Implementation of an IFFT for an Optical OFDM Transmitter with 12.1 Gbit/s

Michael Bernhard, Joachim Speidel
Universität Stuttgart, Institut für Nachrichtenübertragung, 70569 Stuttgart
E-Mail: bernhard@inue.uni-stuttgart.de

Abstract

This paper describes the design of an inverse discrete Fourier transform (IDFT) for an optical orthogonal frequency division multiplexing (O-OFDM) transmitter for a bitrate of $12.1\,\mathrm{Gbit/s}$. The complete transmitter was implemented on a Virtex 5 FX200T field programmable gate array (FPGA) from Xilinx. The main part of the transmitter, which needs the most signal processing hardware resources, is the IDFT. A 256-point radix-2 inverse fast Fourier transform (IFFT) was implemented.

1 Introduction

In wireless systems such as wireless local area network (WLAN) the feasibility of OFDM has been proven for a long time as well as in wireline systems like digital subscriber line (DSL). OFDM for optical communication systems is an important topic in research. The big challenge is the high data rate. This is a main difference to state of the art systems.

It is turned out that O-OFDM is more dispersion tolerant compared to conventional systems [1], [2]. The chromatic dispersion of optical fiber can be seen as a frequency-selective property of the channel. OFDM divides in general a broadband channel into multiples of smaller sub channels which can be considered approximately as non frequency selective. So the distortion at the receiver side can be reduced dramatically. Another advantage is, that it can be easily used in a dynamically reconfigurable network.

The IFFT with butterfly structure is in principle already highly parallel, which makes it suitable for an FPGA implementation. Fast multipliers with low complexity are crucial for the design. Available multiplication cores of current FPGA vendors are not suitable for the envisaged data rate of $12.1\,\mathrm{Gbit/s}$. Therefore we designed a dedicated multiplication unit. The principle structure is based on an optimized shift and add multiplier. For each twiddle factor of the IFFT butterfly we have designed an optimized multiplier.

The word lengths of the twiddle factors and signals for processing are crucial because of the restricted hardware resources within the FPGA. In each stage of the butterfly we have optimized word lengths by saturation or by truncation of data. By simulation we found a good compromise between the required word lengths and the quantization noise at the output of the OFDM transmitter.

Chapter 2 gives a short survey on the IFFT, chapter 3 describes the butterfly unit and the multiplication. The results are shown in chapter 4 and the conclusion is given in chapter 5.

2 Inverse fast Fourier transform

The N-point IDFT with $\{n \in \mathbb{Z} \mid 0 \leq n \leq N-1\}$ is given by:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, \mathrm{e}^{j2\pi kn/N} \tag{1}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn} \tag{2}$$

The kernel W_N^{kn} with $\{k, n \in \mathbb{Z} \mid 0 \leq k, n \leq N-1\}$ is defined as

$$\mathrm{W}_N^{kn} = \mathrm{e}^{\,-j(2\pi/N)kn} \tag{3}$$

Equation (2) can be decomposed into two sums:

$$x(n) = \frac{1}{N} \sum_{m=0}^{N/2-1} X(2m) W_N^{-2mn} + \frac{1}{N} \sum_{m=0}^{N/2-1} X(2m+1) W_N^{-n(2m+1)} \tag{4}$$

$$= \frac{1}{N} \sum_{m=0}^{N/2-1} F_1(m) W_{N/2}^{-mn} + W_N^{-n} \frac{1}{N} \sum_{m=0}^{N/2-1} F_2(m) W_{N/2}^{-mn} \tag{5}$$

$$= f_1(n) + W_N^{-n} f_2(n) \tag{6}$$

$f_1(n)$ and $f_2(n)$ is the inverse Fourier transform of $F_1(m)$ and $F_2(m)$, respectively. The structure is recursive because $f_1(n)$ and $f_2(n)$ can be decomposed into two smaller inverse Fourier transforms and so on. This

is the well known IFFT which was presented by Cooley and Tukey in 1965 [3].

3 Radix-2 IFFT butterfly unit

In Fig. 1 the basic butterfly unit for the radix-2 IFFT algorithm is shown.

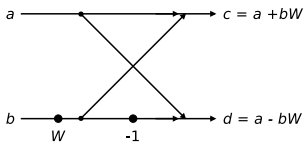

Fig. 1. Butterfly unit of a radix-2 IFFT.

There are two inputs a and b and two outputs c and d. The twiddle factor is W. For the output we have to calculate:

$$c = a + bW \tag{7}$$

$$d = a - bW \tag{8}$$

With these butterfly units you can build the butterfly diagram. Fig. 2 shows an example of an 8-point radix-2 IFFT butterfly diagram.

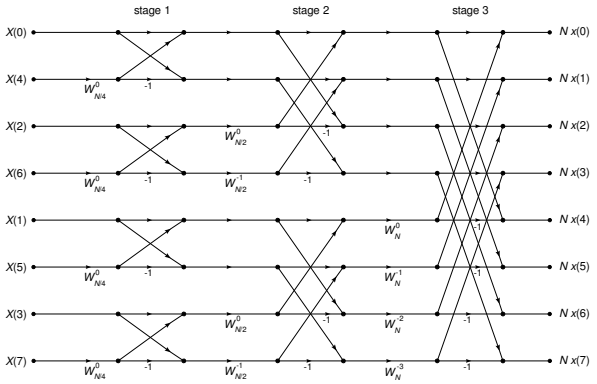

Fig. 2. Butterfly diagram of an 8-point radix-2 IFFT.

On the left hand side we have the frequency domain and on the right hand side the time domain.

Implementation can only be done with limited accuracy of calculations and a suitable number representation is required which is discussed in the next section.

3.1 Number representation

For the number representation a fixed point scheme is used. Floating point arithmetic is not required because in our application the order of magnitude of the input and the output of the IFFT are similar. This also holds for the order of magnitude in each stage of the butterfly diagram. The complex numbers are given by real and imaginary part.

The twiddle factors W are complex with $\mid W \mid = 1$. This means that the magnitude of the real part and the imaginary part of W is between zero and one. To represent these numbers they are multiplied with a scaling factor 2^s with $s \in \mathbb{N}$ and rounded to an integer number. So real part and imaginary part of W range from -2^s up to 2^s. For the complex multiplication we need the magnitude and the sign of the twiddle factor separately (compare to sections 3.3 and 3.4). The word length of the magnitude of W is $s+1$ bits to represent numbers from 0 to 2^s.

For the input of the IFFT integer values are used. They can be interpreted as fixed point numbers, of course. Let the word length of the input numbers be m for the real part and imaginary part respectively. Then with two's complement notation we get integer values between -2^{m-1} and $2^{m-1}-1$.

3.2 Required word lengths for a butterfly

In general, after an addition of two numbers the required word length is increased by one bit. After a multiplication of two unsigned numbers or one unsigned and one signed number the word length of the result is the sum of the word length of the two factors. If both factors are signed, the word length of the result is only the sum of the word length of the two factors minus one. Let w_1 and w_2 be the word length of the factors. Then we get the new word length $w_{12} = w_1 + w_2$ in the case when both numbers are unsigned or one unsigned and the other signed and $w_{12} = w_1 + w_2 - 1$ when both numbers are signed.

Due to the fact that the absolute value of the complex valued twiddle factors is one, we can determine an upper limit for the word length after multiplication.

For the butterfly in Fig. 1 we first have to calculate the complex multiplication bW. Let b_{max} be the maximum magnitude of an input value. Then the maximum magnitude of the real part of bW is:

$$\begin{aligned} \max\{\mid \Re\{bW\} \mid\} &= \max\{\mid \Re\{b\} \mid\} \cdot \sqrt{2} \\ &= b_{\max}\sqrt{2} \end{aligned} \tag{9}$$

Proof: For W we can write $W = \mathrm{e}^{j\varphi} = \cos\varphi + j\sin\varphi$. When we consider the real part of bW we get $\Re\{bW\} = \Re\{b\}\cos\varphi - \Im\{b\}\sin\varphi$. Let $\Re\{b\} = \Im\{b\} = b_{\max}$. Then follows:

$$\max\{\mid \Re\{bW\} \mid\} = b_{\max} \cdot \max\{\mid \cos\varphi - \sin\varphi \mid\} \tag{10}$$

To find the maximum we set the derivate of $f(\varphi) = \cos\varphi - \sin\varphi$ equal to zero.

$$f'(\varphi) = -\sin\varphi - \cos\varphi \stackrel{!}{=} 0 \tag{11}$$

The solution is $\varphi = \varphi_k = \frac{3}{4}\pi + k\pi$ with $k \in \mathbb{Z}$. The second derivation at φ_k is unequal to zero. Therefore we have at $f(\varphi_k)$ either a maximum or a minimum. If we insert φ_k into (10) we obtain:

$$\max\{\mid \Re\{bW\} \mid\} = b_{\max}\sqrt{2} \tag{12}$$

The same holds for the imaginary part and also if we choose $\Re\{b\} = -\Im\{b\} = b_{\max}$.

Let $\Re\{W\} \in [-2^s, \cdots, 2^s]$ and $\Re\{b\} \in [-2^{m-1}, \cdots, 2^{m-1}-1]$, so $b_{\max} = 2^{m-1}$ and then

$$\max\{|\,\Re\{bW\}\,|\} = 2^{m-1} \cdot \lceil 2^s \cdot \sqrt{2} \rceil \tag{13}$$

With this upper limit, we can calculate the required word length after the complex multiplication bW:

$$\begin{aligned} &\lceil 1 + \log_2(2^{m-1} \cdot \sqrt{2} \cdot 2^s) \rceil \\ &= 1 + m - 1 + s + \lceil 1/2 \rceil \\ &= m + s + 1 \end{aligned} \tag{14}$$

The additional one in (14) is because we have positive and negative numbers. If we increase the word length from m up to $m + s + 1$ after the complex multiplication, we can avoid an overflow.

3.2.1 Scaling in the butterfly

With fixed point format and the scaling factor 2^s for W we get from (7) and (8):

$$c = a + bW \cdot 2^s \tag{15}$$
$$d = a - bW \cdot 2^s \tag{16}$$

To represent a and $bW \cdot 2^s$ by the same fixed point format, we have to multiply with 2^{-s}, yielding

$$c = a + bW \cdot 2^s \cdot 2^{-s} \tag{17}$$
$$d = a - bW \cdot 2^s \cdot 2^{-s} \tag{18}$$

Note, that we cannot simplify $2^s \cdot 2^{-s} = 1$ because of fixed point representation of W.

The easiest way to multiply with 2^{-s} is doing a truncation and accepting some rounding errors.

For the addition in (7) and subtraction in (8) we extend the word length of a by one bit from m to $m+1$. Because we use the two's complement for a, we repeat the most significant bit (MSB) for the extension. Finally we get a word length at the output of the butterfly for the real part and imaginary part which is $m + 2$.

3.2.2 Further simplifications

From section 3.2.1 we know that in the general case, the word length from input to output of every stage increases by two bits. However, in the first stage the twiddle factors are all $W_2^0 = 1$. Consequently, the required word length after the first stage is $\lceil 1 + \log_2(2^m) \rceil = m + 1$.

In the second stage, the twiddle factors are $W_4^0 = 1$ and $W_4^{-1} = j$. So we have either the case like in the first stage or we get the output $c = a + b \cdot j$, and for the maximum of the magnitude we obtain $2^{m-1} + 2^{m-1} = 2^m$. In conclusion we can say, that in the first two stages the output length of a butterfly unit is increased by only one bit. For the higher stages, the increase by one butterfly unit is $1 + \sqrt{2}$. In general, for the higher stages (greater or equal to three) the output word length increase is:

$$2\,\text{bit} + \lceil \log_2\{(1+\sqrt{2})^{k-2}\} \rceil\,\text{bit} \tag{19}$$

where $k \in \{3, 4, \cdots N\}$.

In Table I the word length increase values for each stage k is listed.

TABLE I
INCREASE OF WORD LENGTH AT OUTPUT OF BUTTERFLY UNIT.

	word length increase	
stage k	input IFFT to output butterfly	input to output of butterfly
1	1	1
2	2	1
3	4	2
4	5	1
5	6	1
6	8	2
7	9	1
8	10	1
9	11	1
10	13	2

3.3 Complex multiplier

The most costly part of the IFFT is the complex multiplication. That is why we need an efficient solution to execute the multiplication. First, we separate the complex numbers into real part and imaginary part:

$$W = W_\text{r} + jW_\text{i} \tag{20}$$
$$a = a_\text{r} + ja_\text{i} \tag{21}$$
$$b = b_\text{r} + jb_\text{i} \tag{22}$$
$$c = c_\text{r} + jc_\text{i} \tag{23}$$
$$d = d_\text{r} + jd_\text{i} \tag{24}$$

($j^2 = -1$) Then complex multiplication of b and W can be done as follows [4]:

$$\begin{aligned} bW =& (b_\text{r} + jb_\text{i})(W_\text{r} + jW_\text{i}) && (25)\\ =& b_\text{r}(W_\text{r} + W_\text{i}) - W_\text{i}(b_\text{r} + b_\text{i}) \\ &+ j(b_\text{r}(W_\text{r} + W_\text{i}) + W_\text{r}(b_\text{i} - b_\text{r})) && (26) \end{aligned}$$

As W do not change, we do not have to calculate $W_\text{r} + W_\text{i}$ at runtime. We will calculate $W_\text{ri} = W_r + W_\text{i}$ once. Then we get

$$\begin{aligned} bW =& b_\text{r}W_\text{ri} - W_\text{i}(b_\text{r} + b_\text{i}) \\ &+ j(b_\text{r}W_\text{ri} + W_\text{r}(b_\text{i} - b_\text{r})) \end{aligned} \tag{27}$$

and only three real valued multiplications and three additions/subtractions are required.

As described above, the real and imaginary parts of the twiddle factors are in magnitude and sign format and the word length of the magnitude is $s+1$. For (27) we also have to store W_ri. Its magnitude has a word length of $s + 1\,\text{bits}$.

Proof: The maximum of $W_\mathrm{r} + W_\mathrm{i}$ is $\sqrt{2}$. So we have to store in the worst case $\sqrt{2} \cdot 2^s$. With $s+1\,\mathrm{bits}$ we can represent unsigned numbers from 0 to $2^{s+1}-1$. So the following equation must be fulfilled when we want to represent the value $\sqrt{2}$ as a fixed point number with the scaling factor 2^s:

$$2^{s+1} - 1 \geq \sqrt{2} \cdot 2^s \tag{28}$$

This equation is fulfilled for $s \geq 0.7715\cdots$. Consequently we can represent $\sqrt{2} \cdot 2^s$ with $s+1\,\mathrm{bits}$.

3.4 Shift and add multiplier

The concept behind the multiplication which is used here is the shift and add algorithm. For example we multiply $a = a_3a_2a_1a_0$ which is in two's complement representation with $b = b_4b_3b_2b_1b_0$ which is a constant unsigned number. The general principle is shown in Fig. 3.

					a_3	a_2	a_1	a_0	$\cdot$	
					b_4	b_3	b_2	b_1	b_0	
a_3	a_3	a_3	a_3	a_3	a_2	a_1	a_0			$\cdot b_0+$
a_3	a_3	a_3	a_3	a_2	a_1	a_0	0			$\cdot b_1+$
a_3	a_3	a_3	a_2	a_1	a_0	0	0			$\cdot b_2+$
a_3	a_3	a_2	a_1	a_0	0	0	0			$\cdot b_3+$
a_3	a_2	a_1	a_0	0	0	0	0			$\cdot b_4$
q_8	q_7	q_6	q_5	q_4	q_3	q_2	q_1	q_0		

Fig. 3. General shift and add multiplication of ab.

The coefficients a_i and b_i can either be 0 or 1. If a coefficient b_i is 0 we can drop this multiplication and the addition. So the number of required additions (n_1) is the Hamming weight (w) of b minus one:

$$n_1 = \mathrm{w}\,\{b\} - 1 \tag{29}$$

If the Hamming weight is maximal we need for the worst case s additions because the word length of b is $s+1$. To reduce the maximum required numbers of additions, we can use the one's complement of b to execute the multiplication. If the word length of b is $s+1\,\mathrm{bits}$ we can represent b using its one's complement in the following way:

$$b = 2^{s+1} - 1 - \bar{b} \tag{30}$$

Then we get for the number of required additions:

$$n_2 = 2 + \mathrm{w}\,\{\bar{b}\} - 1 \tag{31}$$

With $\mathrm{w}\,\{b\} + \mathrm{w}\,\{\bar{b}\} = s+1$ we get from (31):

$$n_2 = s - \mathrm{w}\,\{b\} + 2 \tag{32}$$

Depending on the Hamming weight, we choose either the first method with n_1 additions or the second method with the one's complement with n_2 additions to execute the multiplication. The maximum number of adders is now in the worst case $\frac{s+1}{2}$.

For the additions, an adder tree is being used. The multiplication with $2^x, x \in \{0, 1, \cdots, s\}$ is a simple left shift. An example is shown in Fig. 4 with $b = 11101001$. The abbreviation *slx* stands for shift left x bits. From (29) the number of additions is: $\mathrm{n}_1\,\{b\} = 4$.

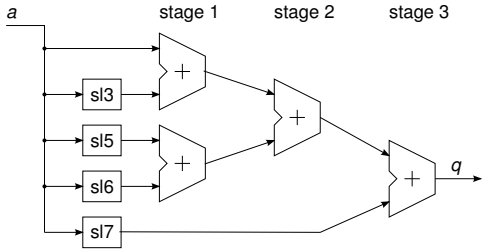

Fig. 4. Multiplication of ab using an adder tree. Constant factor $b = 11101001$.

3.5 Schematic of Butterfly unit

The schematic of the butterfly unit in the register transfer layer is shown in Fig. 5. It is based on Fig. 1 with additional elements. With *saturation/truncation* the output word length can be limited to a maximum fixed amount, either by saturation of data to a maximum value or by truncation. We do not use rounding operations because it costs more hardware resources. But truncations cause a larger quantization error. At the output in Fig. 5 registers for pipelining are used.

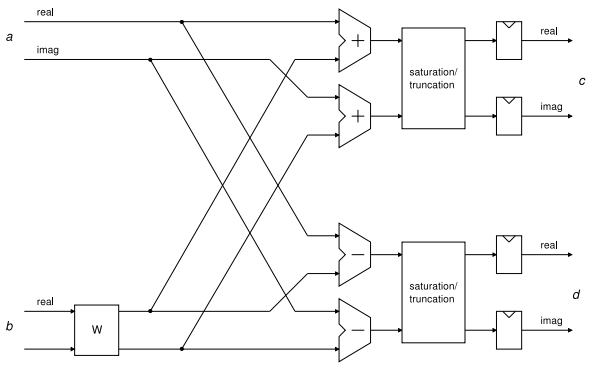

Fig. 5. Schematic of butterfly unit.

The complex multiplication according to (27) with the twiddle factor W is shown in Fig. 6 in more detail. Also the required word lengths are indicated.

3.6 Implementation of IFFT

For the implementation of the IFFT we use the highly parallel structure of Fig. 2 and the butterfly unit of Fig. 5. To save some hardware resources, some identical butterfly units in the inner stage are applied twice at double clock frequency.

We implemented a 256-point IFFT. Simulations have shown a good compromise between the required word length of the twiddle factor and the quantization noise at the output of the OFDM transmitter.

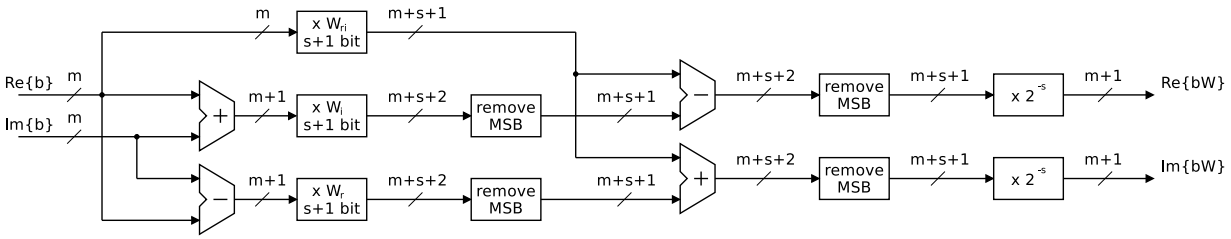

Fig. 6. Schematic of complex multiplier.

The design operates with $7\,\mathrm{bit}$ for the twiddle factors, i.e. $s = 5$. The input of the IFFT are quadrature phase shift keying (QPSK) symbols. At the output of the O-OFDM transmitter word length is reduced down to $6\,\mathrm{bit}$ to adapt to the resolution of the digital-to-analog (D/A) converter.

4 Results

The output of the IFFT exhibits a word length of $10\,\mathrm{bit}$ and a Q-factor of $30\,\mathrm{dB}$. The constellation diagram is shown in Fig. 7.

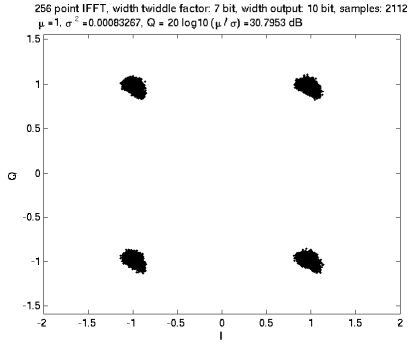

Fig. 7. QPSK constellation diagram at output of IFFT (word length $10\,\mathrm{bit}$)

The achieved Q-factor at the digital output of the O-OFDM transmitter is only $25\,\mathrm{dB}$, because of the $6\,\mathrm{bit/sample}$ resolution of the D/A converter. Fig. 8 shows the constellation diagram.

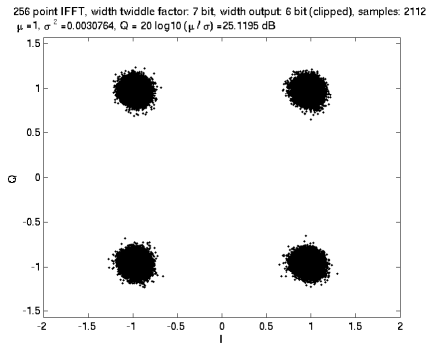

Fig. 8. QPSK constellation diagram at output of O-OFDM transmitter (resolution of D/A converter $6\,\mathrm{bit}$)

The implemented O-OFDM transmitter is verified by experimental laboratory tests with electrical back to back measurement including D/A and A/D conversion. The achieved Q-factor is $19.1\,\mathrm{dB}$ [5].

5 Conclusion

Optical OFDM has become an interesting method to achieve high data rates on optical fiber link. For implementation of a real-time O-OFDM transmitter, the IDFT is the main part, which costs most of hardware resources. Currently available IFFT cores do not reach high data rates in the region of $10\,\mathrm{GSample/s}$. Therefore, we have made an own dedicated design of the IFFT. Because the principle IFFT algorithm is already highly parallel, there is a large potential to reach a high throughput. For the twiddle factors of the IFFT we used dedicated multiplication cores which are based on an optimized shift and add multiplier. A complete O-OFDM transmitter with a 256-point radix-2 IFFT was implemented on a Xilinx Virtex 5 FX200T FPGA.

The achieved Q-factor at the digital output of the OFDM transmitter is about $25\,\mathrm{dB}$. With experimental test the realization of the IDFT and the complete OFDM transmitter was verified, and a Q-factor of $19.1\,\mathrm{dB}$ was achieved for electrical back to back measurements including D/A- and A/D conversion.

The implemented IFFT has a throughput of $10\,\mathrm{GSample/s}$. The structure of the IFFT can also be used for the FFT which is required for the O-OFDM receiver. There are only small modifications in the twiddle factors required.

This design can also be used for other applications in which an IFFT or an FFT with a high throughput is required.

Using the programming language C++ we also implemented an IFFT/FFT core generator which performs depending on some adjustable parameters the complete IFFT/FFT in the hardware description language VHDL. Depending on the available hardware resources and the required accuracy of calculation the word length of the twiddle factor can be chosen.

References

[1] W. Shieh, H. Bao, and Y. Tang, "Coherent optical OFDM: theory and design," *Opt. Express*, vol. 16, no. 2, pp. 841–859, 2008.

[2] A. J. Lowery and J. Armstrong, “Orthogonal-Frequency-Division Multiplexing for Optical Dispersion Compensation,” in *Optical Fiber Communication Conference and Exposition and The National Fiber Optic Engineers Conference*. Optical Society of America, 2007, p. OTuA4.
[3] J. W. Cooley and J. W. Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series,” *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.
[4] D. E. Knuth, *The Art Of Computer Programming*, 2nd ed. Addison-Wesley, 1981, vol. 2 / Seminumerical Algorithms.
[5] F. Buchali, R. Dischler, A. Klekamp, M. Bernhard, and D. Efinger, “Realisation of a real-time 12.1 Gb/s optical OFDM transmitter and its application in a 109 Gb/s transmission system with coherent reception,” *Optical Communication, 2009. ECOC ’09. 35th European Conference on*, vol. 2009-Supplement, pp. 1 –2, Sept. 2009.

Acknowledgement

The authors would like to thank Dr. Fred Buchali and Roman Dischler of Alcatel Lucent Bell Labs, Stuttgart, for helpful discussions.