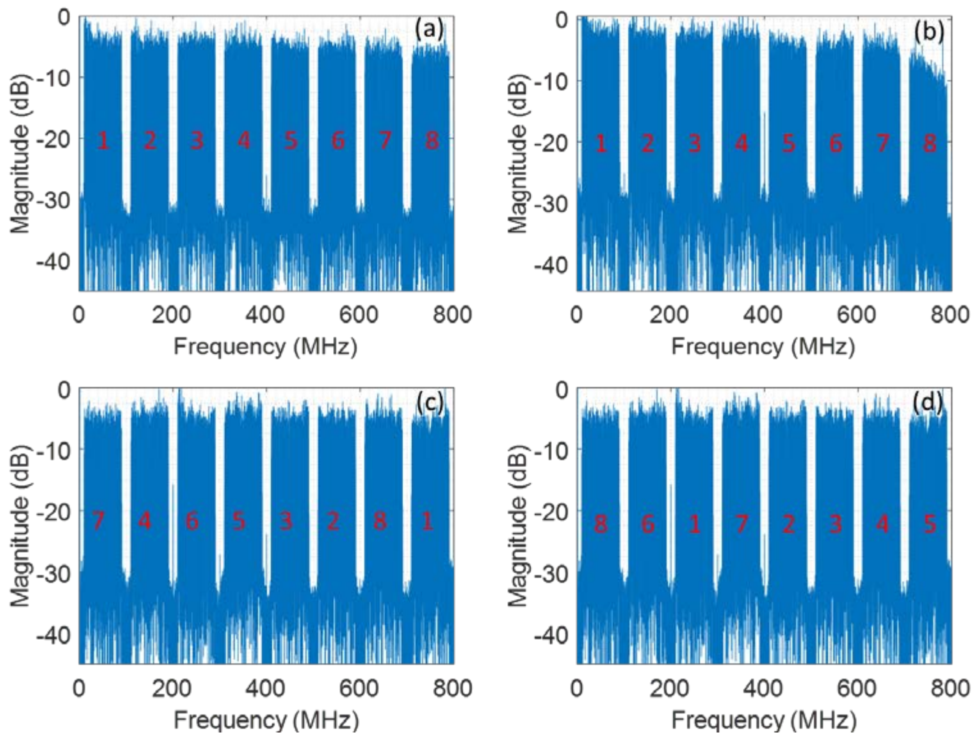


Figure 4.5 Spectrum of (a) output of optical receiver (b) output of DAC without cross-connect switching (c) output of DA-DSXC1 (d) output of DA-DSXC2

For tracking purposes, each SC channel generated by the AWG is assigned a unique identifier [1 2 3 4 5 6 7 8], counting from the lowest frequency to the highest frequency as marked on the spectrum shown in Figure 4.5 (a). All the SC channels are assigned the same power at the AWG. Since the channel has a flat frequency response in the signal band, the SC channels at the DA-DSXC input also have same power. Through the DA-DSXC, these SC channels can be switched from any input FS to any output FS. Figure 4.5 (b) shows the spectrum measured at the DA-DSXC output when the SC channels relative positions are not changed, i.e., channel selection and frequency translation are not applied yet. There is approximately a 10dB roll-off at the highest frequencies, which accounts for the combined transfer function of the optical transmitter, receiver, ADC and DAC circuits.Two distinct channel reassignments at the DA-DSXC output are tested, i.e., DA-DSXC1 [7 4 6 5 3 2 8 1] and DA-DSXC2 [8 6 1 7 2 3 4 5], respectively.

Figure 4.5 (c) and (d) show the post-compensated spectra of the DSXC output for the configurations of DA-DSXC1 and DA-DSXC2, respectively. In these two experiments the roll-off effects of the transmission system are post-compensated offline at the receiver for ease of implementation. However, this compensation can also be performed in real-time by incorporating in the FPGA design filters with frequency responses that are inverse to the roll-off effects.

For the purpose of comparison, we also built a DSXC using resampling filters based on multipliers [30], which has the same switching capabilities as the DA-based DSXC. We refer to this multiplier-based DSXC as MULT-DSXC. Both DA-DSXC and MULT-DSXC are implemented in the same Virtex-7 FPGA platform and employing the same type of resampling FIR filters in terms of orders and coefficients. The output of MULT-DSXC is chosen to match the same two channel switching patters defined earlier. i.e., MULT-DSXC1 [7 4 6 5 3 2 8 1] and MULT-DSXC2 [8 6 1 7 2 3 4 5].

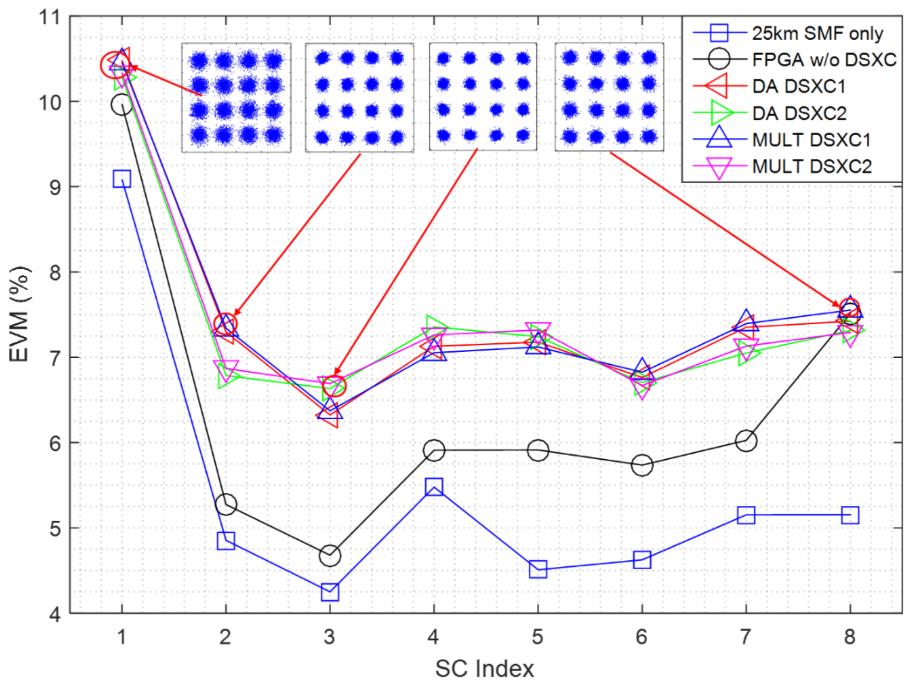


Figure 4.6 Signal EVM of recovered subcarriers

Figure 4.6 shows the error vector magnitude (EVM) for each of the eight subcarrier channels under six different configurations. Open squares show the subcarrier EVM after 25km of SMF transmission at the input of the DSXC. Due to the transceiver low frequency cut-off at 30MHz, the lowest frequency subcarrier channel has an abnormally high EVM. Open circles show the subcarrier EVM at the DSXC output in the absence of any digital processing (simple pass-through). The comparison between open squares and open circles indicates that the EVM values increase by an average of about 1%, due to both the digitizing noise and the non-flat frequency response of the ADC and DAC. When the switching functionality of DSXC is activated, resampling filters are applied to the signals to allow subcarrier frequency up- and down-conversion. Triangles show the EVM values at the DSXC output in four configurations: left- and right-pointing triangles show the EVM values of DA-DSXC1 and DA-DSXC2, while upward- and downward-pointing triangles show the EVM values of MULT-DSXC1 and MULT-DSXC2, respectively. These results clearly indicate that DA-based and multiplier-based resampling filters yield similar performance, as the EVM values for DA-DSXC1 and DA-DSXC2 are essentially the same as those for MULT-DSXC1 and MULT-DSXC2.

In our experiment, each subcarrier channel carries a set of independent data with a modulation format of 16QAM. According to [66, 67], the required EVM threshold for LTE-A is 12.5% for 16QAM. Figure 4.6 shows that this DSXC implementation meets this EVM requirement. In addition to avoid frequency cut-off by the optical transceiver, the signal quality can be further improved by increasing the order of the DA FIR filter, which results in a lower passband ripple and higher stopband attenuation of the FIR filter. However, a higher order DA FIR filter costs more LUTs in the FPGA. A tradeoff between the filter performance and resource consumption has to be found in the design. Both passband ripple and stopband attenuation are dependent on the filter order, and they affect the signal quality. More specifically, passband ripple introduces frequency dependent loss of the signal spectrum, while non-adequate stopband attenuation would introduce crosstalk between closely spaced subcarrier channels. Both of these two effects can significantly deteriorate signal EVM.

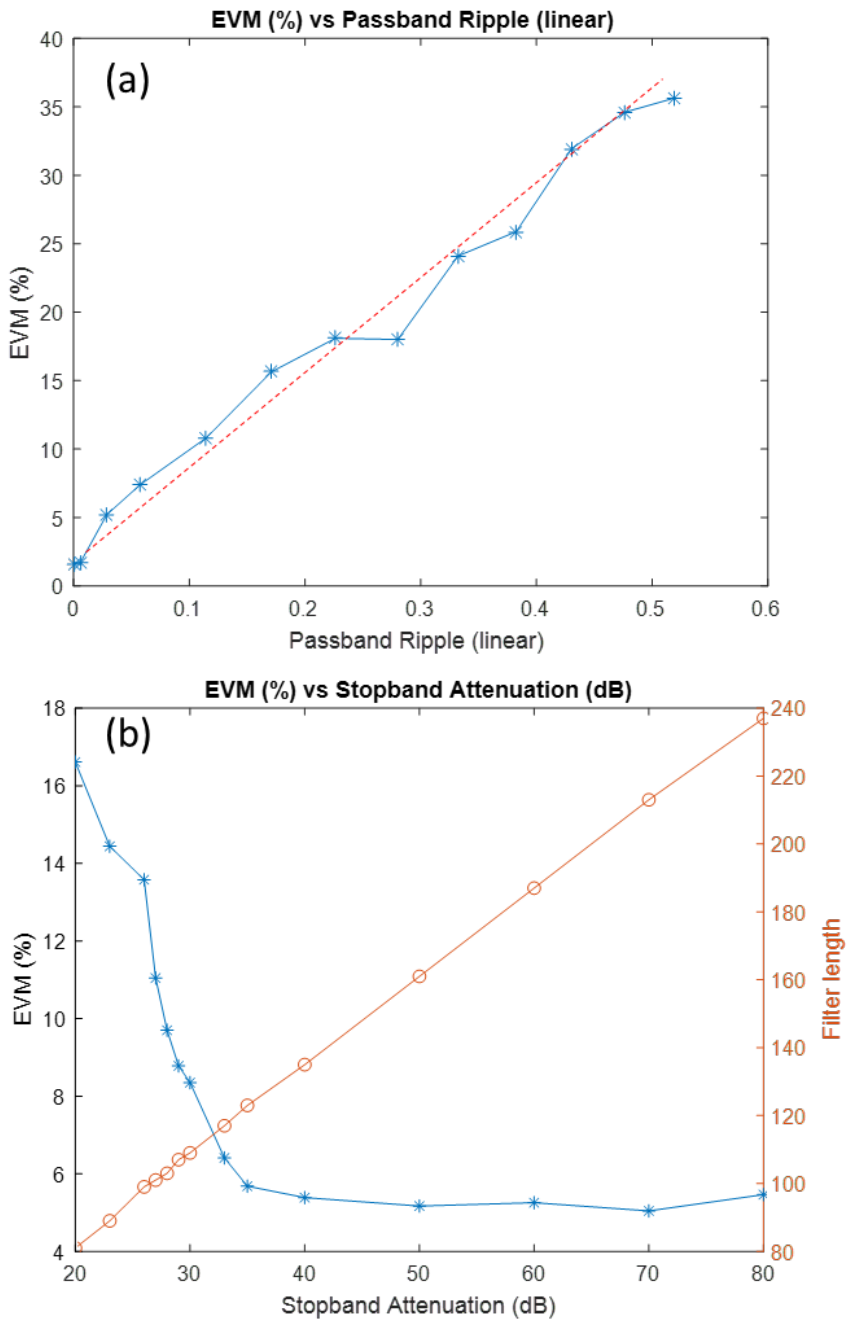


Figure 4.7 (a) EVM vs Passband Ripple, and (b) EVM vs Stopband Attenuation

Figure 4.7 (a) shows the EVM of a subcarrier channel after passing through a FIR filter with different values of passband ripple. The simulation has been conducted by sending a signal with 8 subcarriers into the bandpass FIR filter. The stopband attenuation is kept constant at A_{stop} = 40dB while changing the passband ripple through the change of the filter order. Figure 4.7 (a) indicates that EVM increases linearly with the increase of the passband ripple. The positions of frequency peaks and notches in the passband ripple also have a minor impact on the EVM. Consequently the calculated EVM does not exactly follow a straight line in Figure 4.7 (a). The major impact of insufficient stopband attenuation is the crosstalk from other subcarrier channels. In the frequency down-conversion process, the resampling FIR filter selects a particular subcarrier channel, rejects other subcarriers, and shifts the selected subcarrier to the lowest frequency slot. If stopband attenuation is not high enough, the leakage from all other 7 subcarriers will be shifted to the lowest frequency slot, generating crosstalk. In the frequency up-conversion process, after up-sampling, every selected subcarrier has 8 copies equally spread across the 8 frequency slots. After bandpass filtering with insufficient stopband attenuation, the leakage from all other 7 subcarriers would contribute to crosstalk. To evaluate the impact of stopband attenuation A_{stop} in the DSXC node, simulation is carried out with a fixed passband ripple of 0.5dB, and A_{stop} is varied by changing the filter order. Figure 4.7 (b) shows the calculated EVM as a function of stopband attenuation. For $A_{\text{stop}} < 40$dB, EVM improves rapidly with the increase of A_{stop} due to the significant reduction of inter-channel crosstalk. The EVM improvement saturates when A_{stop} approaches 40dB, at which point the crosstalk impact becomes insignificant. With a fixed passband ripple, the stopband attenuation increases linearly with the filter length (number of taps) as indicated by the right vertical axis of Figure 4.7 (b). As previously mentioned, by setting A_{pass} = 0.5dB and A_{stop} = 40dB, the order of the FIR filter is 134, which is the value chosen in this study.

4.3 Resource requirement and discussion

Compared with MULT-DSXC, DA-DSXC has three advantages: 1) it does not require expensive DSP slices in the FPGA implementation; 2) the DSP-induced latency is only a few FPGA clock periods and is independent of the filter order; and 3) power consumption is reduced as massive DSP multiplications are avoided. These three aspects are discussed. Most of the terms used in this section are defined in Section 4.1.1 and 4.1.2.

4.3.1 Resource Utilization

The major resource cost of a DA-based FIR filter is the lookup table (LUT). Consider a FIR filter with N taps and W bit width of LUT data. Let G be the bit width of the input data. A fully serial implemented DA FIR filter processes 1 bit per clock period (equivalent to process 1 sample per G clock periods), which means its latency is G clock periods. For a FIR filter with asymmetric coefficients, its LUT size (without LUT partition) is $W \cdot 2^N$ bits. Partitioning the LUT can reduce its size by subdividing a LUT into several smaller LUTs. If we perform a M-fold LUT partition, such as $N = N_1 + N_2 + \cdots + N_M$, then the total LUT size becomes $W_1 \cdot 2^{N_1} + W_2 \cdot 2^{N_2} + \cdots + W_M \cdot 2^{N_M}$ bits, where W_i is the bit width of the LUT data which is obtained through the multiplication of coefficients and allowed input data. The value of W_i, which is determined by the bit width of input data, bit width of coefficients, and the LUT partition, is typically smaller than the bit width of the output data. The LUT size can be further reduced by skipping the zero-valued coefficients [61]. In this case the zero-valued coefficients are ignored when LUT partition is performed. If this DA FIR filter is fully parallel implemented, in which it processes G

bits per clock period (equivalent to process 1 sample per clock period), its LUT size is G times that of the fully serially implemented DA FIR filter. In this case, the LUT size of a fully parallel DA FIR is $(W_1 \cdot 2^{N_1} + W_2 \cdot 2^{N_2} + \cdots + W_M \cdot 2^{N_M}) \cdot G$. For example, consider a FIR filter with 12 taps and 12 input bit width, LUT partition of [6 6 2] and corresponding data bit widths of [11 14 8]. If fully serially implemented, its LUT size is $11 \times 2^6 + 14 \times 2^6 + 8 \times 2^2 = 1{,}632$ bits. If fully parallel implemented, its LUT size is $(11 \times 2^6 + 14 \times 2^6 + 8 \times 2^2) \times 12 = 19{,}584$ bits.

For a super sample rate FIR filter based on DA architecture, the estimation of its LUT needs to consider its polyphase decomposition, which is determined by the degree of parallelism. The polyphase decomposition process decomposes this FIR filter into multiple sub-filters, as described in Chapter 4.1.2. Each sub-filter can be treated as a small FIR filter and its LUT size can be estimated by the method described in the previous paragraph, so that the LUT size of the super sample rate FIR filter can be estimated by summing up the LUT sizes of all sub-filters. For example, suppose the super sample rate FIR filter has N taps and has a degree of parallelism of L, then the number of taps of each sub-filter is N/L. For simplicity, we assume N is an integer multiple of L and there are no zero-valued coefficients. As described in Chapter 4.1.2, this super sample rate FIR filter consists of L^2 sub-filters with N/L taps in each sub-filter. Suppose LUT partition is not performed and the bit width of LUT data is W and each sub-filter is fully parallel implemented, then the LUT size of each sub-filter is $W \cdot 2^{N/L} \cdot G$ and the LUT size of this super sample rate FIR filter is $W \cdot 2^{N/L} \cdot G \cdot L^2$. However, the LUT size might be too large if the value N/L is relatively large, so the LUT size can be further reduced through LUT partition.

For a resampling filter which has a degree of parallelism of L, suppose the resampling factor is M, then the resource cost of a resampling FIR filter is only $1/M$ of that of a FIR filter with the same coefficients, so its LUT size is $W \cdot 2^{\frac{N}{L}} \cdot G \cdot L^2/M$. In our DSXC design, the degree of parallelism is 8 and the resampling factor is 8, so the LUT size of each DA based resampling filter is $W \cdot 2^{\frac{N}{8}} \cdot G \cdot 8$. In our system, the ADC resolution is 12 bits, W = 12, the length of coefficients is N = 134. In this case, the filter's LUT size without LUT partition is $12 \times 2^{\frac{134}{8}} \times 12 \times 8 = 1.27 \times 10^8$ bits, which is too large and not practical for hardware implementation. Since N/8 = 16.75, the length of each sub-filter is approximately 17. If LUT partition is performed as [6 6 5], then the LUT size of a resampling filter becomes$(12 \times 2^6 + 12 \times 2^6 + 12 \times 2^5) \times 12 \times 8 = 184{,}320$ bits. After the LUT partition, the LUT size is scaled down to a value that is practical for implementation and this resampling filter can be efficiently implemented with FPGA.

However, there is no analytic formula to accurately estimate the amount of LUTs that are exactly used in FPGA hardware. This is because the mapping from HDL design of DA filter to hardware implementation is a complicated process that is affected by many factors such as the architecture of DA filter, the FPGA tool, and the type of targeted device. Nevertheless, Xilinx Vivado, which conducts the process of this mapping, can provide estimations of resource cost of the design for the targeted device. The mapping performed by Xilinx Vivado consists of two stages: synthesis and implementation. The synthesis process maps the HDL design to netlist, and the implementation process maps the synthesized netlist to the available resources on the targeted device and generates bit-stream file to be downloaded to FPGA hardware. The Xilinx Vivado reports the FPGA hardware resource utilization after synthesis and implementation, respectively. Only the post-implementation resource utilization reveals the actual hardware cost on FPGA. In our experiment, there are 433,200 available 6-input LUTs on a Virtex-7 690T FPGA chip. For convenience, the term LUT cost refers to the number of needed LUTs on the FPGA hardware after synthesis and implementation. The resource utilization is then evaluated through Xilinx Vivado after synthesis and implementation.

Table 4.1 FPGA resource utilization of DA DSXC IP and MULT DSXC IP

FPGA Resource	**Available resource**	**Post-synthesis**			
		DA DSXC IP		**MULT DSXC IP**	
		Utilization	**Utilization %**	**Utilization**	**Utilization %**
LUT	433,200	111,711	25.79	29,435	6.79
LUTRAM	174,200	0	0	26,904	15.44
FF	866,400	108,412	12.51	30,923	3.57
BRAM	1,470	0	0	0	0
DSP slice	3,600	0	0	2,147	59.64

Since the designed DSP unit of DSXC is packaged into an intellectual property (IP) that can be conveniently imported into a Vivado project, we use the term DSXC IP to refer to the design of DSXC inside FPGA. Table 4.1 shows the FPGA resource utilization for both DA DSXC IP and MULT DSXC IP, which mainly consists of FIR filters. Before importing the DSXC IP into the FPGA project that contains all other logics, we estimated its resource cost by running synthesis under Xilinx Vivado. The post-synthesis results show that compared with MULT DSXC IP, DA DSXC IP consumes more LUTs and flip-flops (FFs), but it does not consume any DSP slice which is most often the bottleneck of the hardware resources.

It is important to point out that for MULT-DSXC, the number of DSP slices is equal to the number of required multipliers in the design of FIR filters. Instead of using DSP slice, a multiplier can also be built by only using LUTs. According to the synthesis results of the LUT based multiplier IP in Vivado, a 12×16 bit multiplier consumes 204 LUTs in the Virtex-7 690T FPGA chip. Since the number of required multipliers in the MULT DSXC IP is 2147, a total of

2,147 × 204 LUTs would be required if all multipliers are implemented with LUTs. So, the total cost of LUTs for this MULT DSXC IP would be 2,147 × 204 + 29,435 = 467,423, which exceeds the number of total available LUTs on Virtex-7 690T FPGA. Another problem with this FIR filter implementation is its linear dependence on the filter order. Therefore, it is more efficient to implement DA based FIR filters on hardware instead of implementing FIR filters based on multipliers, even when each multiplier is implemented by LUTs.

Table 4.2 FPGA resource utilization of DA DSXC and MULT DSXC

FPGA Resource	**Available resource**	**Post-synthesis**		**Post-implementation**	
		DA DSXC	**MULT DSXC**	**DA DSXC**	**MULT DSXC**
		Utilization %	**Utilization %**	**Utilization %**	**Utilization %**
LUT	433,200	31.17	12.18	29.52	9.12
LUTRAM	174,200	1.37	16.81	1.10	11.34
FF	866,400	16.08	7.13	15.58	6.60
BRAM	1,470	6.53	6.53	6.53	6.53
DSP slice	3,600	0	59.64	0	59.64

Table 4.2 shows the FPGA resource utilization of MULT DSXC and DA DSXC, which contain all other utility logics, after synthesis and implementation. Since the DSXC design includes utility logics such as the Microblaze IP for controlling and JESD204B IPs for interfacing ADC, DAC, and FPGA, we can run synthesis and implementation in Xilinx Vivado to estimate the overall resource cost. It shows that in both cases of post-synthesis and post-implementation, DA DSXC does not consume DSP slices. The table shows that the MULT DSXC consumes nearly 60% of the total available DSP slices on a Virtex-7 690T FPGA. This means that increasing the filter order may quickly use up all the available DSP slices. With the same functionality and performance, the DA DSXC requires 0 DSP slice, but increases the use of LUTs from 9.12% to

29.52%, allowing for more room to increase the switching capability with the remaining hardware resources. Increasing the available memory in digital hardware is also significantly cheaper than increasing the highly specialized DSP slices in FPGA.

4.3.2 Latency

If we define the latency of a FIR filter as the delay between the time of occurrence of the first non-zero input and the first non-zero output of the FIR filter, then traditional direct-implemented FIR filter based on multipliers has a latency which is proportional to its filter order. Whereas the latency of a DA-based FIR filter is mainly introduced by the reading operation of LUTs and the shifting and adding operation of digital sequence, which is fixed and is independent of the filter order.

In order to have a filter with linear phase, we need to design a FIR filter whose coefficients are symmetric around its center. Theoretically, for a direct-implemented FIR filter with symmetric coefficients and a filter order of N, its latency is $N \cdot T_{clk}/2$, where T_{clk} is the clock period of the digital circuit. For the DA filter used in our DSXC, its latency is 2 clock periods if implemented in fully parallel architecture without pipelining. In order to meet the timing constraint of the FPGA design, the DA FIR has been pipelined and its latency is increased to 7 clock periods. According to the block diagram in Figure 4.1, the time shift delay and summation operation add an additional delay of 2 clock periods, so that a DA FIR filter in our design has an overall latency of 9 clock periods. Since each subcarrier needs a down-conversion and an up-conversion which require two resampling filters, the filter induced latency of a DA DSXC is 19 clock periods including the 1 clock latency introduced by the multiplexer. With the clock period of 5 ns in the FPGA platform that we used, the accumulated latency of this DA DSXC is about 0.1 μs due to DSP. For the MULT DSXC, the filter induced latency is $(N+1) \cdot T_{clk}$, where N = 132 is the order of filter used in our design. Thus, the total filter-induced latency of MULT DSXC is approximately 0.67 μs. Note that the actually latency due to DSXC IP might be slightly longer than the theoretical estimation, since there are some other utility logics that may increase the latency by a few clock cycles.

As a circuit-based cross-connect, the DSXC has a deterministic latency. The DSXC latency mainly comes from the data converters (ADC/DAC), the data interfaces between converters, FPGA, and the DSXC IP inside FPGA. In order to measure the DSXC actual latency, we built three FPGA projects: in the 1st project the signal passes through the system without any DSP processing; in the 2nd project the signal passes through MULT DSXC; and in the 3rd project the signal passes through DA DSXC. For each project, we sent a triangular waveform with relatively long period (5 μs) and compared the delay between the falling edges of the transmitted waveform (input) and received waveform (output). The measured latency of the 1st project is 1.82 μs, which is caused by the signal path between the input of ADC, output of DAC, and the interfaces between ADC, DAC, and FPGA board. This latency can be greatly reduced by integrating ADC, DAC and FPGA onto a single chip. The measured latencies of the 2nd project (MULT DSXC) and the 3rd project (DA DSXC) are 2.75 μs and 1.96 μs, respectively. Both of them are longer than the latency of the 1st project because of the additional processing latency introduced by the DSXC IP. In this experiment, the additional latency introduced by DSXC IP of MULT DSXC is 0.93 μs while the latency introduced by DSXC IP of DA DSXC is 0.14 μs, both slightly longer than the corresponding theoretical estimations presented earlier. Nonetheless, the achievable reduction of processing latency through the use of DA-based resampling filters is confirmed.

4.3.3 Power Consumption

In terms of electrical power consumption, the post-implementation results of the FPGA project show that the on-chip power consumption of MULT DSXC and DA DSXC are both approximately 12W. In practical applications, FPGAs are usually used for DSP prototyping, while the final designs are often integrated into task specific ASICs. As discussed in [65], for a generic DSP design there is a mapping relation between the integrated circuit (IC) area required in FPGAs and the IC area required in ASICs. According to [65], the area required to implement LUT in ASICs is on average 35 times smaller than that in FPGAs, while the area required to implement multipliers in ASICs is on average only 25 times smaller than that in FPGAs. As MULT-DSXC uses a large number of multipliers while DA-DSXC only uses LUTs, after converting from the FPGA design to ASIC design, the IC area required to implement DA-DSXC is estimated to be on the order of 70% of that required to implement MULT-DSXC, and thus, there is a potential for the reduction of power consumptions in ASIC design.

4.4 Conclusion

We demonstrated the use of DA-based resampling filters for both frequency translation and channel selection in DSXC. Compared with traditional FIR filters, which are based on multipliers and require costly DSP slices to be implemented in FPGA, the DA algorithm makes use of look-up-tables, which require only digital memories that are usually more abundant and less costly. DA-based resampling filters provide a hardware resource-efficient solution for implementing DSXC, which must be able to switch multiple digital subcarrier channels from any input to any output port. In addition, a DA-based resampling filter has reduced processing latency compared with a multiplier-based FIR filter with same transfer function. We have experimentally implemented a real-time 8x8 DSXC in a Xilinx Virtex-7 FPGA platform, and investigated the signal EVM penalties introduced by the DSXC. A comparison based on both required hardware resources and introduced processing latency was presented between a DA-based DSXC implementation and a multiplier-based DSXC implementation. The experimental results show that a DSXC using DA-algorithm for frequency translation and channel selection is a suitable technology to provide subcarrier circuit switching cross-connection in optical networks, and may find useful applications in 5G mobile fronthaul, where improved spectral efficiency and flexibility are of the essence.

Chapter 5: Conclusion and Future work

5.1 Conclusion

This dissertation focuses on the investigation of real-time DSP-enabled DSXC that aims to improve the spectral efficiency and provide a more flexible schemes of cross-connect switching in future optical networks. We presented the principle of DSXC and demonstrated two different implementations of them.

The main contribution of this research work consists of three parts: 1) investigated and experimentally demonstrated the first real-time DSP-enabled DSXC; 2) introduced resampling filters in the design of DSXC to reduce the computational cost of the expensive multipliers, and to limit the resource cost increment with the number of subcarriers; 3) introduced DA architecture in the design of DSXC based on resampling filters to eliminate the use of multipliers and reduce the processing latency and potential power consumption.

5.2 Future work

There are many other interesting topics worth further investigation.

The DSXC presented in this dissertation is based on Nyquist-FDM. Since DSCM can also be based on OFDM, it is possible to implement OFDM-based DSXC, and to analyze its resource cost, power consumption and performance, and to compare it with its Nyquist-FDM-based counterpart.

Limited by the capability of hardware platform, we mainly demonstrated the function of frequency translation and channel selection of one DSXC. It would be interesting to investigate the performance of multiple DSXC nodes, and to evaluate their performance in optical networks. The performance of DSXCs with high number of subcarriers can also be investigated. For example, a ZCU111 FPGA board has eight ADCs, each with a maximum sampling rate of 4.096 GSPS, and eight DACs, each with a maximum sampling rate of 6.4 GSPS. With a new hardware platform based on the ZCU111 FPGA board, a DSXC with more wavelength channels, as well as more subcarriers, can be implemented. The increasing number of subcarriers may lead to a high peak to average power ratio (PAPR), which mandates further investigation. This new platform also has more hardware resources (DSP, BRAM, LUT, etc.) that may enable implementing a DSXC with higher performance powers. The hardware imperfections, such as ADC nonlinearity and DAC spectral roll-off, can also be compensated by using low order FIR filters implemented within the FPGA.

References

[1] O. Gerstel, M. Jinno, A. Lord, and S. B. Yoo, “Elastic optical networking: A new dawn for the optical layer?” *IEEE Communications Magazine*, *50*(2), 2012.

[2] M. Jinno, H. Takara, B. Kozicki, Y. Tsukishima, Y. Sone, and S. Matsuoka, “Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies,” *IEEE communications magazine*, 2009, 47(11): 66-73.

[3] K. Christodoulopoulos, I. Tomkos, and E. A. Varvarigos, "Elastic bandwidth allocation in flexible OFDM-based optical networks," *Journal of Lightwave Technology,* 29.9 (2011): 1354-1366.

[4] K. Christodoulopoulos, I Tomkos, and E. A. Varvarigos, "Dynamic bandwidth allocation in flexible OFDM-based networks," *Optical Fiber Communication Conference*. Optical Society of America, 2011: OTuI5.

[5] M. Jinno, B. Kozicki, H. Takara, A. Watanabe, Y. Sone, T. Tanaka, and A. Hirano, “Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network [topics in optical communications],” *IEEE Communications Magazine*, 48(8), (2010).

[6] P. M. Hill, and R. Olshansky, "A 20-channel optical communication system using subcarrier multiplexing for the transmission of digital video signals," *Journal of lightwave technology* 8.4 (1990): 554-560.

[7] R. Hui, B. Zhu, R. Huang, C. T. Allen, K. R. Demarest, and D. Richards, "Subcarrier multiplexing for high-speed optical transmission," *Journal of lightwave technology* 20, no. 3 (2002): 417.

[8] C. Laperle, and M. O'Sullivan, "Advances in high-speed DACs, ADCs, and DSP for optical coherent transceivers," *Journal of lightwave technology*, 2014, 32(4): 629-643.

[9] E. Dutisseuil, J. M. Tanguy, A. Voicila, R. Laube, F. Bore, H. Takeugming, F. de Dinechin, F. Cerou, and G. Charlet, "34 Gb/s PDM-QPSK coherent receiver using SiGe ADCs and a single FPGA for digital signal processing," *Optical Fiber Communication Conference*, Optical Society of America, 2012: OM3H. 7.

[10] C. Fludger, J. C. Geyer, T. Duthel, S. Wiese, and C. Schulien, "Real-time prototypes for digital coherent receivers," *Optical Fiber Communication Conference*, Optical Society of America, 2010: OMS1.

[11] A. Leven, N. Kaneda, and S. Corteselli, "Real-time implementation of digital signal processing for coherent optical digital communication systems," *IEEE Journal of Selected Topics in Quantum Electronics*, 2010, 16(5): 1227-1234.

[12] A. Leven, N. Kaneda, and Y. K. Chen, "A real-time CMA-based 10 Gb/s polarization demultiplexing coherent receiver implemented in an FPGA," *Optical Fiber Communication Conference*, Optical Society of America, 2008: OTuO2.

[13] R. Schmogrow, M. Winter, M. Meyer, D. Hillerkuss, S. Wolf, B. Baeuerle, A. Ludwig, B. Nebendahl, S. Ben-Ezra, J. Meyer, M. Dreschmann, M. Huebner, J. Becker, C. Koos, W. Freude, and J. Leuthold, "Real-time Nyquist pulse generation beyond 100 Gbit/s and its relation to OFDM," Optics Express, 2012, 20(1): 317-337.

[14] R. Schmogrow, R. Bouziane, M. Meyer, P. A. Milder, P. C. Schindler, R. I. Killey, P. Bayvel, C. Koos, W. Freude, and J. Leuthold, "Real-time OFDM or Nyquist pulse generation–which performs better with limited resources? " Optics Express, 2012, 20(26): B543-B551.

[15] W. Jin, X. Duan, Y. Dong, B. Cao, R. P. Giddings, C. Zhang, K. Qiu, and J. M. Tang, “DSP-enabled flexible ROADMs without optical filters and OEO conversions,” *Journal of Lightwave Technology*, 2015, 33(19): 4124-4131.

[16] R. P. Giddings, E. Al-Rawachy, and J. M. Tang, “Experimental Demonstration of Real-Time Add/Drop Operations in DSP-enabled Flexible ROADMs for Converging Fixed and Mobile Networks,” *Optical Fiber Communication Conference*, Optical Society of America, 2018: W2A. 33.

[17] R. Hui, W. Huang, Y. Zhang, M. Hameed, Miguel Razo, Marco Tacca, and A. Fumagalli, "Digital subcarrier cross-connects (DSXCs)," *Transparent Optical Networks (ICTON), 2012 14th International Conference on*, pp. 1-6. IEEE, 2012.

[18] R. Hui, W. Huang, Y. Zhang, M. Hameed, M. Razo, M. Tacca, and A. Fumagalli, “Digital subcarrier optical networks and cross-connects,” *Journal of High Speed Networks*, 19, no. 1 (2013): 55-69.

[19] Y. Zhang, M. O’Sullivan, and R. Hui, “Digital subcarrier multiplexing for flexible spectral allocation in optical transport network,” *Optics Express*, Vol. 19, No. 22, pp. 21880-21889, October, 2011.

[20] W. Huang, M. Razo, M. Tacca, A. Fumagalli, and R. Hui, “Digital subcarrier optical networks (DSONs),” *2012 14th International Conference on Transparent Optical Networks (ICTON)*, IEEE, 2012: 1-5.

[21] E. Yetginer and G. N. Rouskas. "Power efficient traffic grooming in optical WDM networks," *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. IEEE, 2009.

[22] C. Kachris, and T. Ioannis, “A survey on optical interconnects for data centers,” *IEEE Communications Surveys & Tutorials*, 14.4 (2012): 1021-1036.

[23] A. Pizzinat, P. Chanclou, F. Saliou, and T. Diallo, “Things you should know about fronthaul,” *Journal of Lightwave Technology*, 2015, 33(5): 1077-1083.

[24] X. Liu, H. Zeng, N. Chand, and F. Effenberger, “Efficient mobile fronthaul via DSP-based channel aggregation,” *Journal of Lightwave Technology*, 34, no. 6 (2016): 1556-1564.

[25] P. T. Dat, A. Kanno, and T. Kawanishi, "Radio-on-radio-over-fiber: efficient fronthauling for small cells and moving cells," *IEEE Wireless Communications*, 22.5 (2015): 67-75.

[26] L.Giorgi, G. Bruno, J. Nijhof, P. J. Urban, G. Vall-llosera, F. Ponzini and J. Ladvánszky, "Subcarrier multiplexing RF plans for analog radio over fiber in heterogeneous networks," *Journal of Lightwave Technology*, 34.16 (2016): 3859-3866

[27] X. Liu, and F. Effenberger, “Emerging optical access network technologies for 5G wireless,” *Journal of Optical Communications and Networking*, 8, no. 12 (2016): B70-B79.

[28] H. Zeng, X. Liu, S. Megeed, A. Shen, and F. Effenberger, "Digital Signal Processing for High-Speed Fiber-Wireless Convergence," *Journal of Optical Communications and Networking* 11.1 (2019): A11-A19.

[29] C. Browning, E. P. Martin, A. Farhang, and L. P. Barry, "60 GHz 5G Radio-Over-Fiber Using UF-OFDM With Optical Heterodyning," *IEEE Photonics Technology Letters* 29.23 (2017): 2059-2062.

[30] T. Xu, A. Fumagalli, and R. Hui, “Real-Time DSP-Enabled Digital Subcarrier Cross-Connect Based on Resampling Filters,” Journal of Optical Communications and Networking, 2018, 10(12): 937-946.

[31] T. Xu, and R. Hui, “Real-Time Digital Subcarrier Cross-Connect Based on Distributed Arithmetic DSP Algorithm,” *2019 IEEE Optical Interconnects Conference (OI)*, IEEE, 2019: 1-2.

[32] P. J. Winzer, “High-spectral-efficiency optical modulation formats,” *Journal of Lightwave Technology*, 2012, 30(24): 3824-3835.

[33] B. Bäuerle, A. Josten, M. Eppenberger, E. Dornbierer, D. Hillerkuss, and J. Leuthold, “FPGA-based Real-Time Receiver for Nyquist-FDM at 112 Gbit/s sampled with 32 GSa/s,” *2017 Optical Fiber Communications Conference and Exhibition (OFC)*, IEEE, 2017: 1-3.

[34] J. Armstrong, “OFDM for optical communications,” *Journal of lightwave technology*, 2009, 27(3): 189-204.

[35] Y. Benlachtar, P. M. Watts, R. Bouziane, P. Milder, D. Rangaraj, A. Cartolano, R. Koutsoyannis, J. C. Hoe, M. Püschel, M. Glick, and R. I. Killey, “Generation of optical OFDM signals using 21.4 GS/s real time digital signal processing,” *Optics Express*, 2009, 17(20): 17658-17668.

[36] Q. Yang, S. Chen, Y. Ma, and W. Shieh, “Real-time reception of multi-gigabit coherent optical OFDM signals,” *Optics express*, 2009, 17(10): 7985-7992.

[37] S. Chen, Q. Yang, Y. Ma, and W. Shieh, “Real-time multi-gigabit receiver for coherent optical MIMO-OFDM signals,” *Journal of Lightwave Technology*, 2009, 27(16): 3699-3704.

[38] N. Kaneda, Q. Yang, X. Liu, S. Chandrasekhar, W. Shieh, and Y. K. Chen, “Real-time 2.5 GS/s coherent optical receiver for 53.3-Gb/s sub-banded OFDM,” *Journal of lightwave technology*, 2009, 28(4): 494-501.

[39] R. P. Giddings, X. Q. Jin, H. H. Kee, X. L. Yang, and J. M. Tang, "First experimental demonstration of real-time optical OFDM transceivers," *2009 35th European Conference on Optical Communication*, IEEE, 2009: 1-2.

[40] R. P. Giddings, X. Q. Jin, and J. M. Tang, "Experimental demonstration of real-time 3Gb/s optical OFDM transceivers," *Optics express*, 2009, 17(19): 16654-16665.

[41] R. P. Giddings, X. Q. Jin, and J. M. Tang, "First experimental demonstration of 6Gb/s real-time optical OFDM transceivers incorporating channel estimation and variable power loading," *Optics express*, 2009, 17(22): 19727-19738.

[42] X. Q. Jin, E. Hugues-Salas, R. P. Giddings, J. L. Wei, J. Groenewald, and J. M. Tang, "First real-time experimental demonstrations of 11.25 Gb/s optical OFDMA PONs with adaptive dynamic bandwidth allocation," *Optics express*, 2011, 19(21): 20557-20570.

[43] R. P. Giddings, "Real-time digital signal processing for optical OFDM-based future optical access networks," *Journal of Lightwave Technology*, 2013, 32(4): 553-570.

[44] Xilinx, "Virtex 5 Product Table (online)", https://www.xilinx.com/support/documentation/selection-guides/virtex5-product-table.pdf

[45] Xilinx, "Virtex 7 Product Table (online)", https://www.xilinx.com/support/documentation/selection-guides/virtex7-product-table.pdf

[46] JEDEC Standard: Serial Interface for Data Converters (JESD204B Specification)

[47] T. Hill, "Comprehensive JESD204B Solution Accelerates and Simplifies Development," *White Paper: All Programmable FPGAs and SoCs* v1. 0.1) (2014).

[48] Analog Devices. "JESD204B Survival Guide." (2014). (Online) https://www.analog.com/media/en/technical-documentation/technical-articles/JESD204B-Survival-Guide.pdf

[49] P. Milder, F. Franchetti, J. C. Hoe, and M. Püschel, “Computer generation of hardware for linear digital signal processing transforms,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2012, 17(2): 15.

[50] F. J. Harris, “Multirate signal processing for communication systems,” *Prentice Hall PTR*, 2004, Chapter 2.

[51] Xilinx, “FIR compiler V7.2 logicCORE IP Product Guide (online),” https://www.xilinx.com/support/documentation/ip_documentation/fir_compiler/v7_2/pg149-fir-compiler.pdf , November 2015.

[52] F. J. Harris, C. Dick, and M. Rice, “Digital receivers and transmitters using polyphase filter banks for wireless communications,” *IEEE transactions on microwave theory and techniques*, 51.4 (2003): 1395-1412.

[53] G. Bosco, A. Carena, V. Curri, P. Poggiolini, and F. Forghieri, “Performance Limits of Nyquist-WDM and COOFDM in High-Speed PM-QPSK Systems,” *IEEE Photonics Technology Letters*, 22.15 (2010): 1129-1131.

[54] R. Schmogrow, B. Nebendahl, M. Winter, A. Josten, D. Hillerkuss, S. Koenig, J. Meyer, M. Dreschmann, M. Huebner, C. Koos, J. Becker, W. Freude, and J. Leuthold, “Error vector magnitude as a performance measure for advanced modulation formats,” *IEEE Photonics Technology Letters*, 2011, 24(1): 61-63.

[55] D. Novak, R. B. Waterhouse, A. Nirmalathas, C. Lim, P. A. Gamage, T. R. Clark, M. L. Dennis, and J. A. Nanzer, “Radio-over-fiber technologies for emerging wireless systems,” *IEEE Journal of Quantum Electronics*, 52.1 (2016): 1-11.

[56] D. A. A. Mello, A. N. Barreto, F. A. Barbosa, C. Osorio, M. Fiorani, and P. Monti, “Spectrally efficient fronthaul architectures for a cost-effective 5G C-RAN,” *Transparent Optical Networks (ICTON), 2016 18th International Conference on*, pp. 1-5, IEEE, 2016.

[57] X. Liu, H. Zeng, and F. Effenberger, “Bandwidth-efficient synchronous transmission of I/Q waveforms and control words via frequency-division multiplexing for mobile fronthaul,” *Global Communications Conference (GLOBECOM)*, IEEE, 2015.

[58] M. Mehendale, M. Sharma, and P. K. Meher, "DA-Based Circuits for Inner-Product Computation," *Arithmetic Circuits for DSP Applications* (2017).

[59] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE Assp Magazine* 6.3 (1989): 4-19.

[60] Mathworks Inc, “Distributed Arithmetic for FIR Filters”, https://www.mathworks.com/help/hdlfilter/distributed-arithmetic-for-fir-filters.html

[61] Mathworks Inc, “Filter Design HDL Coder™ User's Guide”, 2019.

[62] P. K. Meher, "LUT optimization for memory-based computation," *IEEE Transactions on Circuits and Systems II: Express Briefs* 57.4 (2010): 285-289.

[63] P. K. Meher, S. Chandrasekaran, and A. Amira. "FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic," *IEEE transactions on signal processing* 56.7 (2008): 3009-3017.

[64] S. Y. Park, and P. K. Meher, "Efficient FPGA and ASIC realizations of a DA-based reconfigurable FIR digital filter," *IEEE Transactions on Circuits and Systems II: Express Briefs* 61.7 (2014): 511-515.

[65] A. Paek, “Super Sample Rate FIR Implementation using Vivado HLS,” 2014.

[66] “3GPP specification: Requirements for further advancements for E-UTRA (LTE Advanced)”.

[67] “Base Station (BS) radio transmission and reception,” 3GPP TS 36.104, V.12.6.0, Feb. 2015.

[68] I. Kuon, and J. Rose, “Measuring the gap between FPGAs and ASICs,” IEEE Transactions on computer-aided design of integrated circuits and systems, 2007, 26(2): 203-215.

Acronyms

ADC	analog to digital converter
APC	antisymmetric product coding
ARoF	analog radio over fiber
ASIC	application-specific integrated circuit
AWG	arbitrary waveform generator
AXI	advanced extensible interface
BPF	bandpass filter
BRAM	block RAM
C-RAN	cloud radio access network
CO	central office
DA	distributed arithmetic
DAC	digital to analog converter
DCT	discrete cosine transform
DDC	digital down-conversion
DDS	direct digital synthesizer
DFT	discrete Fourier transform
DWT	discrete wavelet transform
DUC	digital up-conversion
DRoF	digital radio over fiber
DSCM	digital subcarrier multiplexing
DSP	digital signal processing
DSXC	digital subcarrier cross-connect
DTFT	discrete-time Fourier transform
DXC	digital cross-connect

EON	elastic optical networking
EVM	error vector magnitude
FDM	frequency division multiplexing
FFT	fast Fourier transform
FIFO	first in first out
FIR	finite impulse response
FMC	FPGA mezzanine card
FPGA	field programmable gate array
FS	frequency slot
GSPS	giga samples per second
HDL	hardware description language
IP	intellectual property
LPF	low-pass filter
LO	local oscillator
LUT	look-up table
LUTRAM	LUT random access memory
LSB	least significant bit
MAC	multiply-accumulate circuitry
OFDM	orthogonal frequency division multiplexing
OMS	odd-multiple-storage
OOK	on-off key
OSC	oscilloscope
OXC	optical cross-connect
QAM	quadrature amplitude modulation
QPSK	quadrature phase shift keying

PAPR	peak to average power ratio
RAM	random access memory
RF	radio frequency
ROADM	reconfigurable optical add/drop multiplexing
RRH	remote radio head
SC	subcarrier
SCM	subcarrier multiplexing
SE	spectral efficiency
SMF	single mode fiber
UART	universal asynchronous receiver-transmitter
WDM	wavelength division multiplexing

Appendix I

Design of DSXC in Xilinx Vivado

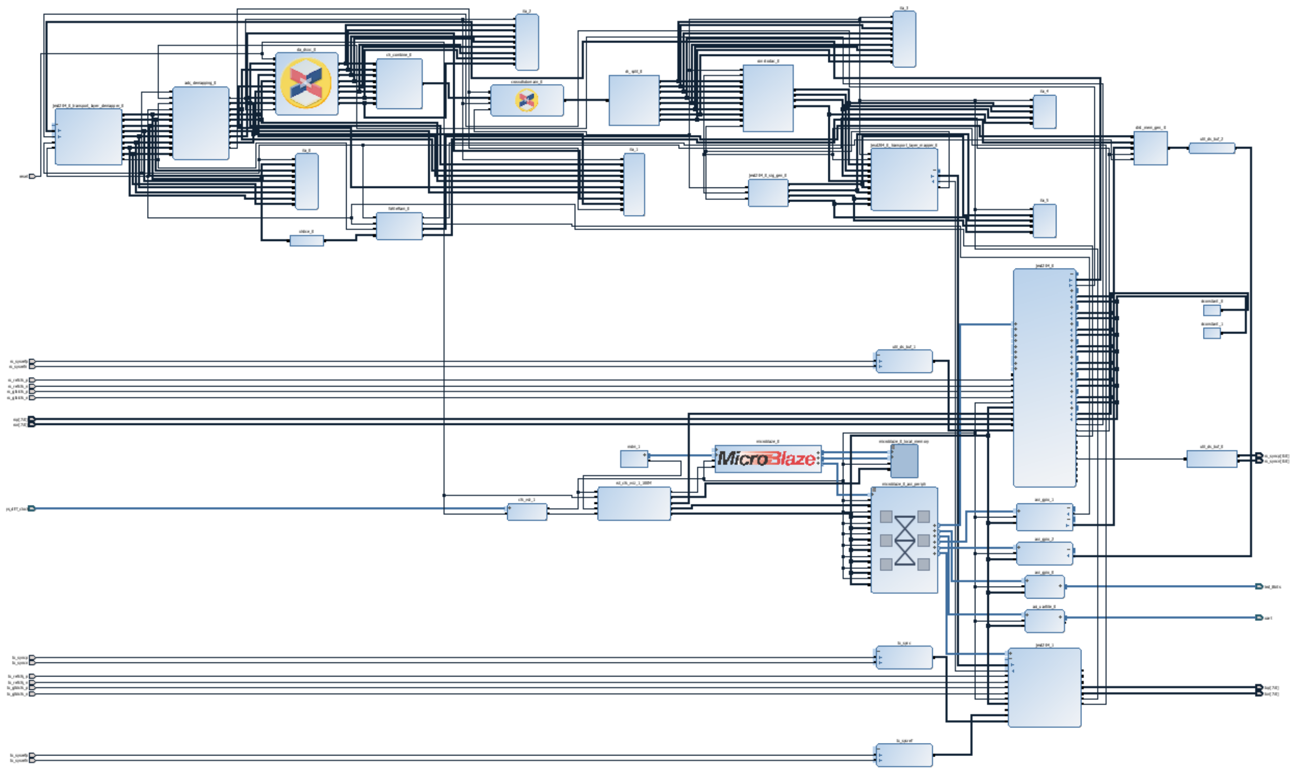


Figure 1 Block design in Xilinx Vivado

Figure 1 shows the block design in Xilinx Vivado 2015.4, each block is an intellectual property (IP). The IPs can be Vivado built-in IP, user packaged IP or thirty party IP.

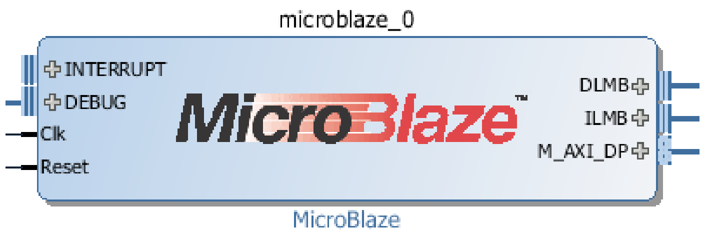


Figure 2 Microblaze IP

Figure 2 shows the Microblaze IP, which is a softcore processor that can be programmed in C language. In this design, this processor is used to configure and monitor IPs such as JESD204B IP through AXI interface. Microblaze can communicate with a computer through the UART interface, which can be used for debugging.

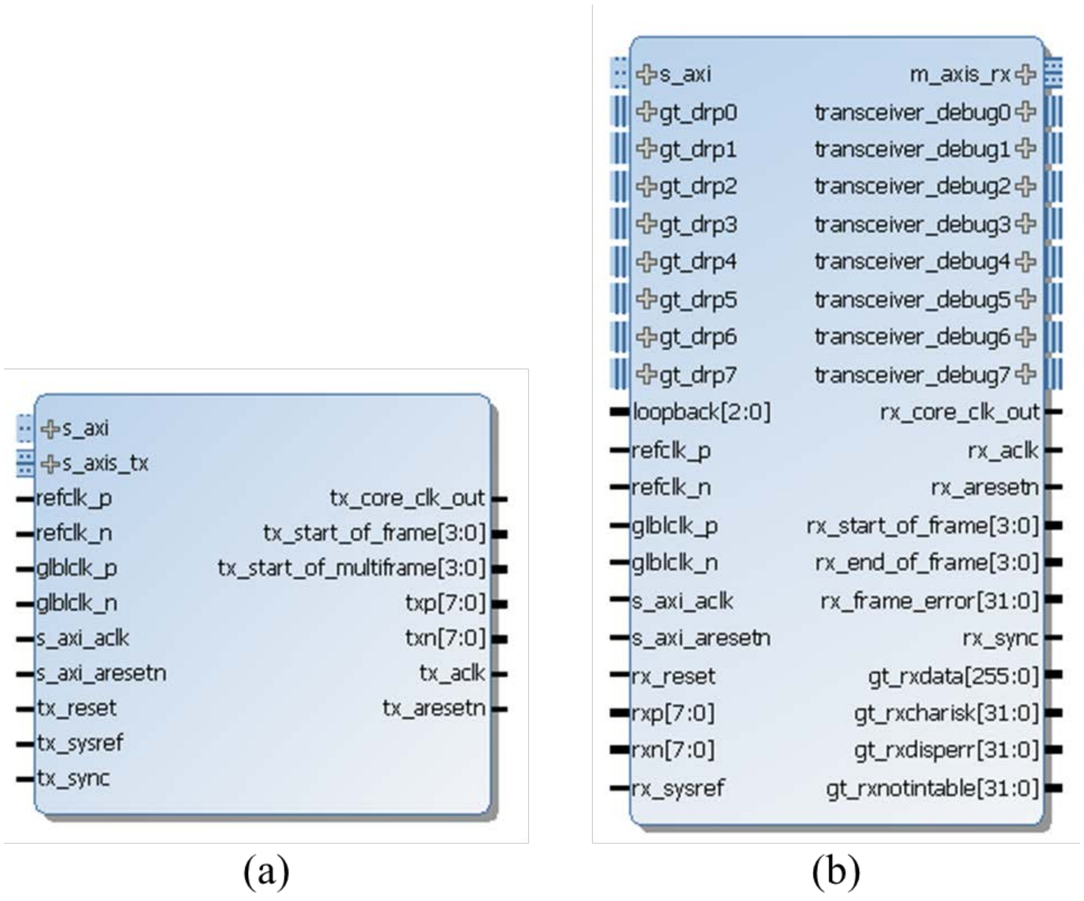
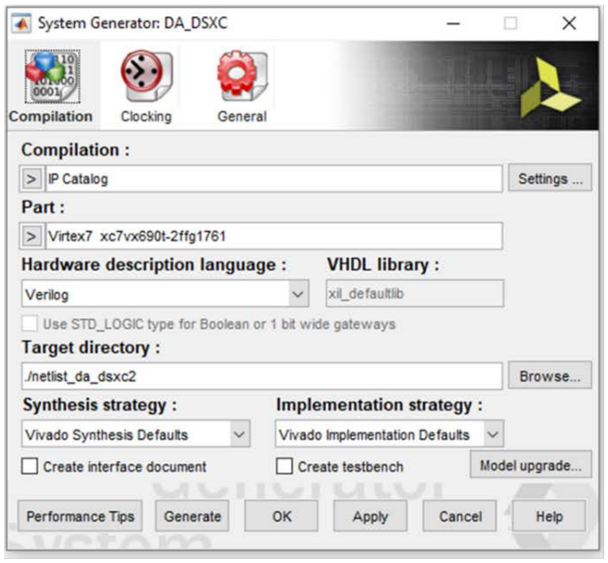


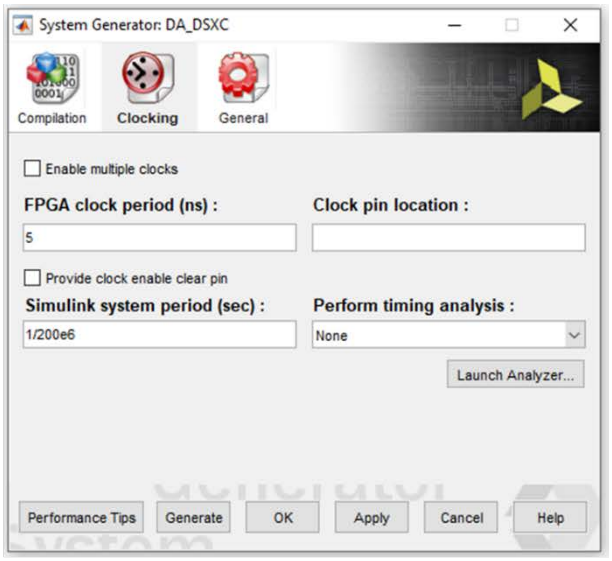
Figure 3 (a) JESD204B TX IP and (b) JESD RX IP

Figure 3 shows the blocks of JESD204B TX IP and JESD204B RX IP in the block design in Xilinx Vivado. As shown in Figure 3(a), the JESD204B TX IP has 8 differential pairs of serial lanes, which transmits data from FPGA board to DAC board. The DAC board is operating in dual-channel mode, and each DAC has an input data rate of 1.6 GSPS and 16 bits resolution. Considering the 8B/10B encoding, the line rate of this JESD204B TX IP core is 1.6GSPS × 16 bits × 10/8 × 2 × 1/8 = 8 Gbps. Figure 3(b) shows the JESD204B RX IP which receives data from an ADC board and sends to FPGA logic for processing. Since the ADC is operating in single channel mode with output data rate of 1.6 GSPS and 12 bits resolution. The ADC aggregates every 5 samples and add 4 bits overhead to form a frame of 64 bits. Considering the overhead and 8B/10B encoding, the line rate of this JESD204B RX IP core is 1.6 GSPS × 12 bits × 64/60 × 10/8 × 1/8 = 3.2 Gbps.


System Generator: DA_DSXC

Compilation :
IP Catalog
Part :
Virtex7 xc7vx690t-2ffg1761
Hardware description language : Verilog
VHDL library : xil_defaultlib
Use STD_LOGIC type for Boolean or 1 bit wide gateways
Target directory :
./netlist_da_dsxc2
Synthesis strategy : Vivado Synthesis Defaults
Implementation strategy : Vivado Implementation Defaults
Create interface document
Create testbench

(a)

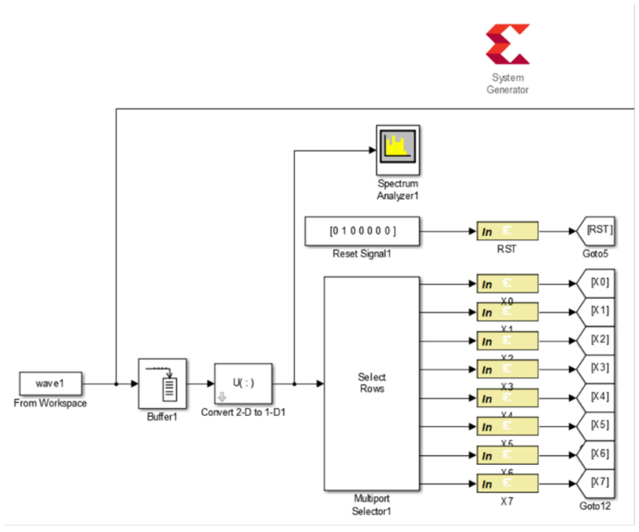

System Generator: DA_DSXC

Enable multiple clocks
FPGA clock period (ns) : 5
Clock pin location :
Provide clock enable clear pin
Simulink system period (sec) : 1/200e6
Perform timing analysis : None

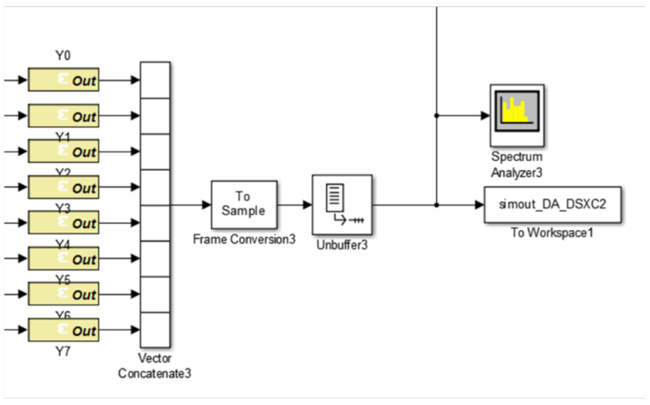
(b)

Figure 4 Configuration of (a) compilation and (b) clocking in System Generator

Figure 4 shows the configurations of compilation and clocking in Xilinx System Generator. As shown in Figure 4 (a) the target device is Virtex7 xc7vx690t-2ffg176, and the targeted hardware description language of generated IP is Verilog. As shown in Figure 4 (b), both the FPGA clock period and the Simulink system period are set to be 5 ns.



(a)



(b)

Figure 5 (a) Input data path and (b) output data path of design in System Generator

As shown in Figure 5, both the input data path and output data path consist of eight parallel data channels. Since the clock rate of FPGA is 200MHz, the overall sampling rate of eight parallel data channels is 1.6GHz. The design of DSXC between the input data path and output data path can be packaged into an IP that can be imported into the Vivado project in Figure 1.

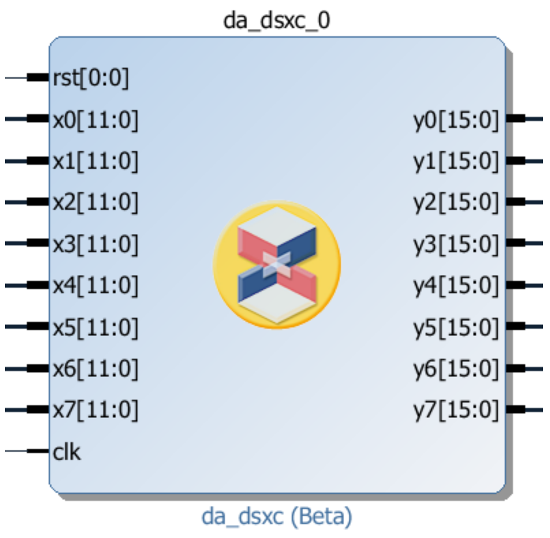
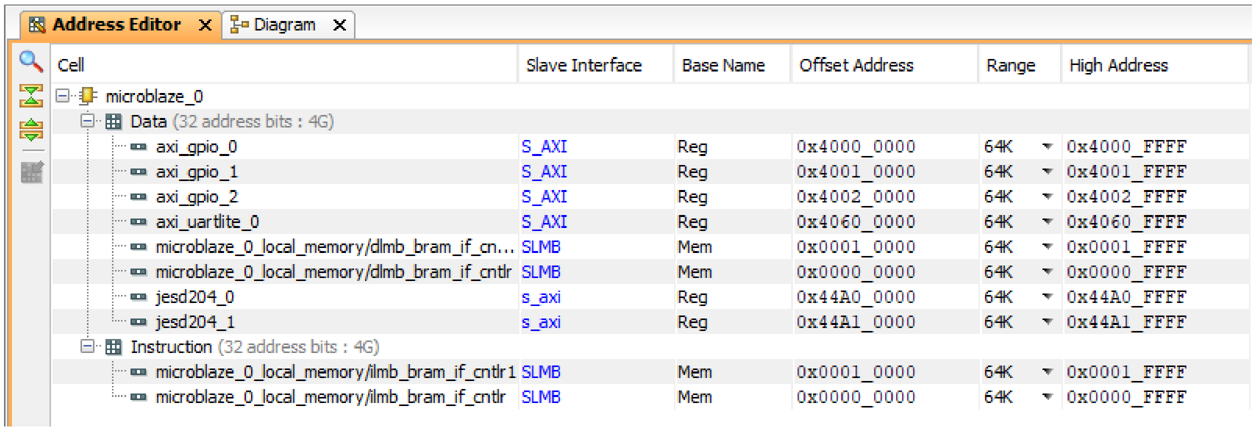


Figure 6 DSXC IP generated by System Generator

Figure 6 shows the DSXC IP that is generated and packaged in System Generator. As shown in Figure 6, the input of DSXC IP has eight parallel data channels with 12 bits width in each channel, which is equal to the bit resolution of ADC. The output of DSXC IP also has eight parallel data channels with 16 bits width, which is equal to the bit resolution of DAC.

After the design in Figure 1 is completed and bitstream is generated, the hardware, including bitstream, can be exported into the Xilinx Software development Kit (SDK) environment for software development. C programs for Microblaze can be developed in SDK.



Cell	Slave Interface	Base Name	Offset Address	Range	High Address
microblaze_0					
Data (32 address bits : 4G)					
axi_gpio_0	S_AXI	Reg	0x4000_0000	64K	0x4000_FFFF
axi_gpio_1	S_AXI	Reg	0x4001_0000	64K	0x4001_FFFF
axi_gpio_2	S_AXI	Reg	0x4002_0000	64K	0x4002_FFFF
axi_uartlite_0	S_AXI	Reg	0x4060_0000	64K	0x4060_FFFF
microblaze_0_local_memory/dlmb_bram_if_cn...	SLMB	Mem	0x0001_0000	64K	0x0001_FFFF
microblaze_0_local_memory/dlmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	64K	0x0000_FFFF
jesd204_0	s_axi	Reg	0x44A0_0000	64K	0x44A0_FFFF
jesd204_1	s_axi	Reg	0x44A1_0000	64K	0x44A1_FFFF
Instruction (32 address bits : 4G)					
microblaze_0_local_memory/ilmb_bram_if_cntlr1	SLMB	Mem	0x0001_0000	64K	0x0001_FFFF
microblaze_0_local_memory/ilmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	64K	0x0000_FFFF

Figure 7 Address Map

Figure 7 shows the address map of IPs in Vivado Design, each IP has its own unique offset address which can be used as its base address in SDK. With the knowledge of SDK and C functions, Microblaze is able to configure and monitor IPs such as JESD and UART through the AXI interface.

In conclusion, both the design and simulation of DSXC can be accomplished in Xilinx System Generator, which is a development environment similar to Simulink.