A Generic Reconfigurable Mixed Time and Frequency Domain QAM Transmitter with Forward Error Correction

Shalina Percy Delicia Figuli, Peter Figuli, Alberto Sonnino, and Juergen Becker

***Abstract*—In the past three decades, Field Programmable Gate Arrays (FPGAs) have emerged to be the backbone of digital signal processing, especially in high-speed communication systems. However, today, these devices are clocked below 1 GHz and improvement in performance stays a big challenge on all abstraction layers, right from system architecture down to physical technology. Far and wide, myriad number of researches are done on methodologies and techniques which can deliver higher throughput with lower operating frequencies. Towards this projected objective, in this paper an efficient modulation technique like Quadrature Amplitude Modulation (QAM) along with mixed time and frequency domain approach and Forward Error Correction (FEC) technique have been utilized to employ a generic scalable FPGA based QAM transmitter with filter parallelization being executed in mixed domain. The system developed in this paper achieves an effective throughput of 12.8 Gb/s for 256-QAM with 16 parallel inputs having an operating frequency of 201.25 MHz, while a 18.7 Gb/s effective throughput is realized with 32 parallel inputs at 146 MHz. Thereby, it paves down a promising methodology for applications where having higher clock frequencies is a hard limit.**

***Keywords*—FPGA, mixed domain, parallelization, QAM, SRRC filter.**

I. Introduction

Field Programmable Gate Arrays (FPGAs), due to their extensive reconfigurability and flexibility, play a constantly increasing role in digital communication technology. Indeed, growth of these systems not only claim for high speed hardware but also for a flexible, low-cost and standardized environment where facile modulation techniques like Quadrature Amplitude Modulation (QAM) can be adopted and fostered. As they offer possibilities to easily test, modify and update the entire design and implementation, optimizing the system design is no longer a disputed issue. In order to transmit QAM signals, the crucial setting is to band limit transmitted signals and to suppress InterSymbol Interferences (ISIs). In that purpose, Square Root Raised Cosine (SRRC) filter is one of the most frequently used pulse shaping Finite Impulse Response (FIR) filters in digital modulation. Thanks to its matched filtering property and fulfillment of Nyquist criteria. Designing such a filter is a demanding task as the design parameters are in contradiction with each other. Optimizing and discussing the nature of the filter and choice of its parameters are left to related works [1], [2].

The first part of the paper deals with parallelizing these modern filters which is a challenging task as their convolution form is a significant speed limitation in digital communication systems. This is invalidated by the fact that linear operations performed in one domain have corresponding operations in another domain. Therefore, convolution operation in time domain becomes a pointwise multiplication in frequency domain. The second part focuses on developing a variable FPGA based generic QAM transmitter where the user through core parametrization can choose from different modulation orders, filter coefficients (filter order) and degree of parallelization (number of parallel inputs). Finally, in order to cut down the Signal-to-Noise Ratio (SNR) requirements, Forward Error Correction (FEC) technique using Convolution Encoder is enforced. Though data redundancy benefits higher order modulations in noisy channels, the introduced overhead will impair performance gain. Therefore, investigations have been made to find out bottlenecks in performance and resource utilization of various combinations of modulation orders, filter length, number of parallel inputs and FEC code rates to determine optimal solutions, whereby FEC overhead will not overrule the performance gain of the overall system.

In order to highlight today's top technology, a qualitative chart as shown in Fig.1 analyzes other works related to this field. In 2003, Yongbin Wu and Yousef R. developed a high-speed 64-QAM transceiver with filter selection as same as the one considered in this paper and reached an operating frequency of 55 MHz [3] on Xilinx Virtex - 2 FPGA. A complete 16-QAM system with an achievable frequency of 111.11 MHz was built using two Xilinx Virtex - 4 FPGA boards for transmitter and receiver respectively in 2010 ([4]). The very same year, in [5], a 64-QAM receiver based on Xilinx Virtex -5 FPGA operating at a maximum frequency of 125 MHz was developed. In 2012, a modular QAM transmitter working with 16-QAM to 256-QAM formats with an operating frequency of 128.6 MHz has been implemented on Xilinx Virtex - 4 FPGA platform [6]. The next year, a 16-QAM transceiver on Xilinx Virtex - 6 board with an achievable frequency of 625 MHz at the cost of low precision has been brought forth [7]. More recent state-of-the-art took advantage of the powerful Xilinx Virtex - 6 FPGA by building a 256-QAM transceiver at 750 MHz. Nevertheless, this impressive result is attenuated by the fact that their system doesn't comprise a filter [8]. Though much better and higher performances could have been

Manuscript received November 2, 2016; revised July 13, 2017.

S. Figuli, P. Figuli and J. Becker are with the Institute for Information Processing Technologies, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. E-mail:{shalina.ford, peter.figuli, becker}@kit.edu

A. Sonnino is with Department of Computer Sciences, University College London (UCL), London, UK. E-mail: alberto.sonnino.15@ucl.ac.uk

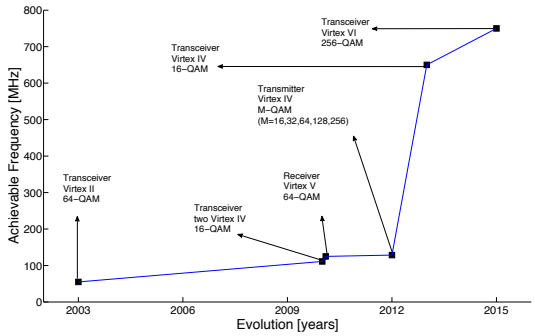


Fig. 1. Today's state-of-the-art

achieved by ASIC platforms and multi-FPGA systems, our work is confined to single channel FPGA systems.

This paper is an extended version of the work published in [9]. Consequently, the contributions of this work are:

- Extended modulation order upto 256-QAM
- Reduction of SNR requirement by inclusion of FEC technique.
- Pipelined mixed domain approach.
- Improved performance of almost 60% compared to [9].
- Flexible and scalable parallel system architecture.

The rest of the paper is organized as follows: Section II briefs the fundamental aspects of the employed mixed domain QAM transmitter. It's implementation is explained in Section III with experimental results substantiated quantitatively in Section IV, and conclusions summarized in Section V.

II. Fundamentals and Concept of Mixed Domain QAM Transmitter

Due to the convolution nature of filtering process, the filter input needs to be fed sequentially which detains the potency of the system when the preference shifts to system parallelization. Therefore, a mixed domain approach with filter operation being shifted to frequency domain to accompany parallel inputs and outputs as shown in Fig. 2 is utilized and the intermediate steps are explained in the following subsections.

A. Forward Error Correction

Increasing the modulation order leads to faster data rates and higher levels of spectral efficiency at the price of lower resistance to noise and higher InterSymbol Interference (ISI). This is due to crowding of constellation points with increase in modulation order and therefore, the receiver fails to distinguish them appropriately during reception. The receiver should also

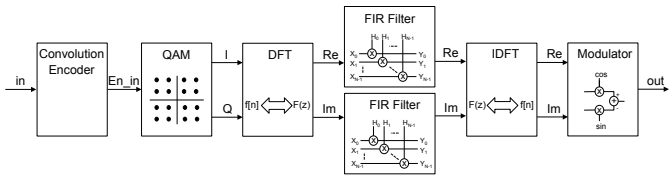


Fig. 2. Mixed-domain QAM transmitter

tolerate corrupted data bits that get flipped due to noises in the channel. Hence, incorporating FEC technique using Convolutional codes can ensure reliable communication, especially in Additive White Gaussian Noise (AWGN) channels [10], as the encoded data aids in recovering the original data at the time of error detection.

B. QAM Mapper

The encoded bit stream is clustered into k=$\log_2$(M) bits, where M is the modulation order and these k-tuples called symbols can be effectively represented using a constellation diagram. The standard rectangular constellation is preferred because of its less overhead implementation and simplicity. Although there are many ways to associate a symbol, Gray code [11] is chosen as it reduces erroneous symbol decision to one bit error. QAM symbols are then interleaved to form In-phase (I) and Quadrature (Q) components. After normalization they are sent as inputs to the filter block.

C. Fourier Transform

Fourier Transform (FT) is a mathematical tool that decompose a signal into its sinusoidal components and Inverse FT (IFT) reverses it. More specifically, Discrete Fourier Transform (DFT) takes the interleaved components from QAM mapper in time domain and transforms them into corresponding frequency domain components to be used by SRRC filter. After the filtering operation, the components are taken back in time domain by Inverse Discrete Fourier Transform (IDFT).

D. Square Root Raised Cosine Filter

Modern communication systems not only require higher data rates but also reduced ISI. Therefore, filtering becomes a delicate operation and the reason for having a mixed time and frequency domain approach is cardinal. FIR filters, though their computational requirements are more than that of an Infinite Impulse Response (IIR) filter, are chosen for the following advantages: 1) Since they don't have feedback, the total error doesn't sum up over each cycle as they have the same rounding error in each iteration. 2) They ensure good stability as the output is the sum of finite number of finite multiples of the input and cannot become greater than a fixed multiple of the input value. 3) Their linear phase property delays only the input signal and does not distort the phase. The call for an efficient spectrum usage and less ISI projects SRRC filter as one of the promising filters because of their matched filtering and fulfillment of Nyquist criteria [12]. As said above, the barrier in having parallel filters is eliminated by performing the filter operation in frequency domain where the convolution operation becomes a simple pointwise multiplication as described by equations (2) and (1).

$$y[n] = x[n] \otimes h[n] \tag{1}$$

$$Y[k] = X[k] \cdot H[k] \tag{2}$$

where X, Y and H are Fourier transforms of input, output and filter's impulse response respectively.

E. Quadrature Amplitude Modulator

The main functions of QAM modulator are to group the incoming input bit stream into symbols as per the modulation order, map them onto the signal constellation, filter the interleaved I and Q components and then modulate them with two orthogonal carriers. The former operations are done by QAM mapper and SRRC filter. IDFT is performed on the filtered real (I) and imaginary (Q) components which are then multiplied with the carrier waves. The products are then subtracted from each other to deliver the resulting modulated QAM signal which will be transmitted through the channel.

III. Implementation of QAM Transmitter

Handwritten Verilog codes, Xilinx IP cores and auto-generated Verilog modules (Java *FQM Utility*) are some of the design mechanisms that aid in the implementation of QAM transmitter. A parallel bus packing technique is used where all the parallel inputs and outputs are packed onto the same bus as shown in Fig. 3 with each $data_i$ being a 16-bit vector. The QAM transmitter accepts an arbitrary number of parallel inputs, different FEC code rates customizable filter orders and supports multiple modulation formats. Although the system has been formulated to achieve the highest order of modularity, Xilinx IP cores require the bus width parameter to be entered manually in the graphic interface at the time of core configuration. Fig. 4 depicts the implementation of the whole system with **N = 16** parallel inputs. The dotted lines represent that each main module is completely isolated from others and has its own parameterizable interface and thereby, the system designers can reuse any of these modules in their custom designs without any need for reimplementation.

A. QAM Transmitter

The design's top entity called "*transmitter.v*" comprises of three input parameters **N**, **CODE** and **FORMAT**, which represent number of parallel inputs, data redundancy (i.e., number of encoded bits) and desired modulation format respectively. Table I shows an abstract view of the system focusing on its input and output ports. The input stream is encoded and clustered into symbols defined by Verilog parameters **CODE** and **FORMAT** respectively. For example, for a 16-QAM modulation, the user needs to enter **N** parallel inputs of length specified by **CODE**, which after encoding gets packed into a single bus of width **(FORMAT*N)**, where **FORMAT** = $\log_2(16)$ = **4**. The modulated signal (**out**) is also delivered in this packed representation as **tvalid** flag validates the output data. The output width is **(16*N)** since the signal precision is fixed to 16 bits. A Java application program called *Fourier QAM Modulator (FQM) Utility* as shown in Fig. 5 has been developed to aid in breaking the complexity of design usage. By default, fifteen rows are available to enter filter

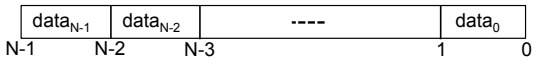


Fig. 3. Parallel bus packing

TABLE I
Transmitter - Specifications

Parameters	**N**	Number of parallel inputs
	CODE	Number of encoded bits
	FORMAT	QAM order
Inputs	*clk*	Clock
	reset	Reset
	in	Clustered input stream
Outputs	*tvalid*	Output's valid flag
	out	Output
IP Cores	*Adder Subtracter v12.0*	$4\mathbf{N}^2 - 2\mathbf{N}$
	Multiplier v12.0	$4\mathbf{N}$
	Complex Multiplier v6.0	$2\mathbf{N}^2$
	Convolution Encoder v9.0	$(\mathbf{N}*\mathbf{FORMAT})/\mathbf{CODE}$

coefficients, while extra rows can be added or suppressed depending upon the desired filter order. Only the filled rows are considered valid. Once the carrier frequency (in Hz) has been entered, the information field on the bottom right of the interface updates the entered parameters in real time and the Verilog files *filter_coeff.v*, *dft_coeff.v* and *carriers.v* which are essential to run the top module *transmitter.v* get generated. Additionally, explicit information like system precision in bits, number of entered filter coefficients as well as the number of zeros that will be padded in order to reach **N** and also the DFT size [1] can be gathered from the utility.

B. Convolutional Encoder

The incoming input bits are encoded by either of the code rates 1/2 or 1/3 having constraint lengths of 7 and 4 with polynomials [171 133] and [1 15 17] respectively ([1]). The constraint length gives the code its unique error protection quality, whereas the polynomial gives the relation between inputs and outputs. The code rate specifies the added redundancy. The encoded parallel bit stream is then sent to QAM mapper to be mapped onto the $2^{\mathbf{FORMAT}}$ points rectangular constellation. The only core used in this module is $(\mathbf{N}*\mathbf{FORMAT})/\mathbf{CODE}$ instances of *Convolution Encoder v9.0*.

C. QAM Mapper

The QAM mapper receives **N** encoded, clustered inputs and delivers **N** corresponding I and Q signals using LUT approach. This module has been set up using three Verilog parameters **N**, **W** and **FORMAT**. Respectively, they indicate number of parallel inputs, bus width and modulation format. Though the bus width (**W**) is set to 16 bits, it is still regarded as a modular parameter for future reutilization of the block. The available modulation formats are 16 (default), 32, 64, 128 and 256-QAM. Since flexibility is also a salient standard of this QAM transmitter, each modulation format has been implemented in a separate Verilog file *qam*<**FORMAT**>*.v* in order for the users to extend this system to higher modulation orders. As only the modulation order specified by **FORMAT** parameter gets generated, changing the order during execution time is not possible. Moreover, generating all the modulation formats even

[1]Even if the DFT algorithm does not require the number of inputs to be a power of two, the FFT does. This design constraint has been added for a further replacement of DFT by FFT.

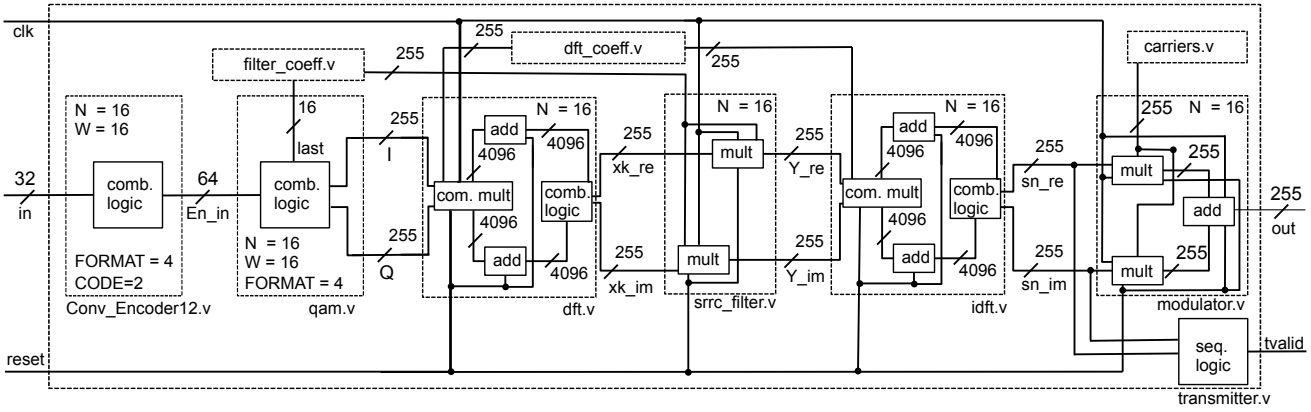


Fig. 4. Parallelized QAM system with 16*2 bits parallel inputs and 16*16 bits parallel outputs

if they are not used shouldn't actually require much additional logic as these blocks are mainly combinatorial. Nevertheless, this design constraint has been set up to keep control over resource utilization and to keep the transmitter's performance as close as possible to the results shown in section IV.

D. Discrete Fourier Transform

The DFT block (resp. IDFT block) receives signal in time domain (resp. frequency domain) and outputs its corresponding transformation in frequency domain (resp. time domain). This module receives **N** as input, which can be viewed either as transform length or number of parallel inputs. In addition to clock and reset signals, the time domain (resp. frequency domain) complex input has to be separated into its real (**xn_re**) and imaginary (**xn_im**) parts. Moreover, the weights of cosine and sine signals (**ccos** and **csin**) also have to be provided to be applied during transformation process. Unfortunately, the current FFT Xilinx IP core cannot be used as it process data serially. Therefore, in order to achieve the aimed parallelization, DFT and IDFT have been implemented in hardware. The *Complex Multiplier v6.0* core is used $\mathbf{N}^2$ times, while the *Adder Subtracter v12.0* core is used $2\mathbf{N}(\mathbf{N}-1)$ in *adder* mode, both working with 16-bit inputs. This fair amount of core's instance is required as both imaginary and real parts have to be processed. In order to achieve maximum throughput, in contrast to [9], all stages of the parallel DFT (resp. IDFT) are pipelined. This builds up additional latency but increases the throughput significantly (Refer: IV-C). Finally, the output is rescaled by 2^{-15} to avoid possible overflow.

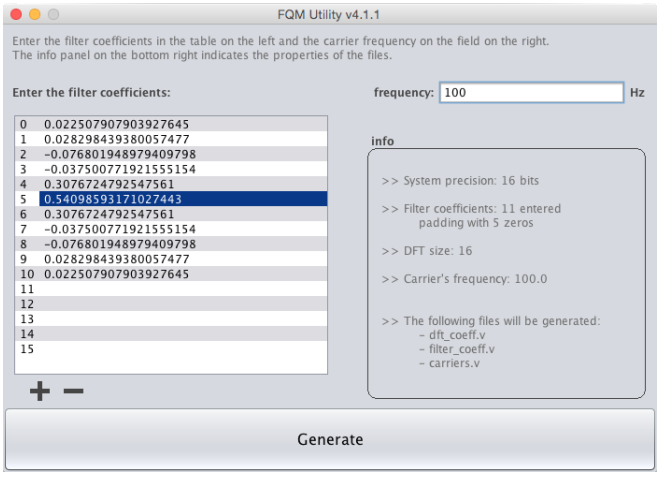


Fig. 5. Fourier QAM Modulation (FQM) Utility

E. Filter

Filter's implementation is one of the main optimization goals of this paper. Implementing the filter in frequency domain is much simpler than in time domain as it requires simple multiplication of inputs with coefficients, and most importantly, it is easily parallelizable. In order to isolate this block and to ensure its re-usability, the parameter **N** sets up the number of parallel inputs and the module accepts an arbitrary number of filter coefficients through auto-generated configuration file *filter_coeff.v*, thereby allows the system to implement a filter of arbitrary order. The only core in this block is $2\mathbf{N}$ instances of *Multiplier v12.0*, which works with 16-bit inputs, generates 16-bit symmetrically rounded outputs and rescales the output by 2^{-15}.

F. Modulator

This block, parameterized by **N**, works with any carrier frequency, which has been set up in *FQM Utility* and passed on to this module through configuration file *carriers.v*. The real input gets multiplied with cosine carrier and the imaginary with sine carrier, which are generated by a Direct Digital Synthesizer (DDS) with the help of an internal counter. These products are then subtracted according to the following equation:

$$\begin{aligned} out(t) &= \mathcal{R}\left\{[I(t)+iQ(t)]e^{2\pi f_0 t}\right\} \\ &= I(t)\cos(2\pi f_0 t) - Q(t)\sin(2\pi f_0 t) \end{aligned} \quad (3)$$

Multiplier v12.0 and *Adder Subtracter v12.0* are the two cores that have been instantiated in this module with the multiplier's configuration exactly the same as that of in the filter module and the *Adder Subtracter* core as subtracter. Since synchronization is no more an issue, the core latency is automatically set to two clock cycles in the pursuit of

performance optimization. This implementation requires 2**N** *Multiplier* cores and **N** *Adder Subtracter* cores with the *Multiplier* core rescaling the output by 2^{-15}.

G. SNR Requirements

The theoretical SNR of QAM transmitter is calculated using the following equation:

$$SNR = EbNo + 10 * \log_{10}(\textbf{FORMAT}) - 10 * \log_{10}(OSF) - 10 * \log_{10}(1/CR) \quad (4)$$

where EbNo is energy per bit to noise spectral density, OSF is over sampling factor and CR is code rate of the encoder. The EbNo ratio, which changes for different QAM orders in order to have a tolerable Bit Error Rate (BER) of 10^{-3} and the OSF parameters are taken from [1].

IV. Experimental Results

In addition to hardware implementation on Xilinx Virtex-7 VC707 evaluation board and on Xilinx Virtex Ultrascale+ device, a complete MATLAB model as shown in Fig. 6 has been developed to pre-evaluate the expected behavior of the system and to serve as a reference for the implemented system. Therefore, each block constituting the system has been first realized in MATLAB using physical and mathematical fundamentals explained before and then the system's performance and resource requirements are investigated.

A. Design Precision

For a set of 16 parallel random inputs, the transmitter's output data is compared with that of the reference MATLAB model (see Fig. 7(a)) in order to evaluate the system's precision with the assumption that the MATLAB simulation is perfect (i.e., all the internal MATLAB rounding errors are ignored). Both the results seem to overlap each other and only one curve is visible due to their proximity and Fig. 7(b) plots this error as an absolute value.

From these figures, it can be observed that the implemented system has less than 1% error with respect to the MATLAB model. For completeness, different sets of random input samples have been tested and the precision still appears to be very similar. Similarly, the outputs of DFT and IDFT blocks

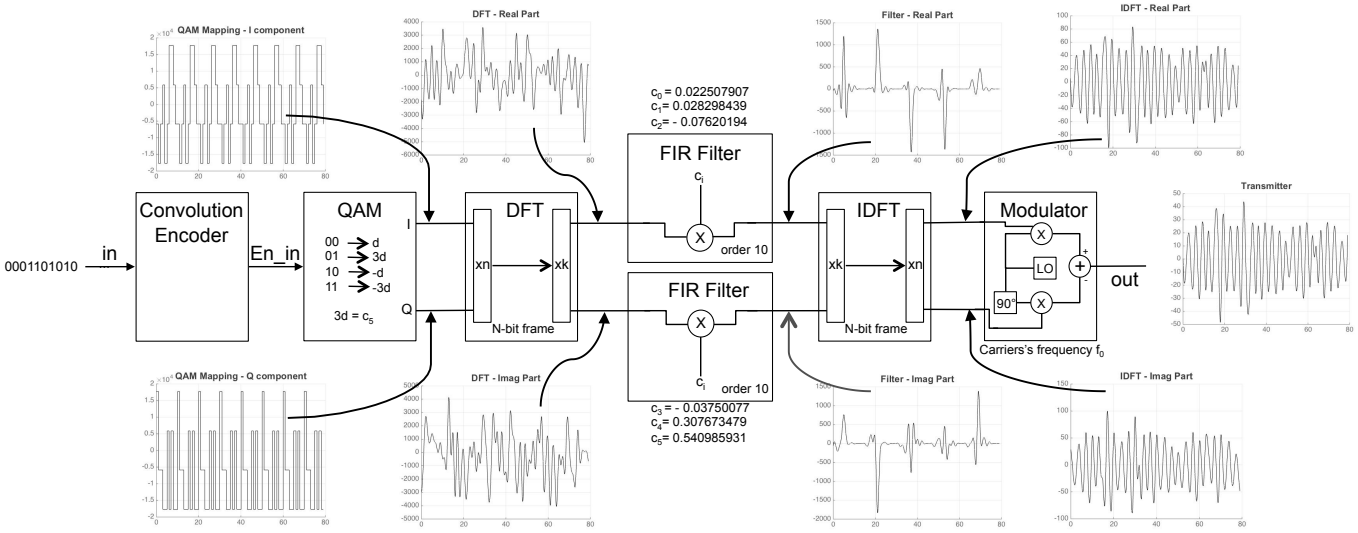
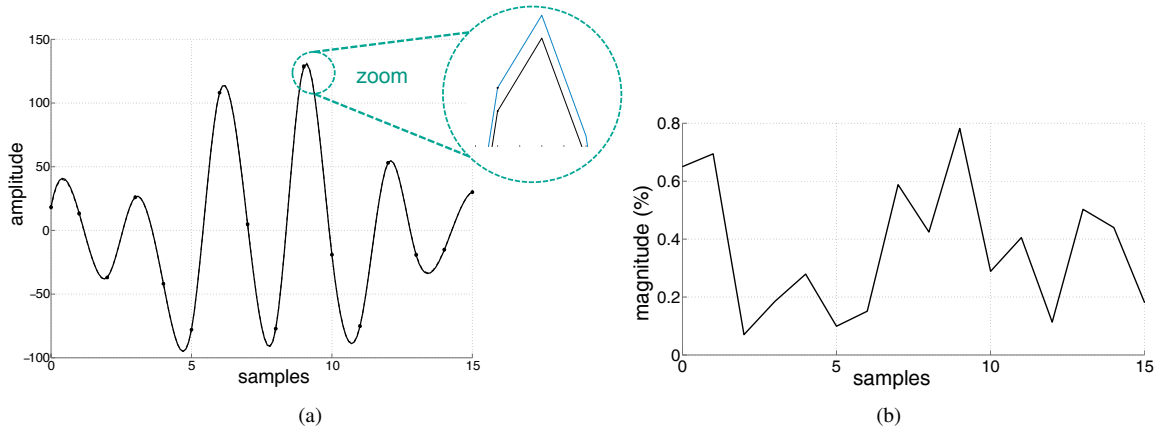
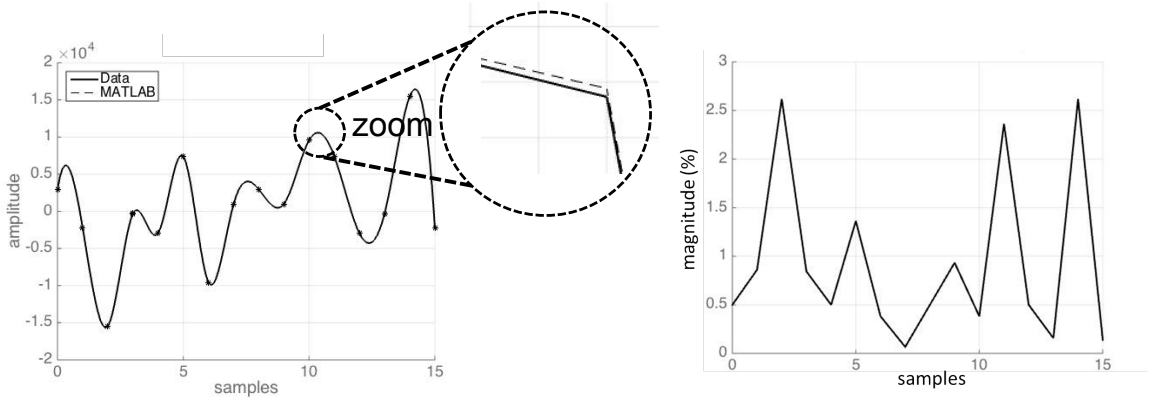


Fig. 6. MATLAB reference model of the QAM transmitter



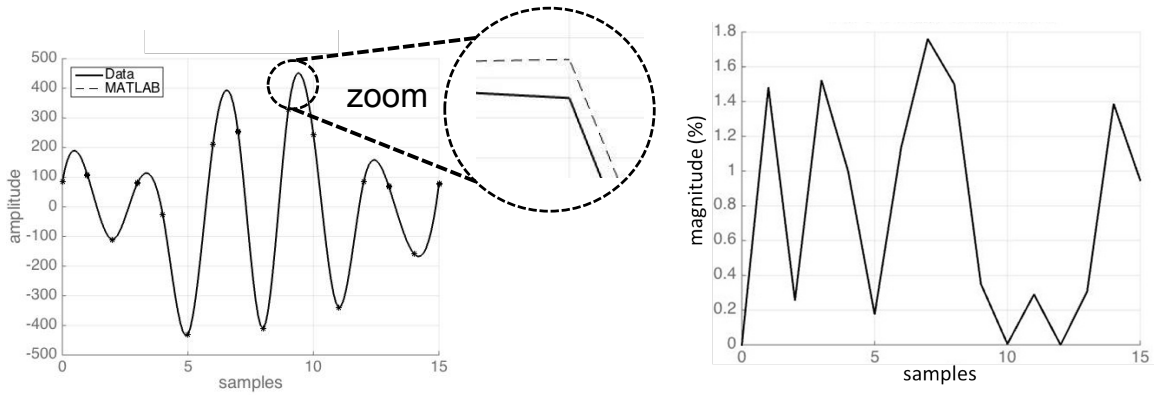
(a) (b)

Fig. 7. (a) Transmitter's precision comparison, (b) Transmitter's error



(a) (b)

Fig. 8. (a) DFT's real part, (b) DFT's real part error



(a) (b)

Fig. 9. (a) IDFT's real part, (b) IDFT's real part error

are examined by comparing their real and imaginary values with that of MATLAB simulated values as shown in Figs. 8(a), 8(b), 9(a), and 9(b). It can be seen that most of the errors in the transmitter system are from DFT and IDFT processes. Additional analysis showed that the errors can be further reduced by preferring the choice of random rounding.

B. Design Resources and Performances without FEC

All simulations in this section have been done on Xilinx Virtex - 7 with a carrier frequency of 100 Hz, and by setting the parameter **N** to 16 with asynchronous DFT and IDFT. Initially, adders and multipliers are configured to use DSPs and Mults respectively and their resource requirements and achievable performance are investigated. From Fig. 10 and 1st column of Table II, it is clear that selecting DSP option for adder configuration is not optimal as the total usage of DSP48E1 blocks is 92% and the maximum achievable frequency is 28.57 MHz. Though routing such a huge amount of DSP blocks requires much effort, simulations have been done for all QAM formats and the results stay identical due to their combinatorial nature. So the adders have been configured using fabric rather than DSPs.

To scrutinize the optimization process, a combination of fabric and LUTs for adders and multipliers respectively have been analyzed. Fig. 11 and 2nd column of Table II reveal that the system performance is improved by a factor of 3 with a clock frequency of 58.82 MHz and DSP utilization is reduced by 50%. When fabric for adders and Mults for multipliers are exercised, a slight increase of 62.5 MHz is obtained (Fig. 12). From the information displayed in the 3rd column of Table II, it can be seen that the most demanded resources are DSP48E1s while more than 50% of LUTs

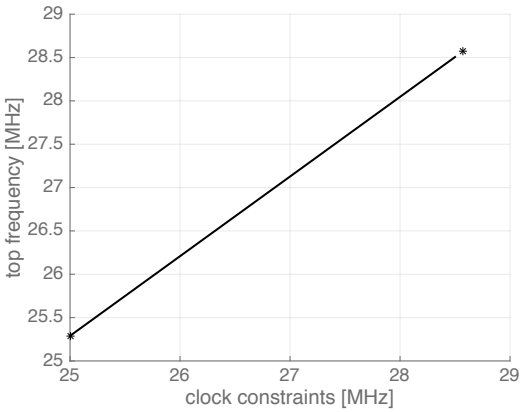


Fig. 10. DSP - Mults performance

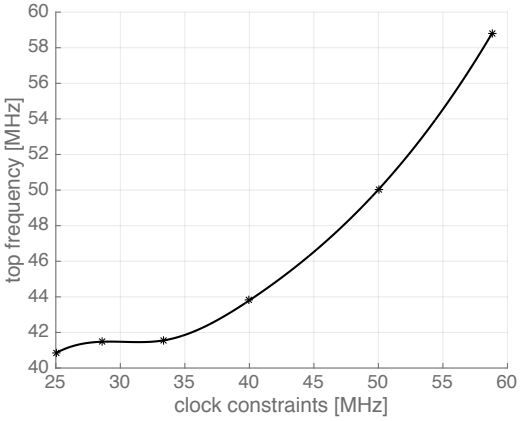


Fig. 11. Fabric - LUTs performance

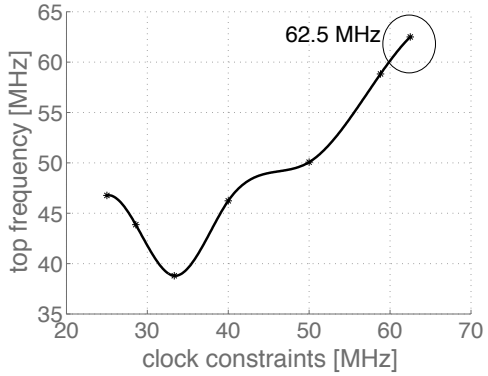


Fig. 12. Fabric - Mults performance

remain unused. Nevertheless, since the system receives **N = 16** parallel inputs, the effective speed is: $16 * 62.5 = 1$ GHz. As each M-QAM symbol contains log_2(M) bits, the achievable throughput for each of the supported modulation format with carrier frequency of 100 Hz is derived as follows: 16-QAM: $4 * 16 * 62.5 = 4$ Gb/s; 32-QAM: $5 * 16 * 62.5 = 5$ Gb/s; 64-QAM: $6 * 16 * 62.5 = 6$ Gb/s. Respective theoretical SNR requirements are 11.71 dB, 14.68 dB and 17.58 dB.

C. Design Resources and Performances with FEC

All QAM modulation orders from 16 to 256 are synthesized, implemented and simulated in Vivado for Virtex UltraScale + (xcvu9pflga2104-1L) with filter orders of 11 and 31 which in turn have highest impact on the number of points required for FFT, thereby determining the potential number of parallel inputs. The SRRC filter with a filter order of 11 necessitates 16-point FFT with 16 parallel inputs and that with a filter order of 31 requires 32-point FFT with 32 parallel inputs.

For **N** = 16, ipcores *Complex Multiplier v6.0* and *Multiplier v12.0* are implemented using mults and *Adder Subtracter v12.0* uses fabric (LUTs) in performance optimization mode. Timing summary shows that the longest propagation path delay (critical path) is in parallel DFT and IDFT. Since critical path influences the maximum achievable frequency of the system, pipelining approach is adopted with additional registers being packed into DFT and IDFT modules with the objective of reducing the path delay. With regard to performance, Table III lists the maximum clock frequency and the achievable throughput of all the supported QAM orders for a carrier frequency of 100 Hz. Though a FEC code rate of 1/3 will enable 64-QAM to withstand noisy channels due to its higher data redundancy, performance gain is repealed by the encoder's overhead. As shown in the comparison chart 14, on one hand, incorporation of FEC technique with a code rate of 1/2 reduces the SNR requirement of the QAM transmitter. On the other hand, it is remarkable that despite its overhead, it has shown a significant improvement in throughput by almost 60% for all the modulation orders. This is mainly due to the architectural improvements (such as pipelining) over [9] and the availability of more resources in Virtex UltraScale + which allows higher degree of parallelization. Nevertheless, with FEC, the SNR improvement for 256-QAM is 73% while an enhancement of 60% is obtained for 16-QAM.

For **N** = 32, due to limitation of the available DSP slices and LUTs, both *Complex Multiplier v6.0* and *Multiplier v12.0* are implemented to operate in area optimization mode. In this mode, utilization of DSP slices gets reduced to 3 from 4 in *Complex Multiplier v6.0* and multiplication is split between DSP slices and LUTs in *Multiplier v12.0*. Table IV shows the maximum clock frequency, the achievable throughput and the resource utilization of all the supported QAM orders for a carrier frequency of 100 Hz. When compared to **N** = 16 parallel inputs, the operating frequency has been reduced notably, however higher parallelization of processing 32 inputs concurrently enables the system to yield larger throughput. It is also interesting to observe that the maximum clock frequency follows an irregular trend-line. This may be due to the heuristics employed in place&route steps and their partially random characteristics that can trap the system optimization in local minima, missing the global minimum, thus leading to sub-optimal results, especially in circuits of large complexity.

TABLE II
RESOURCE UTILIZATION

Resources	DSP - Mults	Fabric - LUTs	Fabric - Mults
Slice Registers	5%	8%	5%
Slice LUTs	1%	12%	7%
LUTs used as Logic	1%	11%	6%
Occupied Slices	14%	22%	17%
Unused Flip Flop	6%	22%	28%
Unused LUTs	82%	33%	51%
Bounded IOBs	46%	46%	46%
DSP48E1s	92%	54%	57%

TABLE III
ACHIEVABLE THROUGHPUT WITH FEC FOR 16 PARALLEL INPUTS

QAM Order	Code Rate	Max.Freq (MHz)	Throughput (Gb/s)
16	1/2	202.96	6.5
32	1/2	201.74	8.07
64	1/2	178.41	8.57
64	1/3	202.3	6.47
128	1/2	191.68	10.73
256	1/2	201.25	12.88

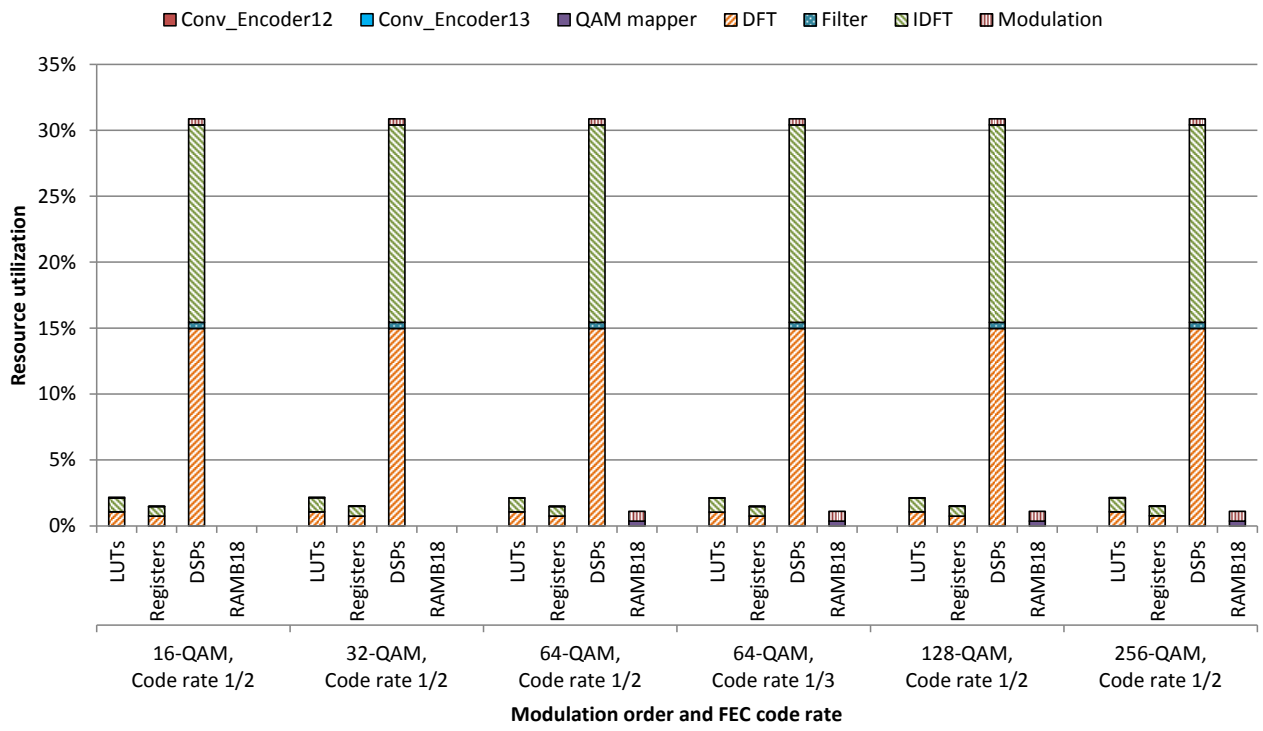


Fig. 13. Resource Utilization of QAM Transmitter for **N** = 16 parallel inputs

The relative module level resource utilization of QAM transmitter for 16 parallel inputs is shown in Fig. 13. It is easy to distinguish that most of the LUTs, flip-flops and DSP slices are utilized by parallel DFT and IDFT. Therefore, the increase in modulation order does not propose a significant threat on resource overhead. Figs. 15, 16, 17, and 18 show the relative resource utilization comparison between **N** = 16 and **N** = 32 for modules *convolution encoder*, *QAM mapper*, *filter* and *modulator*. *DFT* and *IDFT* are omitted in the comparison chart for the purpose of detailed readability. From 64-QAM onwards, *QAM mapper* make uses of BRAMs rather than LUTs and registers. This choice of decision is done automatically by the tool as it follows its own optimization technique in order to make an effective trade-off between area and speed. Similarly, the *modulator* also employs BRAM cells and exercises little usage of register blocks but for **N** = 32, it prefers LUTs and registers over BRAMs. Even though the modules *filter* and *modulator* engage almost 2% of DSP slices when compared to 15% utilization by *DFT* and *IDFT* (see Fig. 13), they are not sensitive to increase in modulation orders as they are essential only for arithmetic operations.

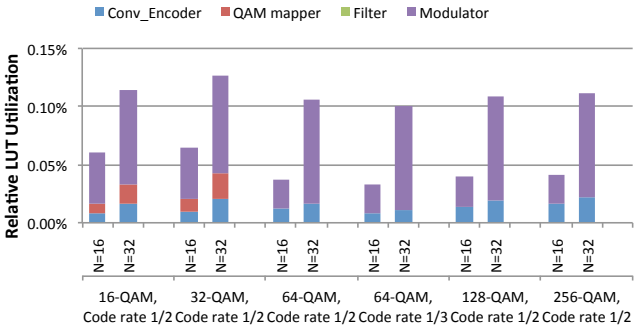


Fig. 15. Relative LUT Utilization Comparison

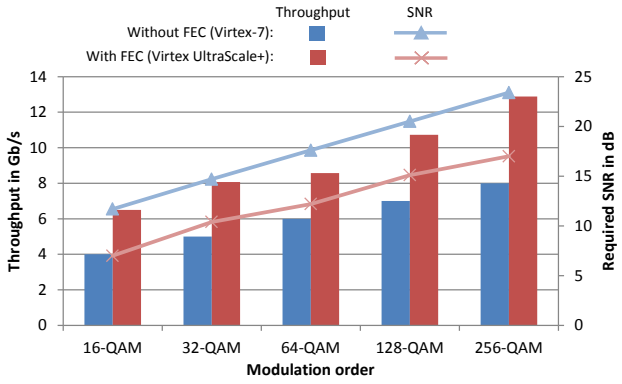


Fig. 14. Throughput comparison between "with FEC" and "without FEC" [9]

TABLE IV
QAM TRANSMITTER WITH FEC FOR 32 PARALLEL INPUTS

QAM Order	Max.Freq (MHz)	Through-put (Gb/s)	LUTs	Registers	DSPs	BRAM
16	168.55	10.79	121917	144613	6272	-
32	112.37	8.99	122073	145007	6272	-
64	134.39	12.90	121831	144799	6272	32
128	130.02	14.56	121864	144992	6272	32
256	146.16	18.71	121896	145184	6272	32

Furthermore, we have observed that a resource utilization of 91.7% is possible with Virtex UltraScale + due to its improved routing architecture, thereby making it feasible to process 32 inputs in parallel, while for Virtex - 7 routing congestions would impede such a high resource utilization limiting the

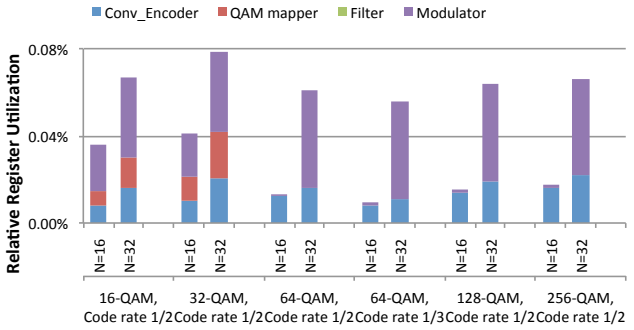


Fig. 16. Relative Register Utilization Comparison

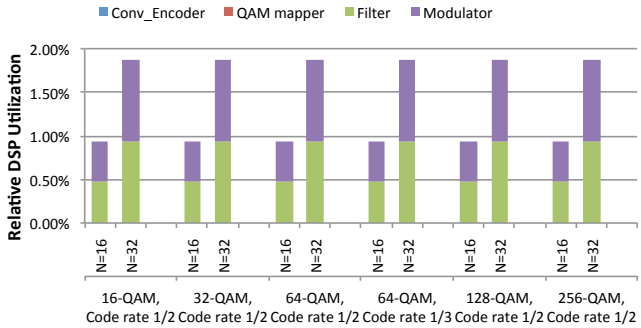


Fig. 17. Relative DSP Slices Utilization Comparison

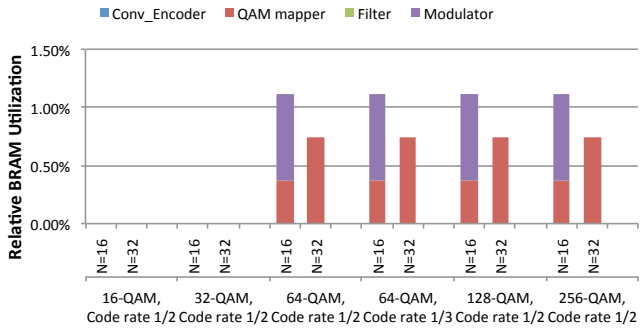


Fig. 18. Relative BRAM Utilization Comparison

degree of parallelization to a maximum of 16 parallel inputs.

V. Conclusion

This paper describes a new approach to optimize the performance of high-speed Quadrature Amplitude Modulation implemented on FPGAs by exploiting the advantageous properties of mixed time and frequency domain approach. While standard transmitters operating entirely in time domain need to process data serially due to the convolution nature of filtering operation, this mixed-domain transmitter has the theoretical capability to work with an arbitrary number of parallel inputs N. Moreover, ISIs due to crowding of symbols in higher order QAM constellations and the effect of noisy channels during data transmission have been weakened by the appendage of FEC technique as data redundancy curtails the SNR requirement of the system. The design has been simulated, synthesized, routed and tested on a Xilinx Virtex UltraScale+ FPGA platform for $\mathbf{N}=16$ and 32 parallel inputs and for multiple QAM formats (16-QAM, 32-QAM, 64-QAM, 128-QAM and 256-QAM) with a system precision of 16 bits, Although FEC overhead can impair the performance gain of the system, additional architectural improvements by pipelining and availability of bounteous resources in Virtex UltraScale+ have boosted the system performance by almost 60% when compared to [9] for 16 parallel inputs. Besides, the routing technology of UltraScale+ also plays a role in utilizing more than 90% of the available resources without congestion issues. With $\mathbf{N}=16$ parallel inputs, an effective throughput of 12.8 Gb/s is achievable for 256-QAM, while a 14.5 Gb/s effective throughput is realized with $\mathbf{N}=32$ parallel inputs even for 128-QAM and a maximum of 18.7 Gb/s with 256-QAM. In addition to the aforementioned higher performance in terms of throughput and SNR requirement, the realized system is extensively generic with the filter order, the number of parallel inputs N and the desired QAM format being scalable through core parameterization.

Acknowledgment

The authors would like to thank the Karlsruhe School of Elementary Particle and Astroparticle Physics (KSETA) and the Helmholtz International Research School for Teratronics (HIRST) for their substantial support.

References

[1] G. Shalina, P. Figuli, and J. Becker, "Parametric design space exploration for optimizing qam based high-speed communication," *IEEE/CIC International Conference on Communications in China*, November 2015.

[2] M. Ferrario, A. Spalvieri, and R. Valtolina, "Design of transmit fir filters for fdm data transmission systems," *Communications, IEEE Transactions on*, vol. 52, no. 2, pp. 180–182, Feb 2004.

[3] Y. Wu and Y. Shayan, "Implementation of high-speed multi-level qam modems based on xilinx virtex-ii fpga," *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, vol. 1, pp. 195–198 vol.1, May 2003.

[4] X.-T. Vu, N. A. Duc, and T. A. Vu, "16-qam transmitter and receiver design based on fpga," *Electronic Design, Test and Application, 2010. DELTA '10. Fifth IEEE International Symposium on*, pp. 95–98, Jan 2010.

[5] V. Smolyakov, D. Patel, M. Shabany, and P. Gulak, "A wimax/lte compliant fpga implementation of a high-throughput low-complexity 4x4 64-qam soft mimo receiver," *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on*, pp. 385–389, Nov 2010.

[6] S. Ma and Y. Chen, "Fpga implementation of high-throughput complex adaptive equalizer for qam receiver," *Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on*, pp. 1–4, Sept 2012.

[7] A. Al-Bermani, C. Woerdehoff, O. Jan, K. Puntsri, M. Panhwar, U. Rueckert, and R. Noe, "The influence of laser phase noise on carrier phase estimation of a real- time 16-qam transmission with fpga based coherent receiver," *Photonic Networks, 14. 2013 ITG Symposium. Proceedings*, pp. 1–4, May 2013.

[8] M. Stackler, A. Glascott-Jones, and N. Chantier, "A high speed transmission system using qam and direct conversion with high bandwidth converters," *Aerospace Conference, 2015 IEEE*, pp. 1–8, March 2015.

[9] S. Figuli, A. Sonnino, P. Figuli, and J. Becker, "A variable fpga based generic qam transmitter with scalable mixed time and frequency domain signal processing," *39th International Conference on Telecommunications and Signal Processing (TSP)*, July 2016.

[10] P. Elias, "Coding for noisy channels," in *Proc. 3rd London Symposium of Information Theory*, September 1955.

[11] E. Agrell, J. Lassing, E. G. Strom, and T. Ottosson, "Gray coding for multilevel constellations in gaussian noise," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 224–235, Jan 2007.

[12] A. Ashrafi and F. J. Harris, "A novel square-root nyquist filter design with prescribed isi energy," *Signal Process.*, vol. 93, no. 9, Sep. 2013.