Efficient real-time digital subcarrier cross-connect (DSXC) based on distributed arithmetic DSP algorithm

Tong Xu*, Andrea Fumagalli†, and Rongqing Hui*

Abstract— Two real-time functions of digital subcarrier cross-connect (DSXC) are experimentally demonstrated for the first time using distributed arithmetic (DA) in a field programmable gate array (FPGA) platform. Both frequency translation and channel selection in DSXC are implemented using DA-based resampling filters, achieving flexible modulation format and fine data-rate granularity of many concurrent subcarrier channels. Compared with traditional resampling filters that leverage multipliers, the DA-based approach eliminates the need for DSP slices in the FPGA implementation and significantly reduces the hardware cost. By requiring only a few clock periods, the DA-based resampling filter is also significantly faster when compared to conventional FIR filters, whose overall latency is proportional to the filter order. The DA-based DSXC is therefore able to achieve improved spectral efficiency and programmability of multiple orthogonal subcarrier channels, while keeping low cross-connection latency and requiring low cost hardware resources when implemented in a FPGA platform.

Index Terms— Distributed Arithmetic, DSXC, FPGA, frequency translation, resampling filter.

I. INTRODUCTION

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 networks (EON) offer more flexibility, with channel wavelength granularity down to 12.5GHz or lower, which may yield tangible SE improvement in optical networks [1]. 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.

Subcarrier multiplexing (SCM) can provide much finer granularity by multiplexing a large number of subcarrier channels in the electrical domain [2][3]. 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. Processing high data rate signals in the digital domain has become practical and offers many advantages compared to traditional analog techniques. 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 field programmable gate array (FPGA) [4][5]. In addition to being used as a modulation format for optical signal transmission [6], DSCM can also be used to carry orthogonal channel which can be individual switched by digital subcarrier cross-connect (DSXC) devices [7][8]. 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 [9] [10].

For real-time implementation of the DSXC key functions efficient utilization of DSP resources is a major concern. While 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 previously demonstrated a real-time DSP-enabled DSXC based on resampling digital filters to achieve frequency translation and channel selection of subcarriers [11]. Although these resampling filters reduce DSP resource utilization compared to the traditional frequency translation scheme based on I/Q mixing, they still heavily rely on digital multipliers, which are implemented using numerous DSP slices in the FPGA. To overcome this drawback we demonstrate in this paper a more efficient technique to realize real-time frequency translation and channel selection of DSCM channels based on distributed arithmetic (DA). No digital multipliers are required when using DA, thus completely eliminating the need for DSP slices in the FPGA [12]. In addition, the DA-based DSXC reduces DSP-induced latency down to only a few FPGA clock periods, independent of the applied digital filter order. DA has been used to implement digital filters for Nyquist pulse generation in fiber-optic transmitter [4][5], 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. The major contribution of this paper is to report 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.

II. DIGITAL SUBCARRIER CROSS-CONNECT

Fig. 1 shows the block diagram of a generic DSXC node [11]. 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), 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 \(c_{ij}\), \((i = 1, 2, \ldots, n,\) and \(j = 1, 2, \ldots, m)\) can be routed to any output wavelength \(\lambda_i\) 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 our experiment 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.

![Fig. 1. Digital subcarrier cross-connect](image)

In order to be able to route any subcarrier channel of any input wavelength to any subcarrier frequency slot 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 is achieved through decimation filter, in which the decimation factor is the ratio of the input rate to the output rate. The frequency up conversion is 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. 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 the 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 [11]. Both decimation filter and interpolation filter can be categorized as resampling filters, which are essentially finite impulse response (FIR) filters. 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 FPGA. 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.

III. DISTRIBUTED ARITHMETIC

First proposed in the early 1970s [13], DA has been used to efficiently implement sum-of-products without using any multipliers [14][15]. The major advantage of DA is its high computational efficiency. 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 contains 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 [16]. 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 [16]. 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 [17]. FIR filter based on DA can be efficiently implemented on hardware such as FPGA or ASIC to support real-time processing [18]. DA-based reconfigurable FIR filters can also be efficiently implemented in FPGA or ASIC [19]. 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 [4]. 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 [4][5].

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 [13], and DA-architectures have been used to build traditional filters such as pulse shaping, low-pass and bandpass filters [4][5]. 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 [11]. Here we show that DA architecture can further reduce DSP resource consumption and significantly reduce DSXC latency.

In the remainder of this paper, 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 Appendix I, and the principle of polyphase decomposition to realize super-sample rate FIR filter is reviewed in Appendix II. 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.

IV. EXPERIMENT

![Fig. 2. Experimental setup](image)

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 Fig. 2. 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 16 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, DPO72304DX).

In our experiment, the DSCM signal generated by the AWG consists of eight subcarriers (SCs), and each SC carries an independent data stream with a modulation format of 16QAM 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 $A_{pass} = 0.5$dB, and a stopband attenuation $A_{stop} = 40$dB. 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.

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 Fig. 3(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 

![Fig. 3. 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](image)

Fig. 3. 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

DA-DSXC, these SC channels can be switched from any input FS to any output FS. Fig. 3(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. Figs. 3(c) and 3(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.

![Fig. 4. Signal EVM of recovered subcarriers](image)

Fig. 4. Signal EVM of recovered subcarriers

For the purpose of comparison, we also built a DSXC using resampling filters based on multipliers [11], 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 patterns 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].

Fig. 4 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.

According to [21] [22], the required EVM threshold for LTE-A is 12.5% for 16QAM. Fig. 4 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.

Fig. 5(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 $A_{\text{stop}} = 40\text{dB}$ while changing the passband ripple through the change of the filter order. Fig. 5(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 Fig. 5(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 $A_{\text{stop}}$ in the DSXC node, simulation is carried out with a fixed passband ripple of 0.5dB, and $A_{\text{stop}}$ is varied by changing the filter order. Fig. 5(b) shows the calculated EVM as a function of stopband attenuation. For $A_{\text{stop}} < 40\text{dB}$, EVM improves rapidly with the increase of $A_{\text{stop}}$ due to the significant reduction of inter-channel crosstalk. The EVM improvement saturates when $A_{\text{stop}}$ 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 Fig. 5(b). As previously mentioned, by setting $A_{\text{pass}} = 0.5\text{dB}$ and $A_{\text{stop}} = 40\text{dB}$, the order of the FIR filter is 134,
which is the value chosen in this study.

V. 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. Appendix I and II define most of the terms used in this section.

A. 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 \( W \cdot 2^{N/2} \) 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 [16]. 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 take into account 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 Appendix II. 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 Appendix II, 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 \( W \cdot 2^{N/L} \cdot G \cdot L^2/M \). 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 \( W \cdot 2^{134/16} \cdot G \cdot 8 \). 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 \(12 \times 2^{134/16} \times 12 \times 8 = 1.27 \times 10^8 \) 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 \((12 \times 2^6 + 12 \times 6^6 + 12 \times 2^5) \times 12 \times 8 = 184,320 \) 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 all 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.

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 I 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.

<table>
<thead>
<tr>
<th>FPGA Resource</th>
<th>Total available</th>
<th>DA DSXC IP</th>
<th>MULT DSXC IP</th>
<th>Post Synthesis Post-implementation</th>
<th>util</th>
<th>%</th>
<th>util</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT</td>
<td>433,200</td>
<td>111,711</td>
<td>25.79</td>
<td>29,435</td>
<td>6.79</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUTRAM</td>
<td>174,200</td>
<td>0</td>
<td>0</td>
<td>26,904</td>
<td>15.44</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF</td>
<td>866,400</td>
<td>108,412</td>
<td>12.51</td>
<td>30,923</td>
<td>3.57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRAM</td>
<td>1,470</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSP slice</td>
<td>3,600</td>
<td>0</td>
<td>0</td>
<td>2,147</td>
<td>59.64</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table II 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.

### B. Latency

Here we define the latency of an 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. Traditional 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 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 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 Fig. 6 in Appendix II, 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 \( \mu \text{s} \) 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. So the total filter-induced latency of MULT DSXC is approximately 0.67 \( \mu \text{s} \). 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 in Fig. 1 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 \( \mu \text{s} \)) 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 \( \mu \text{s} \), 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 \mu s and 1.96 \mu s, 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 \mu s while the latency introduced by DSXC IP of DA DSXC is 0.14 \mu s, 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.

C. 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 [20], 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 [20], the area required to implement LUT in ASICs is 35 times smaller than that in FPGAs, while the area required to implement multipliers in ASICs is 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.

VI. 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.

APPENDIX I

The DA principles are discussed in [14][18]. For the reader’s convenience, we briefly introduce 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 (1) shows the inner-product computation of A and B, which can be obtained using DA as described next.

\[
C = \sum_{k=0}^{N-1} A_k \cdot B_k
\]

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 (2).

\[
B_k = -b_{k0} + \sum_{r=1}^{R-1} b_{kr} \cdot 2^{-r}
\]

Substituting Equation (2) into Equation (1), the inner-product of A and B can be expanded as in Equation (3).

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

Taking the 2^-r component out of the bracket in (3), we get

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

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. So equation (4) can be re-written as in (5), where the expression of C_r is shown in (6).

\[
C = \sum_{r=1}^{R-1} 2^{-r} \cdot C_r
\]

\[
C_r = \sum_{k=0}^{N-1} A_k \cdot b_{kr}
\]

As shown in (6), every C_r of r = 0,1,...,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,...,N-1\} and its input vector is \{s_n(k), k = 0,1,...,N-1\}, then the output of this FIR filter can be given by (7).

\[
y(n) = \sum_{k=0}^{N-1} h(k) \cdot s_n(k)
\]

Here we assume the input sample of the filter is x(n), and s_n(k) = x(n-k). Comparing with (6), (7) can be rewritten as

\[
y(n) = \sum_{r=0}^{R-1} 2^{-r} \cdot C_r
\]

where

\[
C_r = \sum_{k=0}^{N-1} h(k) \cdot (s_n(k))_r
\]

with (s_n(k))_r being the rth bit of s_n(k).
Equation (8) and (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 $m = 0, 1, ..., M - 1$ and $p = 0, 1, ..., P - 1$. In this case, Equation (8) can be rewritten as

$$y(n) = \sum_{r=0}^{P-1} 2^{-r} \cdot \left( \sum_{p=0}^{M-1} (S_n)_{r,p} \right)$$  \hspace{1cm} (10)

where

$$(S_n)_{r,p} = \sum_{s=0}^{M-1} h(m + pM) \cdot (s_n(m + pM))$$ \hspace{1cm} (11)

for $r = 0, 1, ..., R - 1$ and $p = 0, 1, ..., P - 1$.

Since each $(S_n)_{r,p}$ has $2^M$ possible values this approach requires $P$ relatively small LUTs. Equation (10) can be rewritten using the memory-read operation of LUT as

$$y(n) = \sum_{r=0}^{P-1} 2^{-r} \cdot \left( \sum_{p=0}^{M-1} F(b_n)_{r,p} \right)$$ \hspace{1cm} (12)

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

APPENDIX II

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 [23]. Here we briefly introduce 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 (13).

$$y_k = \sum_{i=0}^{N-1} x_{k-i} * h_i$$ \hspace{1cm} (13)

where $h = [h_0, h_1, h_2, ..., h_{N-1}]$ 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 (14).

$$y_0 = x_0 * h_0 + x_1 * h_1 + x_2 * h_2 + x_3 * h_3 + x_4 * h_4 + x_5 * h_5 + x_6 * h_6 + x_7 * h_7 + ...$$

$$y_1 = x_0 * h_0 + x_1 * h_1 + x_2 * h_2 + x_3 * h_3 + x_4 * h_4 + x_5 * h_5 + x_6 * h_6 + x_7 * h_7 + ...$$

$$y_2 = x_0 * h_0 + x_1 * h_1 + x_2 * h_2 + x_3 * h_3 + x_4 * h_4 + x_5 * h_5 + x_6 * h_6 + x_7 * h_7 + ...$$

$$y_3 = x_0 * h_0 + x_1 * h_1 + x_2 * h_2 + x_3 * h_3 + x_4 * h_4 + x_5 * h_5 + x_6 * h_6 + x_7 * h_7 + ...$$

According to (15), the coefficients of the original FIR filter can be polyphase decomposed into four sub-filters, whose coefficients are in (16).

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

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

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

$$H_3 = [h_3, h_7, h_{11}, h_{15}, ...]$$ \hspace{1cm} (16)

As shown in (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 (15) and (16), the block diagram of this super sample rate FIR filter with a degree of parallelism of 4 is depicted in Fig. (6). As shown in Fig. 6, 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 [11], 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 Fig. 6. For an interpolation FIR filter with the same coefficients as the FIR filter in Fig. 6, its block diagram can be modified as in Fig. 7.
As shown in Fig. 7, 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 Fig. 6. Similarly, for a decimation FIR filter with the same coefficients as the FIR filter in Fig. 6, its block diagram can be modified into the block diagram of a decimation FIR filter as in Fig. 7.

As shown in Fig. 8, 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 Fig. 8, 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.

REFERENCES


[21] “3GPP specification: Requirements for further advancements for E-UTRA (LTE Advanced)”.
