A Variable FPGA Based Generic QAM Transmitter with Scalable Mixed Time and Frequency Domain Signal Processing

Shalina Percy Delicia Figuli*, Alberto Sonnino†, Peter Figuli* and Jürgen Becker*
* Institute for Information Processing Technologies, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
Email: {shalina.ford, peter.figuli, becker}@kit.edu
† Department of Computer Sciences, University College London (UCL), London, UK
Email: alberto.sonnino.15@ucl.ac.uk

***Abstract*—The flexibility of Field Programmable Gate Arrays (FPGAs) as well as their parallel processing capabilities make them a good choice for digital signal processing in communication systems. However, today, further improvements in performance hang in mid-air as we run into the frequency wall and FPGA based devices are clocked below 1 GHz. New methodologies which can cater performance optimization within the frequency wall limitation become highly essential. In this context, efficient modulation techniques like Quadrature Amplitude Modulation (QAM) and mixed time and frequency domain approach have been utilized in this paper to employ a generic scalable FPGA based QAM transmitter with the filter parallelization being executed in mixed domain. The system developed in this paper achieves a throughput of 4 Gb/s for QAM-16 format with a clock frequency as low as 62.5 MHz, thereby, 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

The growth of Field Programmable Gate Arrays (FPGAs) in the field of digital communication technology not only claims 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 embraced and fostered. In order to transmit QAM signals, the crucial setting is to band limit the transmitted signals and also at the same time to suppress the Intersymbol Interference (ISI). In that purpose, the Square Root Raised Cosine (SRRC) filter is one of the most frequently used pulse shaping Finite Impulse Response (FIR) filters in digital modulation. 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 convolutional form acts as a significant speed limitation in digital communication systems. This can be 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 focusses on developing a variable FPGA based generic QAM transmitter, where the user has a variety of options to choose from with regard to modulation orders, filter coefficients (filter order) and degree of parallelization (number of parallel inputs) through core parametrization. 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 using Xilinx Virtex II FPGA. Moreover, their filter selection is same as the one considered in this paper and they were able to reach an operating frequency of 55 MHz [3]. A complete 16-QAM with an achievable frequency of 111.11 MHz was built using two Xilinx Virtex IV FPGA boards, one for the transmitter and one for the receiver respectively in 2010 ([4]). The very same year, in [5], a 64-QAM receiver based on Xilinx Virtex V 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 a Xilinx Virtex IV FPGA platform [6]. The next year, a 16-QAM transceiver on a Xilinx Virtex VI board with an achievable frequency of 625 MHz at the cost of low precision has been brought

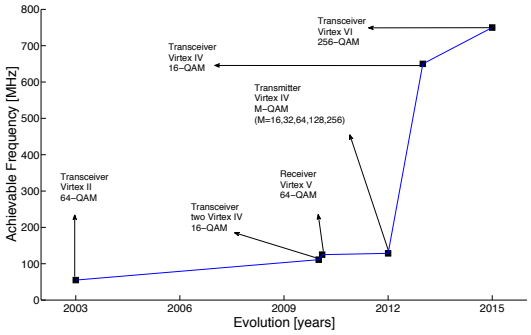

Fig. 1. Today's state-of-the-art

The authors would like to thank the Karlsruhe School of Elementary Particle and Astroparticle Physics (KSETA) and the German Academic Exchange Service (DAAD) for their substantial support.

forth [7]. More recent state-of-the-art took advantage of the powerful Xilinx Virtex VI 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 acheived by ASIC platforms and multi-FPGA systems, our work is confined to single channel FPGA systems. The rest of the paper is organized as follows: Section 2 throws lime light into fundamental aspects of the employed mixed domain QAM transmitter concept. It's implementation is covered in Section 3 with experimental results substantiated quantitatively in Section 4, and conclusions summarized in Section 5.

II. Fundamentals and Concept of Mixed Domain QAM Transmitter

Due to the convolutional nature of the filtering process, the filter input needs to be fed in sequentially which detrains the potency of the system when the preference shifts to system parallelization. Therefore, a mixed domain approach with the filter operation being shifted to frequency domain in order to accompany parallel inputs and outputs as shown in Fig. 2 is utilized. The steps of the illustrated signal processing from left to right are explained in the following subsections.

A. QAM Mapper

The input bit stream is clustered into $k = \log_2(M)$ bits 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. There are many ways to associate a symbol and certainly grey code is the most common choice [9] as it reduces the erroneous symbol decision to one bit error. The 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.

B. Fourier Transform

Fourier Transform (FT) is a mathematical tool that deconstructs the signal into its sinusoidal components and, similarly, Inverse FT (IFT) is the tool to reverse it. More specifically, Discrete Fourier Transform (DFT) takes the interleaved components from the 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 then taken back in time domain by Inverse Discrete Fourier Transform (IDFT).

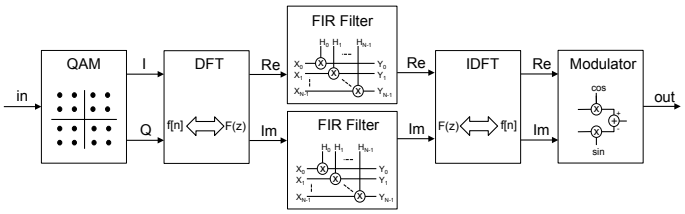

Fig. 2. Mixed-domain QAM transmitter

C. Square Root Raised Cosine Filter

The 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 [10]. 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.

D. 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 and imaginary components (I and Q respectively) and then they are 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 into the same bus as shown in Fig. 3 with each $data_i$ being a 16-bit bus. The QAM transmitter accepts an arbitrary number of parallel inputs, a customizable filter order and supports multiple modulation formats. Despite the fact that the system has been formulated to achieve the highest order of modularity, the 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**. The dotted lines represent that each main module is completely isolated from others and has its own parameterizable interface and thereby, the system designers

Fig. 3. Parallel bus packing

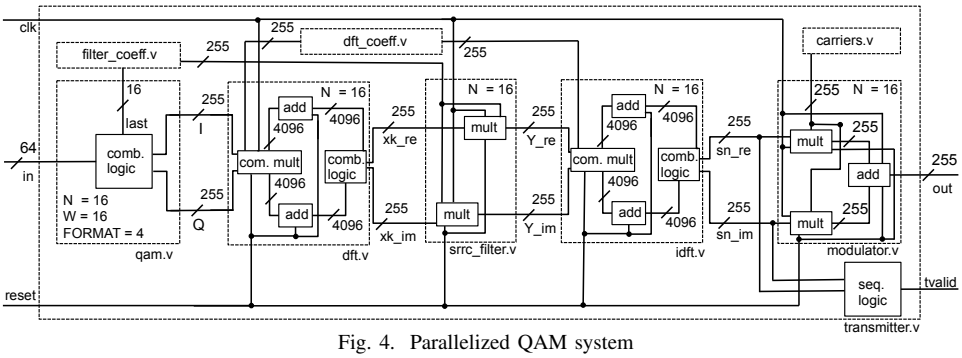

Fig. 4. Parallelized QAM system

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*" puts up with two input parameters **N** and **FORMAT**, which represent the number of parallel inputs and the desired modulation format respectively. Table I shows an abstract view of the system focussing on its input and output ports. The input stream **(in)** is clustered by the number of bits defined by the Verilog parameter **FORMAT**. For example, for a 16-QAM modulation, the user needs to enter **N** parallel inputs of length specified by **FORMAT** = $\log_2(16)$ = **4**, which is then packed into a single bus of width **(FORMAT*N)**. 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* has been developed to aid in breaking the complexity of the design usage. By default, fifteen rows are available to enter the filter coefficients, while extra rows can be added or suppressed depending upon the desired filter order. Once the carrier frequency has been specified, 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, explict 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.

TABLE I. TRANSMITTER - SPECIFICATIONS

`transmitter.v` - Receives N 16-bit inputs and outputs N 16-bit modulated waves		
Latency	17 cycles	
Parameters	*N*	Number of parallel inputs
	FORMAT	QAM order
Inputs	*clk*	Clock
	reset	Reset
	in	Clustered input stream
Outputs	*tvalid*	Output's valid flag
	out	Output

[1]Even if the DFT algorithm doesn't 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.

B. QAM Mapper

It receives **N** clustered inputs and delivers **N** corresponding I and Q signals. This module has been set up using three Verilog parameters **N**, **W** and **FORMAT**. Respectively, they indicate the number of parallel inputs, the bus width and the 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 8, 16 (default), 32 and 64-QAM. Since flexibility is also a salient standard of this QAM transmitter, each modulation format has been implemented in a separate Verilog file *qam.v* in order to encourage the users to extend this system with any other modulation format and then, through a very basic modification of *qam.v* module, to allow the transmitter's top entity to use the newly added format.

As the modulation order specified by **FORMAT** parameter alone gets generated, changing the order during execution time is not possible. Moreover, an accurate reader will notice that generating all the modulation formats even 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.

C. 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 is inputted by **N**, which can be either viewed as the transform length or the number of parallel inputs. In addition to clock and reset signals, this module requires the time domain (resp. frequency domain) complex input 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 the transformation process. Unfortunately, the current FFT Xilinx IP core cannot be used in this parallel design as it inputs and outputs data serially. Therefore, in order to achieve the aimed parallelization, the discrete Fourier and the inverse discrete Fourier transforms have been implemented in hardware. The *Complex Multiplier v5.0* core is used N^2 times, while the *Adder Subtracter v11.2* core is used $2N(N-1)$ in *adder* mode, both working with 16-bit inputs. This fair amount of core's

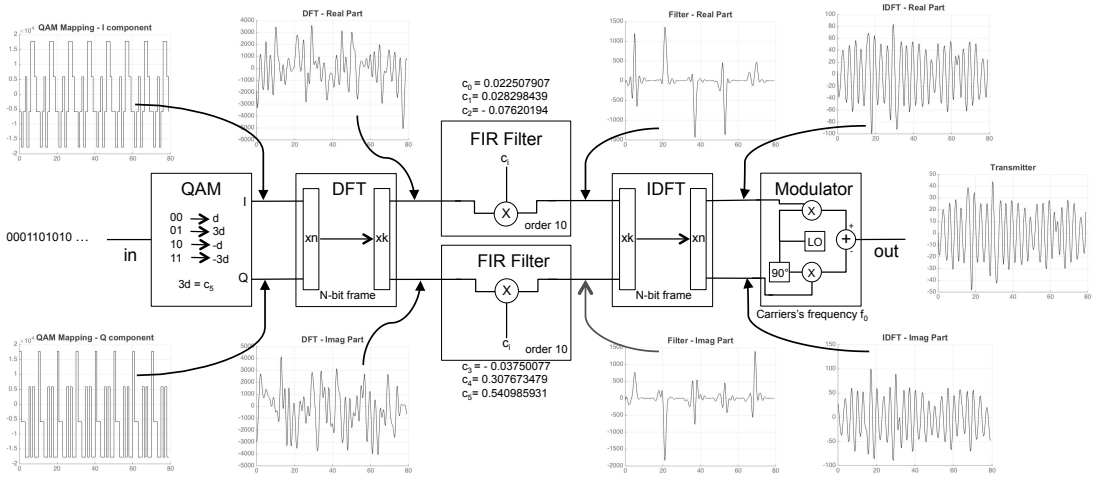

Fig. 5. MATLAB reference model of the QAM transmitter

instance is required as both imaginary and real parts have to be processed. Since these additions need synchronization, the core is manually configured to have zero latency. Finally, the output is rescaled by 2^{-17} to be able to fit into the desired 16-bit bus and to avoid possible overflow.

D. Filter

The filter's implementation is one of the main targets of this paper. Implementing the filter in the frequency domain is much simpler than in the time domain as it requires simple multiplication of inputs with coefficients, and most importantly, it is easily parallelizable, thus, revealing the urge for having a mixed domain transmitter. In order to isolate this block and to ensure its reusability, the parameter **N** sets up the number of parallel inputs and the module accepts an arbitrary number of filter coefficients through the 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 $2N$ instances of *Multiplier v11.2*, which works with 16-bit inputs, generates 16-bit symmetrically rounded outputs and rescales the output by 2^{-16}.

E. Modulator

This block, parameterized by **N**, works with any carrier frequency, which has been set up in the *FQM Utility* and passed on to this module through the configuration file *carriers.v*. The real input gets multiplied with cosine carrier and the imaginary with sine carrier. These products are then subtracted according to the following equation:

$$\begin{aligned} out(t) &= \mathcal{R}\left\{[I(t)+iQ(t)]e^{2\pi f_0 t}\right\} \\ &= I(t)\cos(2\pi f_0 t) - Q(t)\sin(2\pi f_0 t) \end{aligned} \quad (1)$$

Multiplier v11.2 and *Adder Subtracter v11.2* are the two cores that have been instantiated in this module with the multiplier's configuration exactly as same as that of in the filter module and the *Adder Subtracter* core is configured 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 $2N$ *Multiplier* cores and N *Adder Subtracter* cores with the *Multiplier* core rescaling the output by 2^{-16}.

IV. Experimental Results

In addition to the hardware implementation on Xilinx Virtex-7 VC707 evaluation board, a complete MATLAB model as shown in Fig. 5 has been developed in order to pre-evaluate the expected behavior of the system and to serve as a reference for the implemented system. By consequence, each block constituting the system has been first realised in MATLAB using the physical and mathematical fundamentals explained before and then the system's performances and resources 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. 6) in order to observe the system's precision with the assumption that the MATLAB simulation is considered as 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 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, many other sets of random input samples have also been tested and the precision still appears to be very similar to the one exposed above. The outputs

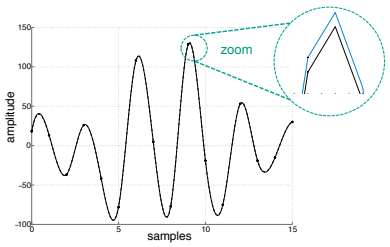

Fig. 6. Transmitter's precision comparison

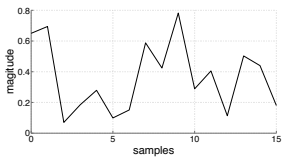

Fig. 7. Transmitter's error

of DFT and IDFT blocks are also examined by comparing their real and imaginary values with that of the MATLAB simulated values and it is observed that most of the errors in the transmitter system are from the DFT and IDFT processes.

B. Design Resources and Performances

All the simulations in this section have been made by selecting the parameter **N** to 16 and with a carrier frequency of 100 Hz. Firstly, the resource requirements and the achievable performance when adders and multipliers are configured to use DSP instead of fabric, and Mults instead of LUTs respectively are investigated. From Fig. 8 and 1st column of Table II, it is clear that selecting DSP option for adder configuration is not optimal as the total usuage of DSP48E1 blocks is 92% and the maximum achievable frequency is 28.77 MHz, which after place and route gets reduced to 28.57 MHz. Though routing such a huge amout of DSP blocks requires much effort, simulations have been done for all the 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. 9 and 2nd column of Table II reveal that the system performance is improved by a factor of 3 yielding a clock frequency of 58.82 MHz, as the DSP utilization is reduced by 50%. When fabric for adders and Mults for mulitipliers are exercised, a slight increase of 62.5 MHz is obtained (Fig. 10). 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 remain unused. Nevertheless, since the system receives **N = 16** parallel inputs, the effective speed is: $16 * 62.5 = 1\,GHz$. From this observation, since each M-QAM format's symbol contains $log_2(M)$, the achievable throughput for each of the supported modulation format with carrier frequency of 100 Hz is derived as follows: 8-QAM: $3 * 16 * 62.5 = 3\,Gb/s$; 16-QAM: $4{*}16{*}62.5 = 4\,Gb/s$; 32-QAM: $5{*}16{*}62.5 = 5\,Gb/s$; 64-QAM: $6 * 16 * 62.5 = 6\,Gb/s$.

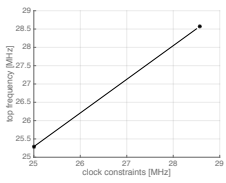

Fig. 8. DSP - Mults performance

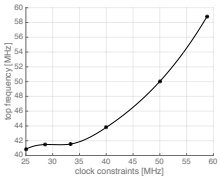

Fig. 9. Fabric - LUTs performance

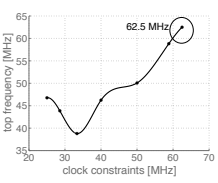

Fig. 10. Fabric - Mults performance

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%

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 a mixed time and frequency domain approach. While standard transmitters operating entirely in time domain need to process serial data due to the convolutional nature of the filtering operation, this mixed-domain transmitter has the theoretical capability to work with an arbitrary number of parallel inputs N. The design has been simulated, synthesised, routed and tested on a Xilinx Virtex 7 FPGA kit with a precision of 16 bits, for N = 16 parallel inputs and for multiple QAM formats; i.e. 8-QAM, 16-QAM, 32-QAM and 64-QAM. However, the concept can be generalised to more parallel inputs and other modulation formats. After a long place and route operation, a top clock frequency of 62.5 MHz has been reached while processing 16 parallel inputs with a carrier frequency of 100 Hz. Therefore, this implementation offers an effective speed of 1 GHz. In addition to the high achieved performances, the realized system is extensively generic. Indeed, an arbitrary number of filter coefficients for the FIR filter, the number of parallel inputs N and the desired QAM format can be chosen by the user though the core parameterization.

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] 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.

[10] 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.