Chapter 8.

Conclusions

Contents

8.1. Summary of the Results

Considering the increasing trend toward the hardware simulation of complex communication systems on prototyping platforms, one of the goals of this thesis was to develop a hardware-efficient wireless **channel simulator** with Doppler fading. The proposed architecture is very scalable and requires very few resources on FPGA. Moreover, it has also been implemented very efficiently in software and used for simulations throughout this thesis.

For modeling the channel we have used the WSSUS (wide-sense stationary with uncorrelated scattering) assumption, the channel being modeled as a tapped delay line with time-varying (fading) coefficients. The latter are narrow-band complex Gaussian processes and were implemented using the white Gaussian noise filtering method. Our selected approach for Gaussian noise generation relies on the central limit theorem, whereby a normally distributed variable can be obtained by summing up a large number of independent uniform variables. The proposed architecture sums a power-of-two number of uniform binary variables generated using LFSR's (linear feedback shift registers). An essential advantage is the flexibility of trading throughput for silicon area, as the number of random bits per clock cycle and the number of clock cycles per generated sample are configurable or parameterizable.

As the typical Doppler spreads are very small relative to the sampling rate, the resulting discrete bandwidth of the fading processes is extremely small, in the order of 1/1000. This fact alone renders a spectrum shaping by mere filtering unfeasible. Instead, a combination of filtering and interpolation is preferred. Various Doppler spreads can be generated by using a fixed low-pass filter followed by a polyphase interpolator with variable and typically large interpolation factors. Existing interpolation solutions for large factors use a multi-stage approach with fixed

interpolation factors. In our proposed solution, however, interpolation is performed in a single stage and the interpolation factor is configurable and not restricted to integers alone.

Thus, the tap generator is a multi-rate system. The output is generated at the sampling rate, which may or may not be the clock frequency. The rate at which the noise generator and the filter operate is much lower, depending on the actual Doppler rate, its actual value being the sampling rate divided by the interpolation factor. It is therefore natural to prefer a sequential architecture for the generator and the filter, which results in a very low-area realization without sacrificing performance.

In the field of **channel estimation**, we started with an overview of the main solutions proposed in the literature. Most of them are either unrealistically complex or do not offer sufficient performance. In this respect, the present thesis has two main contributions. On the one hand we showed how the performance of the channel estimator, represented by its gain, affects the overall receiver performance, expressed as the bit error rate. The analysis, performed experimentally by simulating a DVB-T receiver with a multi-path channel model, shows that an estimation gain of 6 dB is sufficient. Further increases of the gain, which can always be achieved with increased hardware complexity, are no longer accompanied by a corresponding increase in performance. The perfect estimator should achieve a gain of at least 6 dB over various operating conditions, with the lowest possible computational complexity.

On the other hand, we proposed a generic channel estimation architecture consisting of two separable polyphase filters. Their size and coefficients values depend on the operating conditions and the required gain. In this work we also present the design of polyphase interpolation and Wiener filters and a thorough analysis of their performance. For complexity reasons, the filter along the time axis has to be kept as short as possible. I many practical scenarios, a 2-tap linear interpolation filter will suffice. The entire gain will be achieved by the filter along the frequency axis, which is a Wiener filter. For optimum performance at various operating conditions (channel statistics) the coefficients should be selected adaptively from a precomputed set.

Further on, we dealt with with power and area optimizations in the implementation of the **Viterbi algorithm** for decoding convolutional codes. After a brief background on the Viterbi algorithm, we introduced a high-throughput state-parallel decoder architecture with adjustable trace-back length. The trace-back block is implemented as a pipeline whose length can be changed during operation by switching off unused stages, thus saving power dynamically. We also introduced an algorithm for selecting the minimum trace-back length which ensures a desired performance. Post-synthesis power estimation for standard cell libraries demonstrated power savings of up to 62%.

The other Viterbi decoder approach focused on minimizing the area, with emphasis on FPGA implementation. We proposed a lower-area state-serial architecture, for which we optimized all three building blocks of the decoder: the branch-metrics unit, the add-compare-select unit, and the trace-back unit. By exploiting architectural features of new FPGA's, particularly block RAM's and shift registers, we managed to develop a very area-efficient architecture. The design

is completely generic has been described in VHDL and synthesized using the Xilinx Spartan-3 FPGA family, using the convolutional code employed in the DAB and DRM standards. At a target clock frequency of around 150 MHz, the throughput exceeds the requirements of both standards. The whole design occupies less than 1/5 of the smallest FPGA device in the family. We argue that this is the most compact Viterbi decoder presented so far.

The next chapter dealt with the **simulation and design** of OFDM receivers. For simulation we employed untimed dataflow modeling in SystemC, with control tags accompanying data tokens. As a test case we considered a receiver for the IEEE 802.11a OFDM-based wireless LAN standard. Both individual samples and entire OFDM symbols were used as tokens, each solution having its advantages and disadvantages. The purpose of the above modeling is only algorithmic and architectural exploration. No high-level synthesis was envisaged.

On the design side, we considered the **Fast Fourier Transform**, which is the heart of any OFDM system. Various architectures were compared with respect to their area requirements and suitability for various OFDM standards. We showed that sequential architectures offer sufficient performance for implementing most OFDM broadcasting standards, even on FPGA. On the other hand, pipelined architectures offer a higher throughput, but at the cost of increased hardware resources. Of either family we implemented one architecture: a single-RAM sequential architecture and a R2^{2}SDF pipelined one. The designs have all parameters generic, e.g. data width and block size, and rely on VHDL features that promote reusability, such as static functions for computing the ROM twiddle factors. Moreover, we took advantage of the embedded RAM's and multipliers in the modern FPGA's. Implementation results were provided for the Xilinx Spartan-3 FPGA family.

8.2. Future Work

In the field of wireless channel simulators for hardware prototyping platforms, no significant improvements can be imagined for the solution proposed in this thesis, except for some optimizations. For instance, the design time of the spectrum-shaping filter cascade could be reduced by improving the filter design algorithm. The current algorithm becomes very slow when the spectrum is discretized into more than 512 points. A further point worth investigating is the optimal sizing of various parameters of the fading generators, depending on the desired simulation precision. Since the proposed design is completely generic, it can be used without any change for various parameter values.

For the proposed channel estimator architecture, a very promising direction for further research is the estimation of the Doppler and delay spreads. Once these estimates are known, the optimal sets of coefficients can be selected from a precomputed look-up table. Thus, the channel estimator gain is maximized by matching the estimator to the actual channel statistics, which is clearly superior to using a fixed set of coefficients. The challenge consists in performing the estimation in both directions (time and frequency) and selecting the best update rate for the

coefficients.

Likewise, the principle of estimation for dynamic configuration must also be applied in the case of the proposed Viterbi decoder architecture with variable decoding window length. In this case, the estimate is the bit error rate, the difference from the desired rate being used for adjusting the decoding window to the minimum length which still ensures an acceptable BER. The challenge is how to estimate the BER and how often to adjust the trace-back length. The solution of dynamically adjusting the decoding window has been proposed for a high-throughput state-parallel architecture, but it can be applied to the lower-area state-serial architecture as well.

Another research direction stems from the necessity of using a behavioral bit-true model for both system performance simulation and functional verification of the final HDL design. Two aspects are essential here. On the one hand, we need a standard interface for various processing blocks in order to be able to use a black box approach. The behavioral reference model, written in SystemC, should deal only with the actual data being transferred, without implementing any timing and protocol details. On the other hand, a generic mechanism is needed for passing parameters to the reference model and read status information from it. Access to parameters should be performed through names instead of actual addresses. A possible solution for developing a reusable framework is to use the transaction-level modeling (TLM 2.0) guidelines proposed by the Open SystemC Initiative industry group.

Appendix A.

LFSR Generator Polynomials

Table A.1 shows LFSR XOR taps that generate a maximum-length sequence, for LFSR sizes N between 2 and 168, using a Fibonacci configuration. The N and 0 taps have not been shown, since they are always present and require no XOR gates. Also note that the number of taps is always odd. The generator polynomials have the following form: for e.g. N = 16, $P(x) = x^{16} + x^{15} + x^{13} + x^4 + 1$.

Table A.1.: LFSR XOR taps for maximum sequence lengths

Size	**Taps**	**Size**	**Taps**	**Size**	**Taps**	**Size**	**Taps**
2	1	3	2	4	3	5	3
6	5	7	6	8	6, 5, 4	9	5
10	7	11	9	12	6, 4, 1	13	4, 3, 1
14	5, 3, 1	15	14	16	15, 13, 4	17	14
18	11	19	6, 2, 1	20	17	21	19
22	21	23	18	24	23, 22, 17	25	22
26	6, 2, 1	27	5, 2, 1	28	25	29	27
30	6, 4, 1	31	28	32	22, 2, 1	33	20
34	27, 2, 1	35	33	36	25	37	5, 4, 3, 2, 1
38	6, 5, 1	39	35	40	38, 21, 19	41	38
42	41, 20, 19	43	42, 38, 37	44	43, 18, 17	45	44, 42, 41
46	45, 26, 25	47	42	48	47, 21, 20	49	40
50	49, 24, 23	51	53, 36, 35	52	49	53	52, 38, 37
54	53, 18, 17	55	31	56	55, 35, 34	57	50
58	39	59	58, 38, 37	60	59	61	60, 46, 45
62	61, 6, 5	63	62	64	63, 61, 60	65	47
66	65, 57, 56	67	66, 58, 57	68	59	69	67, 42, 40
70	69, 55, 54	71	65	72	66, 25, 19	73	48
74	73, 59, 58	75	74, 65, 64	76	75, 41, 40	77	76, 47, 46

Continued on next page

Size	Taps	Size	Taps	Size	Taps	Size	Taps
78	77, 59, 58	79	70	80	79, 43, 42	81	77
82	79, 47, 44	83	82, 38, 37	84	71	85	84, 58, 57
86	85, 74, 73	87	74	88	87, 17, 16	89	51
90	89, 72, 71	91	90, 8, 7	92	91, 80, 79	93	91
94	73	95	84	96	94, 49, 47	97	91
98	87	99	97, 54, 52	100	63	101	100, 95, 94
102	101, 36, 35	103	94	104	103, 94, 93	105	89
106	91	107	105, 44, 42	108	77	109	108, 103, 102
110	109, 98, 97	111	101	112	110, 69, 67	113	104
114	113, 33, 32	115	114, 101, 100	116	115, 46, 45	117	115, 99, 97
118	85	119	111	120	113, 9, 2	121	103
122	121, 63, 62	123	121	124	87	125	124, 18, 17
126	125, 90, 89	127	126	128	126, 101, 99	129	124
130	127	131	130, 84, 83	132	103	133	132, 82, 81
134	77	135	124	136	135, 11, 10	137	116
138	137, 131, 130	139	136, 134, 131	140	111	141	140, 110, 109
142	121	143	142, 123, 122	144	143, 75, 74	145	93
146	145, 87, 86	147	146, 110, 109	148	121	149	148, 40, 39
150	97	151	148	152	151, 87, 86	153	152
154	152, 27, 25	155	154, 124, 123	156	155, 41, 40	157	156, 131, 130
158	157, 132, 131	159	128	160	159, 142, 141	161	143
162	161, 75, 74	163	162, 104, 103	164	163, 151, 150	165	164, 135, 134
166	165, 128, 127	167	161	168	166, 153, 151		

Appendix B.

Doppler Shaping Filter Coefficients

Table B.1 lists the coefficients of the second-order sections (SOS) for the Doppler spectrum shaping filters designed in **Chapter 4**.

Jakes Doppler filter with $f_D = 0.25$				
SOS stage	numerator coefficients		denominator coefficients	
	b_1	b_2	a_1	a_2
1	−1.35413771459409600	0.99841658177586590	−1.33857691521792140	0.75947473474759186
2	1.26747105260715510	0.78930450430435106	−1.39863709399956340	0.95531470097521487
3	−1.19866280663418420	0.99543248662376516	−1.40922791079687440	0.98532355397444626
4	−1.40515195731594190	0.99969218430916440	−1.16419423413520210	0.35943110404353640
5	−1.39133075718634310	0.99982725668539552	−1.41494205593041200	0.99942021194681541
6	−0.43676993048605334	0.80739524558126341	−1.37815037370731660	0.88972270416275967
7	−1.03573956156185280	0.95640749452201779	−1.26341848363859710	0.55436619990680991
Multiplicative constant K for normalized power: 0.00483235231292577				

Flat Doppler filter with $f_D = 0.25$				
SOS stage	numerator coefficients		denominator coefficients	
	b_1	b_2	a_1	a_2
1	−0.44457315808425180	0.91931225902709734	−1.16343308439212060	0.54647747776787436
2	−0.77446078634358861	0.98981368816063675	−1.35900986133933130	0.81085827290596257
3	−1.29047650959471660	0.99937410279524796	−1.40217900879071420	0.92768253583182181
4	−1.39310108524647510	0.99804262273733535	−0.98336636868427774	0.25196904166531714
5	1.45327883765238750	0.98785427545955384	−1.41698479882818010	0.97717524280876300
6	−1.28573046642038900	0.99400928613098838	−1.42328488912234350	0.99849382257743036
7	−1.41338687347567740	0.99983684800114370	−1.08072148362884970	0.45127169621342850
Multiplicative constant K for normalized power: 0.00530785604999349				

Gaussian Doppler filter with $\sigma_D = 0.25$				
SOS stage	numerator coefficients		denominator coefficients	
	b_1	b_2	a_1	a_2
1	1.65217016762262100	0.68633648261507685	−0.24847595973217060	0.06329494952095784
2	0.92873120055548264	0.29183328855866075	−0.41240362650379853	0.04470150937539093
Multiplicative constant K for normalized power: 0.14764710448880353				

Table B.1.: Biquad coefficients for the designed Doppler filters

Appendix C.

Polyphase Filters Design

Listing C.1 and **Listing C.2** show the Matlab source code of the functions used for designing Lagrange and windowed sinc interpolation filters respectively. Both functions take the number of polyphase filter taps and the number of phases as parameters. When the filters perform interpolation, the first parameter can be also regarded as the interpolation factor.

Listing C.1: Matlab function for computing Lagrange coefficients

```
function h = lagrangefilt(l,m)
% This function designs a Lagrange interpolation filter.
% Syntax:
%   H = LAGRANGEFILT(L,M)
% where:
%   L: interpolation factor (number of phases)
%   M: filter size (number of taps, must be even!)
%   H: polyphase coefficients vector (L*M-1)

t = 0:(m-1)*l+1;
l = ones(m,length(t));
for i = 0:m-1
  for j = 0:m-1
    if j ~= i
      l(i+1,:) = l(i+1,:) .* (t/l-j)/(i-j);
    end;
  end;
end;

h = zeros(1,m*l);
for i = 0:l-1
  for j = 0:m-1
    h(j*l+i+1) = l((m-j), l*(m-2)/2+i+1);
  end;
end;
```

Listing C.2: Matlab function for windowed sinc interpolation filters, e.g. Lanczos

```
function h = winsincfilt(l,m,w)
% This function designs a windowed sinc interpolation filter.
% Syntax:
%   H = WINSINCFILT(L,M,W)
% where:
%   L: interpolation factor (number of phases)
%   M: filter size (number of taps = 2*M)
%   W: window type ('lanczos','sinc')
%   H: polyphase coefficients vector (2*L*M-1)

% Generate vector of phases (2*l*m-1)
```

```
x = -m:1/l:+m;
x = x(2:end-1);
% Compute sinc coefficients
sincvec = sinc(x);
% Compute window
switch lower(w)
  case 'lanczos'
    win = sinc(x/m);
  case 'sinc'
    win = ones(size(x));
  otherwise
    error('Undefined window type');
end
% Apply window
h = sincvec .* win;
```

Listing C.3 shows the Matlab source code of the function used for designing interpolating and regular Wiener filters. The first parameter represents the number of phases and the the second one the filter size. For regular (non-interpolating) symmetrical Wiener filters, the number of phases must be one and the filter size odd. For interpolating Wiener filters, however, the number of taps must be even. The third parameter is either the signal bandwidth, assuming a rectangular spectrum, or the one-sided autocorrelation vector for which the filter is optimized. The forth parameter is optional and represents the variance of the additive noise for which the filter is optimized. If set to zero, a signal-matched interpolation filter will be designed. The computed coefficients are returned in a polyphase array.

Listing C.3: Matlab function for designing interpolating Wiener filters

```
function [h,gd] = wienerintfilt(varargin)
% This function designs an optimal polyphase interpolating Wiener filter.
% Syntax:
%   [H,GD] = WIENERINTFILT(L,M,BW)
%   [H,GD] = WIENERINTFILT(L,M,AC)
%   [H,GD] = WIENERINTFILT(L,M,BW,VAR)
%   [H,GD] = WIENERINTFILT(L,M,AC,VAR)
% where:
%   L: interpolation factor (number of phases)
%   M: filter size (number of taps)
%   BW: signal bandwidth (real 0...1)
%   AC: signal autocorrelation vector (of size M/2)
%   VAR: noise variance
%   H: polyphase coefficients array (L x M)
%   GD: group delay of the filter (samples)

if nargin == 3 || nargin == 4
  l = varargin{1};
  m = varargin{2};
  r = varargin{3};
  if nargin == 4
    var = varargin{4};
  else
    var = 0;
  end
else
  error('Wrong number of arguments.');
end

% Compute autocorrelation vector
if length(r) == 1 % scalar
  % parameter 3 is the signal bandwidth
  r = sinc((0:(m-1)*l)*r/l); % autocorrelation of a rectangular spectrum
else % vector
```

Table C.1.: Coefficients of a 4-tap Lagrange polyphase interpolation filter

Coeff	Phases								
	0/8	1/8	2/8	3/8	4/8	5/8	6/8	7/8	8/8
C0	0.0000	-0.0342	-0.0547	-0.0635	-0.0625	-0.0537	-0.0391	-0.0205	0.0000
C1	1.0000	0.9229	0.8203	0.6982	0.5625	0.4189	0.2734	0.1318	0.0000
C2	0.0000	0.1318	0.2734	0.4189	0.5625	0.6982	0.8203	0.9229	1.0000
C3	0.0000	-0.0205	-0.0391	-0.0537	-0.0625	-0.0635	-0.0547	-0.0342	0.0000

```
35    % parameter 3 is the signal autocorrelation vector
36    % check if vector has exactly L*(M-1)+1 elements
37    r = r(:)';
38    if length(r) < l*(m-1)+1 % append 0's if less
39      r = [r(:) zeros(l*(m-1)+1-length(r),1)];
40    else % truncate if more
41      r = r(1:l*m);
42    end
43  end
44
45  % Sample autocorrelation vector
46  r_vec_pos_idx = l*(0:m-1);
47  r_vec_pos = r(1+r_vec_pos_idx);
48  % Compute autocorrelation matrix
49  r_mat = toeplitz(r_vec_pos) + var*eye(m);
50
51  r_vec_sym_idx = -l*(m-1):l*(m-1);
52  r_vec_sym = [r(end:-1:2) r(1:end)];
53  gd = floor((l*m-1)/2); % group delay
54  r0_idx = l*(0:m-1)-gd; % indices of first slice
55  mid = length(r);
56
57  % compute polyphase coefficients
58  for phi = 0:l-1; % L phases
59    r_vec_sym_phi(phi+1,:) = r_vec_sym(mid+r0_idx+phi);
60    h(phi+1,:) = r_mat \ r_vec_sym_phi(phi+1,:)';
61  end
```

Table C.1, **Table C.2**, and **Table C.3** show the computed polyphase coefficients of an 8-phase Lagrange interpolation filter for 4, 6, and 8 taps respectively. It can be observed that the coefficients for phase n/8 are the coefficients for phase (8-n)/8 reversed and that the coefficients for phase 4/8 are symmetrical. Besides, the coefficients for phases 0/8 and 8/8 are degenerated so that the current and the next sample are produced.

Figure C.1 plots of the coefficients of an 8-tap 8-phase Lagrange filter (see **Table C.3**), as computed using the function in **Listing C.1**.

Listing C.4 shows the MEX C function which implements a multi-channel resampling routine. The function is compiled as a dynamic library and is subsequently called from Matlab. This reduce the simulation time significantly compared to interpreted Matlab code.

Listing C.4: C-MEX multichannel interpolation function for Matlab simulations

```
1  #include "mex.h"
2  #include <stdlib.h>
3  #include <string.h>
```

Table C.2.: Coefficients of a 6-tap Lagrange polyphase interpolation filter

Coeff	Phases								
	0/8	1/8	2/8	3/8	4/8	5/8	6/8	7/8	8/8
C0	0.0000	0.0055	0.0094	0.0115	0.0117	0.0104	0.0077	0.0041	0.0000
C1	0.0000	-0.0522	-0.0846	-0.0989	-0.0977	-0.0837	-0.0604	-0.0313	0.0000
C2	1.0000	0.9397	0.8459	0.7255	0.5859	0.4353	0.2820	0.1342	0.0000
C3	0.0000	0.1342	0.2820	0.4353	0.5859	0.7255	0.8459	0.9397	1.0000
C4	0.0000	-0.0313	-0.0604	-0.0837	-0.0977	-0.0989	-0.0846	-0.0522	0.0000
C5	0.0000	0.0041	0.0077	0.0104	0.0117	0.0115	0.0094	0.0055	0.0000

Table C.3.: Coefficients of an 8-tap Lagrange polyphase interpolation filter

Coeff	Phases								
	0/8	1/8	2/8	3/8	4/8	5/8	6/8	7/8	8/8
C0	0.0000	-0.0011	-0.0019	-0.0023	-0.0024	-0.0022	-0.0016	-0.0009	0.0000
C1	0.0000	0.0112	0.0191	0.0234	0.0239	0.0211	0.0156	0.0082	0.0000
C2	0.0000	-0.0632	-0.1031	-0.1210	-0.1196	-0.1024	-0.0736	-0.0379	0.0000
C3	1.0000	0.9482	0.8592	0.7397	0.5981	0.4438	0.2864	0.1355	0.0000
C4	0.0000	0.1355	0.2864	0.4438	0.5981	0.7397	0.8592	0.9482	1.0000
C5	0.0000	-0.0379	-0.0736	-0.1024	-0.1196	-0.1210	-0.1031	-0.0632	0.0000
C6	0.0000	0.0082	0.0156	0.0211	0.0239	0.0234	0.0191	0.0112	0.0000
C7	0.0000	-0.0009	-0.0016	-0.0022	-0.0024	-0.0023	-0.0019	-0.0011	0.0000

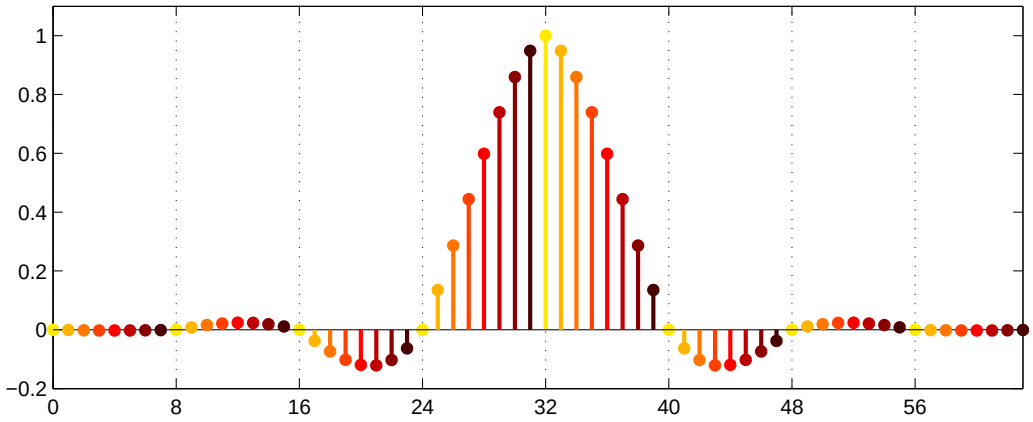

Figure C.1.: Coefficients of an 8-tap 8-phase Lagrange interpolation filter

```
4  #include <math.h>
5
6  void resample(
7    double *din,   /* input samples vector */
8    int nis,       /* number of input samples */
9    int nch,       /* number of parallel data channels */
10   double *dout,  /* output samples vector */
11   int nos,       /* number of output samples */
12   double *coeff, /* coefficients array (nph+1 x nft) */
13   int nph,       /* number of phases */
14   int nft,       /* number of filter taps */
15   double step,   /* resampler step */
16   double *state, /* initial and final state */
17   double *phase) /* initial and intermediate state */
18 {
19   int i, j, k;        /* iterators */
20   int iidx;           /* input data index */
21   double currPhase;   /* current phase */
22   double nextPhase;   /* next output sample phase */
23   double subsPhase;   /* sub-sample phase */
24   int    segmIndex;   /* segment index */
25   double segmPhase;   /* segment phase (within a phase segment) */
26   double* intCoeff;   /* interpolated filter coefficient */
27   int    coeffIdx;    /* linear index in the coefficients table */
28   int    inSamInc;    /* input samples increment */
29
30   intCoeff = calloc(nft, sizeof(double));
31
32   iidx = 0;
33   for (i = 0; i < nos; i = i+1) {
34     currPhase = phase[0] + step*i;
35     nextPhase = phase[0] + step*(i+1);
36     subsPhase = currPhase - floor(currPhase);
37     segmIndex = (int)floor(subsPhase*nph);
38     segmPhase = subsPhase*nph - segmIndex;
39     /* compute coefficients by linear interpolation */
40     /* compute interpolated sample */
41     for (j = 0; j < nft; j = j+1) {
42       coeffIdx = j*(nph+1)+segmIndex;
43       intCoeff[j] = coeff[coeffIdx] + segmPhase * (coeff[coeffIdx+1] - coeff[coeffIdx]);
44     }
45     /* perform interpolation, for all data channels */
46     for (k = 0; k < nch; k = k+1) {
47       dout[k*nos+i] = 0;
48       for (j = 0; j < nft; j = j+1) {
49         dout[k*nos+i] += intCoeff[j] * state[k*nft+j];
50       }
51     }
52     /* update states and input index if necessary */
53     inSamInc = floor(nextPhase)-floor(currPhase);
54     while (inSamInc--) {
55       /* for all data channels */
56       for (k = 0; k < nch; k = k+1) {
57         for (j = nft-1; j > 0; j = j-1) {
58           state[k*nft+j] = state[k*nft+j-1];
59         }
60         /* pad with zeros if we exceed input vector */
61         state[k*nft+0] = (iidx < nis) ? din[k*nis+iidx] : 0;
62       }
63       iidx = iidx+1;
64     }
65   }
66   phase[0] = (phase[0] + nos*step) - floor(phase[0] + nos*step);
67
68   free(intCoeff);
69 }
70
```

```
71  /* The gateway routine */
72  void mexFunction(
73    int nlhs,
74    mxArray *plhs[],
75    int nrhs,
76    const mxArray *prhs[])
77  {
78    double *din;          /* input samples vector */
79    int nis;              /* number of input samples */
80    int nch;              /* number of parallel data channels */
81    int nos;              /* number of output samples */
82    double *coeff;        /* coefficients array */
83    int nph;              /* number of phases */
84    int nft;              /* number of filter taps */
85    double step;          /* resampling step */
86    double *istate;       /* initial filter state */
87    double *iphase;       /* initial phase */
88
89    double *dout;         /* output data */
90    double *state;        /* final filter state */
91    double *phase;        /* final phase */
92
93    int mSize, nSize;     /* size of the input vector */
94    int i;                /* iterator */
95
96    if (nrhs < 4 || nrhs > 6) {
97      mexErrMsgTxt("This function accepts 4 to 6 inputs");
98    }
99    if (nlhs < 1 || nlhs > 3) {
100     mexErrMsgTxt("This function accepts 1 to 3 outputs");
101   }
102   /* Input 1: input samples array */
103   if (!mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]))
104     mexErrMsgTxt("Input 1 (input samples) must be an array of double.");
105   din = mxGetPr(prhs[0]);
106   nis = mxGetM(prhs[0]);
107   nch = mxGetN(prhs[0]);
108   /* Input 2: number of output samples */
109   if (!mxIsDouble(prhs[1]) || mxIsComplex(prhs[1]) || mxGetM(prhs[1])*mxGetN(prhs[1]) != 1)
110     mexErrMsgTxt("Input 2 (number of output samples) must be a real scalar.");
111   nos = (int)floor(mxGetScalar(prhs[1]));
112   /* Input 3: coefficients array */
113   if (!mxIsDouble(prhs[2]) || mxIsComplex(prhs[2]) || mxGetM(prhs[2]) == 1)
114     mexErrMsgTxt("Input 3 (coefficients array) must be an array of double.");
115   coeff = mxGetPr(prhs[2]);
116   nph = mxGetM(prhs[2])-1;
117   nft = mxGetN(prhs[2]);
118   /* Input 4: resampling step */
119   if (!mxIsDouble(prhs[3]) || mxIsComplex(prhs[3]) || mxGetM(prhs[3])*mxGetN(prhs[3]) != 1)
120     mexErrMsgTxt("Input 4 (resampling step) must be a real scalar.");
121   step = mxGetScalar(prhs[3]);
122   /* Input 5:  filter state */
123   if (nrhs > 4) {
124     if (!mxIsDouble(prhs[4]) || mxIsComplex(prhs[4]))
125       mexErrMsgTxt("Input 5 (filter state) must be an array of double.");
126     if (nft != mxGetM(prhs[4]))
127       mexErrMsgTxt("Number of rows in state array must be equal with the number of taps");
128     if (nch != mxGetN(prhs[4]))
129       mexErrMsgTxt("Number of columns in state array must be equal with the number of channels");
130     istate = mxGetPr(prhs[4]);
131   }
132   /* Input 6: initial phase */
133   if (nrhs > 5) {
134     if (!mxIsDouble(prhs[5]) || mxIsComplex(prhs[5]) || mxGetM(prhs[5])*mxGetN(prhs[5]) != 1)
135       mexErrMsgTxt("Input 5 (initial phase) must be a real scalar.");
136     iphase = mxGetPr(prhs[5]);
137   }
```

```
138
139   /* Output 1: output samples array */
140   plhs[0] = mxCreateDoubleMatrix(nos, nch, mxREAL);
141   dout = mxGetPr(plhs[0]);
142   /* Output 2: final state vector */
143   plhs[1] = mxCreateDoubleMatrix(nft, nch, mxREAL);
144   state = mxGetPr(plhs[1]);
145   /* Output 3: final phase */
146   plhs[2] = mxCreateDoubleMatrix(1, 1, mxREAL);
147   phase = mxGetPr(plhs[2]);
148
149   /* Copy state and phase */
150   for (i = 0; i < nft*nch; i = i+1) {
151     state[i] = (nrhs > 4) ? istate[i] : 0;
152   }
153   phase[0] = (nrhs > 5) ? iphase[0] : 0;
154
155   /* call the actual resampling routine */
156   resample(din, nis, nch, dout, nos, coeff, nph, nft, step, state, phase);
157 }
```

Appendix D.

Disk Contents

This appendix describes the contents of the disk that accompanies the present dissertation. The disk contains the following directories:

- `Biblio`: cited articles, in PDF format, including the referenced IEEE and ETSI standards
- `Code`: C/C++/SystemC, Matlab, and VHDL code produced during this research work
- `Papers`: published conference papers
- `Thesis`: publication version of the thesis in PDF format and defense presentation in Powerpoint format
- `Students`: supervised student theses
- `Visio`: drawings for the dissertation in Visio format

The subdirectory directory `Code/C` contains the following: 1) C++ code for the modeling of the Doppler taps generators used in the channel simulator Channel simulator, and 2) SystemC code for the simulation of the Viterbi decoder. The subdirectory `Code/Matlab` contains classes and functions used for various simulations throughout this dissertation. Because of the object-oriented features employed, at least Matlab 2007a is required in order to execute the code. The subdirectory `Code/VHDL` contains VHDL code for the hardware modeling of the following blocks: CORDIC rotator, FFT (pipelined and sequential), and Viterbi decoder (state serial and parallel), as well as various other small blocks, such as LFSR, FIFO, sine ROM, etc.

References

[1] P. A. Bello. Characterization of randomly time-variant linear channels. *IEEE Trans. on Communications*, 11:360–393, Dec. 1963.

[2] E. Bidet, D. Castelain, C. Joanblanq, and P. Senn. A fast single-chip implementation of 8192 complex point FFT. *IEEE Journal of Solid-State Circuits*, 30(3):300–305, Mar. 1995.

[3] J. A. C. Bingham. Multicarrier modulation for data transmission: An idea whose time has come. *IEEE Communications Magazine*, 28(5):5–14, May 1990.

[4] E. Boutillon, J.-L. Danger, and A. Ghazel. Design of high speed AWGN communication channel emulator. *Analog Integrated Circuits and Signal Processing*, 34(2):133–142, Feb. 2003.

[5] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *Annals Math. and Statistics*, 29:610–611, 1958.

[6] P. P. Chu and R. E. Jones. Design techniques of FPGA based random number generator. In *Proc. Military and Aerospace Applications of Programming Devices and Techniques Conf.*, 1999.

[7] L. J. Cimini Jr. Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing. *IEEE Trans. on Communications*, COMM-33(7):665–675, July 1985.

[8] R. H. Clark. A statistical theory of mobile reception. *BSJT*, 49:957–1000, 1968.

[9] F. Classen. *Systemkomponenten für eine terrestrische digitale mobile Breitbandübertragung.* PhD dissertation, RWTH-Aachen, Germany, 1996. In German.

[10] F. Classen, M. Classen, and H. Meyr. Channel estimation units for an OFDM system suitable for mobile communication. In *ITG-Fachbericht: Mobile Kommunikation*, 1995.

[11] F. Classen and H. Meyr. Frequency synchronization algorithms for OFDM systems suitable for communications over frequency selective fading channels. In *Proc. Vehicular Technology Conf. (VTC)*, volume 3, pages 1655–1659, 1994.

[12] D. Cohen. Simplified control of FFT hardware. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 24(6):577–579, Dec. 1976.

[13] S. Coleri, M. Ergen, A. Puri, and A. Bahai. Channel estimation techniques based on pilot arrangement in ofdm systems. *IEEE Trans. on Broadcasting*, 48(3):223–229, Sept. 2002.

[14] R. E. Crochiere and L. R. Rabiner. Interpolation and decimation of digital signals - a tutorial review. *Proceedings of the IEEE*, 69(3):300–331, 1981.

[15] P. Dent, G. E. Bottomley, and T. Croft. Jakes fading model revisited. *Electronics Letters*, 29(13):1162–1163, June 1993.

[16] A. Dowler, A. Doufexi, and A. Nix. Performance evaluation of channel estimation techniques for a mobile fourth generation wide area OFDM system. In *Proc. Vehicular Technology Conf. (VTC)*, volume 4, pages 2036–2040, 2002.

[17] M. F. (ed.). COST 207: Digital land mobile radio communications, final report. Technical report, European Commision, Brussels, Belgium, 1989.

[18] O. EDFORS, M. SANDELL, J.-J. VAN DE BEEK, S. K. WILSON, and P. O. BÖRJESSON. OFDM channel estimation by singular value decomposition. *IEEE Trans. on Communications*, 46(7):931–939, July 1998.

[19] ETSI. *EN 300 401 v.1.3.3. Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers*, May 2001.

[20] ETSI. *EN 300 744 v.1.5.1. Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, Nov. 2004.

[21] ETSI. *ES 201 980 v.2.1.1. Digital Radio Mondiale (DRM); System specifications*, Apr. 2004.

[22] C. W. FARROW. A continuously variable digital delay element. In *Proc. Intl. Symp. on Circuits and Systems (ISCAS)*, pages 2641–2645, 1988.

[23] G. D. FORNEY. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):218–278, Mar. 1973.

[24] W. N. FURMAN and J. W. NIETO. Understanding HF channel simulator requirements in order to reduce HF modem performance measurement variability. In *Proc. Nordic Shortwave Conf.* Harris Corporation, 2001.

[25] A. GHAZEL, E. BOUTILLON, J. DANGER, G. GULAK, and H. LAAMARI. Design and performance analysis of high speed AWGN communication channel emulator. In *Proc. IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing (PACRIM)*, Victoria, B.C., Aug. 2001.

[26] M. GOEL and N. R. SHANBHAG. Low-power channel coding via dynamic reconfiguration. In *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASP)*, pages 1893–1896, 1999.

[27] T. GROETKER, S. LIAO, G. MARTIN, and S. SWAN. *System Dessign with SystemC*. Kluwer Academic Publishers, 2003.

[28] M. HASAN and T. ARSLAN. Scheme for reducing size of coefficient memory in FFT processor. *IEEE Electronics Letters*, 38(4):163–164, Feb. 2002.

[29] S. HE and M. TORKELSON. A new approach to pipeline FFT processor. In *Proc. Intl. Parallel Processing Symp. (IPPS)*, pages 766–770, 1996.

[30] S. HE and M. TORKELSON. Designing pipeline FFT processor for OFDM (de)modulation. In *Proc. URSI Intl. Symp. on Signals, Systems, and Electronics (ISSSE)*, pages 257–262, 1998.

[31] A. P. HEKSTRA. An alternative to metric rescaling in Viterbi decoders. *IEEE Trans. on Communications*, 37(3):1220–1222, Nov. 1989.

[32] R. HENNING and C. CHAKRABARTI. Low-power approach for decoding convolutional codes with adaptive Viterbi algorithm approximations. In *Proc. Intl. Symp. on Low Power Electronics and Design (ISLPED)*, pages 68–71, 2002.

[33] P. HOEHER. A statistical discrete-time model for the WSSUS multipath channel. *IEEE Trans. on Vehicular Technology*, 41(4):461–468, 1992.

[34] P. HOEHER, S. KAISER, and P. ROBERTSON. Two-dimensional pilot-symbol-aided channel estimation by wiener filtering. In *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASP)*, volume 3, pages 1845–1848, 1997.

[35] P. HÖHER. TCM on frequency-selective land-mobile fading channel. In *Proc. of Tirrenia Int. Workshop on Digital Communications*, 1991.

[36] M.-H. HSIEH and C.-H. WEI. Channel estimation for OFDM systems based on comb-type pilot arrangement in frequency-selective fading channels. *IEEE Trans. on Consumer Electronics*, 44(1):217–225, Feb. 1998.

[37] C.-P. HUNG, S.-G. CHEN, and K.-L. CHEN. Design of an efficient variable-length FFT processor. In *Proc. Intl. Symp. on Circuits and Systems (ISCAS)*, volume 2, pages 833–836, 2004.

[38] IEEE. *Wireless LAN Medium Access Control and Physical Layer Specifications; High-Speed Physical Layer in the 5 GHz Band*, 1999.

[39] IEEE. *Wireless MAN Air Interface for Fixed Broadband Wireless Access Systems*, 2004.

[40] M. ISAKA and H. IMAI. On the iterative decoding of multilevel codes. *IEEE Journal on Selected Areas in Communications*, 19(5):935–943, May 2001.

[41] ITU. CCIR recommendation 520-1, use of HF ionospheric channel simulators. *Recommendations and Reports of the CCIR*, 3:57–58, 1986.

[42] ITU. CCIR report 549-2, HF ionospheric channel simulators. *Recommendations and Reports of the CCIR*, 3:59–67, 1986.

[43] W. C. JAKES, ed. *Microwave Mobile Communications.* John Wiley Sons, 1974.

[44] A. JANTSCH. *Modeling Embedded Systems and SOC's.* Morgan Kaufmann Publishers, 2004.

[45] L. JOHNSON. Conflict free memory addressing for dedicated FFT hardware. *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, 39(5):312–316, May 1992.

[46] Y. JUNG, H. YOON, and J. KIM. New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications. *IEEE Trans. on Consumer Electronics*, 49(1):14–20, Feb. 2003.

[47] G. KAHN. The semantics of a simple language for parallel programming. In J. L. ROSENFELD, ed., *Information Processing*, pages 46–77. North-Holland Publishing Company, 1974.

[48] I. KANG and A. N. WILLSON. Low-power viterbi decoder for CDMA mobile terminals. *IEEE Journal of Solid-State Circuits*, 33(3):473–482, Mar. 1998.

[49] W. R. KNIGHT and R. KAISER. A simple fixed-point error bound for the fast Fourier transform. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 27(6):615–620, Dec. 1979.

[50] D. E. KNUTH. *The Art of Computer Programming*, volume 2, chapter Seminumerical Algorithms. Addison-Wesley, 3 edition, 1997.

[51] C. KOMNINAKIS. A fast and accurate rayleigh fading simulator. *Proc. IEEE Global Telecommunications Conf. (GLOBECOM)*, 6:3306–3310, Dec. 2003.

[52] D.-U. LEE, W. LUK, J. D. VILLASENOR, and P. Y. K. CHEUNG. A gaussian noise generator for hardware-based simulations. *IEEE Trans. on Computers*, 53(12):1523–1534, Dec. 2004.

[53] E. A. LEE and D. G. MESSERSCHMITT. Synchronous dataflow. In *Proceedings of the IEEE*, volume 75, pages 1235–1245, Sept. 1987.

[54] Y. LI. Pilot-symbol-aided channel estimation for OFDM in wireless systems. *IEEE Trans. on Vehicular Technology*, 49(4):1207–1215, July 2000.

[55] Y. LI, L. J. CIMINI JR., and N. R. SOLLENBERGER. Robust channel estimation for OFDM systems with rapid dispersive fading channels. *IEEE Trans. on Communications*, 46(7):902–915, July 1998.

[56] H. LIM and E. E. SWARTZLANDER. Multidimensional systolic arrays for the implementation of discrete Fourier transforms. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 47(5):1359–1370, May 1999.

[57] H.-F. LO, M.-D. SHIEH, and C.-M. WU. Design of an efficient FFT processor for DAB system. In *Proc. Intl. Symp. on Circuits and Systems (ISCAS)*, volume 4, pages 654–657, 2001.

[58] Y. MA. A VLSI-oriented parallel FFT algorithm. *IEEE Trans. on Signal Processing*, 44(2):445–

448, Feb. 1996.

[59] Y. Ma. An effective memory addressing scheme for FFT processors. *IEEE Trans. on Signal Processing*, 47(3):907–911, Mar. 1999.

[60] Y. Ma and L. Wanhammar. A hardware efficient control of memory addressing for high-performance FFT processors. *IEEE Trans. on Signal Processing*, 48(3):917–921, Mar. 2000.

[61] G. Marsaglia and W. W. Tsang. The ziggurat method for generating random variables. *Journal of Statistical Software*, 5(8):1–7, 2000.

[62] R. M. Mersereau and T. C. Speake. The processing of periodically sampled multidimensional signals. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-31(1):188–194, Feb. 1983.

[63] M. Morelli and U. Mengali. A comparison of pilot-aided channel estimation methods for OFDM systems. *IEEE Trans. on Signal Processing*, 49(12):3065–3073, Dec. 2001.

[64] R. Negi and J. Cioffi. Pilot tone selection for channel estimation in a mobile OFDM system. *IEEE Trans. on Consumer Electronics*, 44(3):1122–1128, Aug. 1998.

[65] A. M. Obeid, A. Garcia, M. Petrov, and M. Glesner. A multi-path high speed Viterbi decoder. In *Proc. Intl. Conf. on Electronics, Circuits, and Systams (ICECS)*, volume 3, pages 1160– 1163, Dec. 2003.

[66] G. Oetken, T. W. Parks, and H. W. Schüssler. New results in the design of digital interpolators. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 23(3):301–309, June 1975.

[67] OSCI. *Functional Specification for SystemC 2.0*, Apr. 2002.

[68] OSCI. *SystemC 2.0, User's Guide*, Apr. 2002.

[69] C. Park, J. Jung, and S. Ha. Extended synchronous dataflow for efficient DSP system prototyping. In *Proc. Design Automation and Test in Europe (DATE)*, volume 6, pages 295–322, Mar. 2002.

[70] M.-Y. Park, W.-C. Lee, J.-H. Kwak, C.-H. Cho, and H.-M. Park. A demapping method using the pilots in COFDM system. *IEEE Trans. on Consumer Electronics*, 44(3):1150–153, Aug. 1998.

[71] S. Y. Park, N. I. Cho, S. U. Lee, K. Kim, and J. Oh. Design of 2k/4k/8k-point FFT processor based on CORDIC algorithm in OFDM receiver. In *Proc. IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing (PACRIM)*, volume 2, pages 457–460, 2001.

[72] M. Pätzold, ed. *Mobile Fading Channels*. John Wiley Sons, 2002.

[73] M. C. Pease. Organization of large scale Fourier processors. *Journal of the ACM*, 16(3):474–482, July 1969.

[74] M. Petrov, A. Obeid, and T. Murgan. An adaptive trace-back solution for state-parallel viterbi decoders. In *Proc. IFIP Intl. Conf. on VLSI (VLSI-SOC)*, 2003.

[75] M. F. Pop and N. C. Beaulieu. Design of wide-sense stationary sum-of-sinusoids fading channel simulators. In *Proc. IEEE Intl. Conf. on Communications (ICC)*, volume 2, pages 709–716, Apr. 2002.

[76] ptolemy.eecs.berkeley.edu. *The Ptolemy Project Website*. UC Berkeley, EECS, Last visited in 2006.

[77] R. Pyndiah, A. Picart, and A. Glavieux. Performance of block turbo coded 16-QAM and 64-QAM modulations. In *Proc. IEEE Global Telecommunications Conf. (GLOBECOM)*, volume 2, pages 1039–1043, 1995.

[78] C. M. RADER. Memory management in a Viterbi decoder. *IEEE Trans. on Communications*, 29(9):1399–1401, Sept. 1981.

[79] J. RINNE and M. RENFORS. Pilot spacing in orthogonal frequency division multiplexing systems on practical channels. *IEEE Trans. on Consumer Electronics*, 42(4):959–962, Nov. 1996.

[80] F. SANZI and J. SPEIDEL. An adaptive two-dimensional channel estimator for wireless OFDM with application to mobile DVB-T. *IEEE Trans. on Broadcasting*, 46(2):128–133, June 2000.

[81] H. SARI, G. KARAM, and I. JEANCLAUDE. Transmission techniques for digital terrestrial tv broadcasting. *IEEE Communications Magazine*, 33(2):100–109, Feb. 1995.

[82] D. SCHAFHUBER, G. MATZ, and F. HLAWATSCH. Simulation of wideband mobile radio channels using subsampled ARMA models and multistage interpolation. *IEEE-SP Workshop on Statistical Signal Processing*, 6:571–574, Aug. 2001.

[83] D. SCHAFHUBER, G. MATZ, and F. HLAWATSCH. Adaptive prediction of time-varying channels for coded OFDM systems. In *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASP)*, volume 3, pages 2549–2552, 2002.

[84] C. SGRAJA and J. LINDNER. Estimation of rapid time-variant channels for OFDM using wiener filtering. In *Proc. IEEE Intl. Conf. on Communications (ICC)*, volume 4, pages 2390–2395, May 2003.

[85] M. SPETH, S. A. FECHTEL, G. FOCK, and H. MEYR. Optimum receiver design for wireless broadband systems using OFDM - part 1. *IEEE Trans. on Communications*, 47(11):1668–1677, Nov. 1999.

[86] M. SPETH, S. A. FECHTEL, G. FOCK, and H. MEYR. Optimum receiver design for wireless broadband systems using OFDM - part 2. *IEEE Trans. on Communications*, 49(4):571–578, Apr. 2001.

[87] B. STANTCHEV and G. FETTWEIS. Time-variant distortions in OFDM. *IEEE Communications Letters*, 4(10):312–314, Oct. 2000.

[88] K. STEIGLITZ. Computer-aided design of recursive digital filters. *IEEE Trans. on Audio and Electroacoustics*, 18:123–129, June 1970.

[89] F. TOSATO and P. BISAGLIA. Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2. In *Proc. IEEE Intl. Conf. on Communications (ICC)*, volume 2, pages 664– 668, 2002.

[90] T. K. TRUONG, M. SHIH, I. S. REED, and E. H. SATORIUS. A VLSI design for a trace-back Viterbi decoder. *IEEE Trans. on Communications*, 40(3):616–624, Mar. 1992.

[91] P. P. VAIDYANATHAN. *Handbook for Digital Signal Processing, editors S. K. Mitra and J. F. Kaiser*, chapter Robust Digital Filter Structures. John Wiley & Sons, 1993.

[92] J.-J. VAN DE BEEK, O. EDFORS, M. SANDELL, S. K. WILSON, and P. O. BOERJESSON. On channel estimation in OFDM systems. In *Proc. Vehicular Technology Conf. (VTC)*, volume 2, pages 815–819, July 1995.

[93] A. J. VITERBI. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Information Theory*, 13(2):260– 269, Apr. 1967.

[94] C. C. WATTERSON, J. R. JUROSHEK, and W. D. BENSEMA. Experimental confirmation of an HF channel model. *IEEE Trans. on Communications*, COMM-18(6):792–803, Dec. 1970.

[95] S. WEINSTEIN and P. M. EBERT. Data transmission by frequency-division multiplexing using the discrete Fourier transform. *IEEE Trans. on Communications*, COMM-19(5):628–634, Oct.

1971.

[96] P. D. WELCH. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Trans. on Audio and Electroacoustics*, 15:70–73, June 1967.

[97] Xilinx, Inc. *Additive White Gaussian Noise (AWGN) Core*, Oct. 2002. Product Specification.

[98] Xilinx, Inc. *Linear Feedback Shift Register v3.0*, Mar. 2003. Product Specification.

[99] B. YANG, K. B. LETAIEF, R. S. CHENG, and Z. CAO. Windowed DFT based pilot-symbol-aided channel estimation for OFDM systems in multipath fading channels. In *Proc. Vehicular Technology Conf. (VTC)*, volume 2, pages 1480–1484, 2000.

[100] B. YANG, K. B. LETAIEF, R. S. CHENG, and Z. CAO. Channel estimation for OFDM transmission in multipath fading channels based on parametric channel modeling. *IEEE Trans. on Communications*, 49(3):467–479, Mar. 2001.

[101] D. J. YOUNG and N. C. BEAULIEU. On the generation of correlated rayleigh random variables by inverse discrete fourier transform. In *Proc. of the Intl. Conf. on Universal Personal Communications (ICUPC)*, volume 1, pages 231–235, Cambridge, MA, Sept. 1996.

[102] Y. R. ZHENG and C. XIAO. Simulation models with correct statistical properties for rayleigh fading channels. *IEEE Trans. on Communications*, 51(6):920–928, June 2003.

List of Publications

[103] T. Murgan, A. García Ortiz, M. Petrov, and M. Glesner. A stochastic framework for communication architecture evaluation in networks-on-chip. In *Proc. IEEE Intl. Symp. on Signal, Circuit and Systems (ISSCS)*, volume 1, pages 253–256, Iasi, Romania, July 2003.

[104] T. Murgan, A. García Ortiz, M. Petrov, and M. Glesner. A linear model for high-level delay estimation in VDSM on-chip interconnects. In *Proc. Intl. Symp. on Circuits and Systems (ISCAS)*, Kobe, Japan, May 2005.

[105] T. Murgan, A. García Ortiz, C. Schlachta, H. Zimmer, M. Petrov, and M. Glesner. On timing and power consumption in inductively coupled on-chip interconnects. In *Proc. Intl. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, volume 3254, pages 819–828, Santorini, Greece, Sept. 2004.

[106] T. Murgan, M. Petrov, A. García Ortiz, R. Ludewig, P. Zipf, T. Hollstein, M. Glesner, B. Oelkrug, and J. Brakensiek. Evaluation and run-time optimisation of on-chip communication structures in reconfigurable architectures. In *Proc. Intl. Conf. on Field Programmable Logic and Applications (FPLA)*, pages 1111–1114, Lisbon, Portugal, Sept. 2003.

[107] T. Murgan, M. Petrov, M. Majer, P. Zipf, M. Glesner, and U. Heinkel. Adaptive architectures for an OTN processor: Reducing design costs through reconfigurability and multiprocessing. In *Proc. of the ACM Computing Frontiers Conference*, pages 408–414, Ischia, Italy, Apr. 2004.

[108] T. Murgan, M. Petrov, M. Majer, P. Zipf, M. Glesner, and U. Heinkel. Flexible overhead processing architectures for G.709 optical transport networks. In *GI/ITG/GMM Workshop on "Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen"*, Kaiserslautern, Germany, Feb. 2004.

[109] T. Murgan, C. Schlachta, M. Petrov, L. I. A. García Ortiz, M. Glesner, and R. Reis. Accurate capture of timing parameters in inductively-coupled on-chip interconnects. In *Proc. Intl. Symp. on Integrated Circuits and Systems Design (SBCCI)*, pages 117–122, Porto de Galinhas, Pernambuco, Brazil, Sept. 2004.

[110] A. M. Obeid, A. García Ortiz, M. Petrov, and M. Glesner. A multi-path high speed Viterbi decoder. In *Proc. Intl. Conf. on Electronics, Circuits, and Systams (ICECS)*, volume 3, pages 1160–1163, Darmstadt, Germany, Dec. 2003.

[111] S. Pandey, P. Zipf, O. Soffke, M. Petrov, T. Murgan, and M. Glesner. An infrastructure for distributed computing and context aware computing. In *Proc. of Intl. Conf. on Ubiquitous Computing*, Seattle, USA, Oct. 2003.

[112] M. Petrov. A scalable delay element for efficient resampling. *Accepted for publication to the IEEE Global Communications Conf. (GLOBECOM)*, 2007.

[113] M. Petrov and M. Glesner. Optimal FFT architecture selection for OFDM receivers on FPGA. In *Proc. Conf. on Field Programmable Technology (FPT)*, pages 313–314, Singapore, Dec. 2005.

[114] M. PETROV and M. GLESNER. A state-serial Viterbi decoder architecture for digital radio on FPGA. In *Proc. Conf. on Field Programmable Technology (FPT)*, pages 323–324, Singapore, Dec. 2005.

[115] M. PETROV and M. GLESNER. An efficient fractional-rate interpolation architecture. In *Accepted for publication to the IEEE Global Communications Conf. (GLOBECOM)*, 2007.

[116] M. PETROV and M. GLESNER. Filter design for Doppler spectrum shaping. In *Accepted for publication at the IEEE Intl. Conf. on Communications (ICC)*, 2007.

[117] M. PETROV and M. GLESNER. A scalable and hardware-efficient architecture for white Gaussian noise generation. In *Accepted for publication at the IEEE Intl. Conf. on Communications (ICC)*, 2007.

[118] M. PETROV, T. MURGAN, F. MAY, M. VORBACH, P. ZIPF, and M. GLESNER. The XPP architecture and its co-simulation within the Simulink environment. In *Proc. Intl. Conf. on Field Programmable Logic and Applications (FPLA)*, pages 761–770, Antwerpen, Belgium, Aug. 2004.

[119] M. PETROV, T. MURGAN, A. OBEID, C. CHIŢU, P. ZIPF, J. BRAKENSIEK, and M. GLESNER. Dynamic power optimization of the trace-back process for the Viterbi algorithm. In *Proc. Intl. Symp. on Circuits and Systems (ISCAS)*, volume 2, pages 721–724, Vancouver, Canada, May 2004.

[120] M. PETROV, T. MURGAN, P. ZIPF, and M. GLESNER. Functional modeling techniques for a wireless LAN OFDM transceiver. In *Proc. Intl. Symp. on Circuits and Systems (ISCAS)*, Kobe, Japan, May 2005.

[121] M. PETROV, A. OBEID, T. MURGAN, P. ZIPF, J. BRAKENSIEK, B. ÖLKRUG, and M. GLESNER. An adaptive trace-back solution for state-parallel Viterbi decoders. In *Proc. IFIP Intl. Conf. on VLSI (VLSI-SOC)*, pages 167–172, Darmstadt, Germany, Dec. 2003.

Supervised Theses

[122] Z. Dai. Entwurf einer ATA-Schnittstelle für den Wishbone-Bus als synthetisierbares SystemC-Modell. Studienarbeit, Technische Universität Darmstadt, Nov. 2003. In German.

[123] T. Geisbüsch. On-Chip-Bus Vergleich und parametrisierbare Beispiel-Anwendung. Studienarbeit, Technische Universität Darmstadt, Jan. 2005. In German.

[124] J. Giegerich. Design, simulation and synthesis of a generic FFT processor. Studienarbeit, Technische Universität Darmstadt, Oct. 2005. In German.

[125] M. Hicham. Untersuchung eines Kanal-Dekoders fr die OFDM-Übertragung. Studienarbeit, Technische Universität Darmstadt, Feb. 2005. In German.

[126] R. Methuku. Design & FPGA implementation of a multi-core architecture for ITU-T G.709 overhead processing. Master's thesis, Technische Universität Darmstadt, Dec. 2004.

[127] V. B. Nguyen. Algorythms for channel estimation in OFDM digital broadcasting receivers. Master's thesis, Technische Universität Darmstadt, Aug. 2005.

[128] M. J. Renczmin. Entwurf eines synthesefhigen MMC-Adapters in VHDL. Studienarbeit, Technische Universität Darmstadt, Feb. 2004. In German.

[129] A. Sonntag. Entwurf einer konfigurierbaren Interleaver-Architektur. Studienarbeit, Technische Universität Darmstadt, Aug. 2004. In German.

[130] N. Tu. Processor-based overhead processing in optical transport networks. Diploma thesis, Technische Universität Darmstadt, Oct. 2004.

[131] S. Wang. Modellierung eines 32-bit-Prozessor-Cores auf Transaktionsebene. Studienarbeit, Technische Universität Darmstadt, Oct. 2003. In German.

Curriculum Vitae

Mihail Petrov

Zur Person:

Geburtsdatum:	27. Dezember 1977
Geburtsort:	Constanta, Rumänien

Ausbildung und beruflicher Werdegang:

1984 – 1992	Besuch der Grundschule in Constanta, Rumänien
1992 – 1996	Besuch des Lyzeums für Elektrotechnik und Telekomunikation in Constanta, Rumänien Abschluss: Abitur (*Bacalaureat*)
1996 – 2001	Student an der Fakultät für Elektronik, Nachrichtentechnik und Informationstechnik, Universität 'Politehnica' Bukarest, Rumänien Abschluss: Diplom Ingenieur (*Inginer Diplomat*)
11.2001 – 10.2005	Doktorand und wissenschaftlicher Mitarbeiter am Fachgebiet Mikroelektronische Systeme der Technischen Universität Darmstadt
01.2002 – 12.2004	Stipendiat des Graduiertenkollegs "Systemintegration für ubiquitäres Rechnen in der Informationstechnik"
11.2005 – 06.2007	Systemingenieur bei Micronas GmbH Villach, Österreich
seit dem 07.2007	Forschungsingenieur bei Panasonic R&D Center Langen, Deutschland