Model-based Waveform Design for Heterogeneous SDR Platforms with Simulink

Dipl.-Ing. Michael Schwall, Dipl.-Ing. Stefan Nagel, Prof. Dr.rer.nat. Friedrich K. Jondral
2011 Symposium & Wireless Summer School, Virginia Tech, Blacksburg, USA
Content

- Introduction
- Model-based Waveform Design with Simulink
- The Software Defined Radio Platform
- Case Study: Digital Communication Link
- Benchmarks, Waveform Implementations and Conclusion
Introduction
Introduction

„Traditional“ Radios

„One radio for one communication standard“

Software Defined Radios

„One radio for several communication standards“
The basic concept of Software Defined Radios (SDR)

- Non-functional unit
- Hardware
- Operating System (OS)
- Application Programming Interfaces (APIs)

- Describes the functional relations
- Configures the platform according to a specified communication standard

<table>
<thead>
<tr>
<th>ADC</th>
<th>Analog-Digital Converter</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC</td>
<td>Digital-Analog Converter</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
</tbody>
</table>

- ADC/DAC
- RF frontend
- Digital Signal Processing
Introduction

What are the advantages of such an approach?
What are the challenges for the SDR platform?

- **Reconfigurable**
  - Programmable digital signal processing units
    - DSP
    - FPGA
    - GPP
  - Configurable ADC/DAC and RF frontends

- **High capabilities**
  - Fast digital signal processing units
  - Fast bus systems

- **Cover a large frequency range**
  - Tunable RF frontends
  - High bandwidth

**Abbreviations:**
- DSP – Digital Signal Processor
- FPGA – Field Programmable Gate Array
- GPP – General Purpose Processor
Introduction

What are the challenges for the waveform design?

- Reconfigurable
- High capabilities
- Cover a large frequency range

Requirements

Waveform design

- Handling of heterogeneous SDR platforms
  → Combinations of different digital signal processing units on one platform
- Consideration of processor specific aspects
- Generation of efficient machine code
- Integration of several APIs
Introduction

Solution …

Reconfigurable  High capabilities  Cover a large frequency range

Requirements

Waveform design

Model-based design with Simulink

- Modeling of complex dynamic systems
- Code generation tools for various processors
- Several debug and analyze possibilities
Model-based Waveform Design with Simulink
Model-based Waveform Design with Simulink

Introduction

- Model-based design is a mathematical and visual method to describe complex systems, e.g. waveforms or control systems.

- The presented waveform design approach [NSJ10][Nag11] is derived from the Model Driven Architecture (MDA) [MDA11]. The MDA was launched by the Object Management Group (OMG) in 2001.

- The entire design flow, starting from a waveform specification to the point of machine code, is divided into four steps.

- The approach is tailored to the physical (PHY) and data link (MAC) layer of wireless communication systems.

- The main objectives are efficiency and portability.
Model-based Waveform Design with Simulink

Conversion with Simulink

Simulink is a software tool from The MathWorks [Mw11] featuring …

- an entire model-based design environment
- the simulation and analyzing of dynamic systems
- a comprehensive library with predefined functions from different technical domains, e.g. Communications or Image Processing
- diverse add-on products like the Real-Time Workshop (now Simulink Coder) or the HDL Coder
Model-based Waveform Design with Simulink

- Simulink ... a quick overview

![Library Browser](image1)

![Model](image2)

![Visualization](image3)
Model-based Waveform Design with Simulink

- **Computation Independent Model:**
  - Verbal or textual specification of the waveform
  - Requirements of the waveform
  - Independent of the underlying platform
  - Examples:
    - IEEE 802.11g
    - ETSI TETRA
    - IEEE 802.15.1 Bluetooth

![Diagram of Model-based Waveform Design with Simulink](image)

**Abbreviations:**
- IEEE – Institute of Electrical and Electronics Engineers
- ETSI – European Telecommunications Standards Institute
- TETRA – Terrestrial Trunked Radio
Model-based Waveform Design with Simulink

- **Platform Independent Model:**

  - Modeling of the waveform in Simulink without platform specific aspects

  - Simulink enables an intuitive way to model complex systems:
    - Signal processing elements are mapped to functional blocks
    - A system is created by interlinking and parameterizing these blocks

  - This design step tends to simulate and analyze the waveform's functionality
Model-based Waveform Design with Simulink

Platform Specific Model:

- Extending the PIM with hardware specific aspects

Infrastructural aspects:
- Examples:
  - Data buses within the platform
  - Configuration of the RF frontends
  - Application Programming Interfaces

Processor Specific aspects:
- Examples:
  - Fixed-Point representation
  - Memory allocation
  - Compiler Options, model parameterization and adaption
Model-based Waveform Design with Simulink

- **Build Process:**
  - The build process depends on the processor
  - **DSP/GPP:**
    - C/C++ code is automatically generated using the inbuilt Real-Time Workshop
    - Processor specific libraries are linked to the project
    - Compilation with an appropriate compiler
      - Examples: Texas Instruments Code Composer Studio, Microsoft Visual Studio compiler
  - **FPGA:**
    - Full-proprietary: The entire build process is accomplished by a plug-in from the FPGA vendor
      - Examples: XILINX System Generator for DSP
    - Semi-proprietary: The HDL description is generated by Simulink (HDL-Coder), the remain is handled by the vendor software
Model-based Waveform Design with Simulink

Machine Code:

- The machine code is uploaded to the processor

- Dependent on the processor (DSP or GPP) and the execution mode, the Real-Time Workshop features tuning and monitoring of the running waveform
The Software Defined Radio Platform
The Software Defined Radio Platform

- Universal Software Radio Peripheral (USRP)
  - Vendor: Ettus Research LLC [Ett11]
  - Open source project (Schematics available)
  - “Just” the air interface
General setup:

- **PC** — Personal Computer
- **USB** — Universal Serial Bus
The Software Defined Radio Platform

- **USRP mainboard assembly:**
  - **USB interface**
    - Max. data rate 8 MHz (@16 bit)
  - **Altera FPGA EP1C12 [Cyc11]**
  - **Analog Devices 9862 [AD11]**
    - Conversion Chip
    - Two ADC (64 MSPS @12 bit)
    - Two DAC (128 MSPS @14 bit)
    - Interpolation filters
    - Digitally tunable up/down converters
  - **Connector to the daughterboards**

---

**Diagram:**

- **USB interface**
- **Rx FIFO**
- **Tx FIFO**
- **Control**
- **I**
- **Q**
- **Rx chain**
- **Tx chain**
- **HBF**
- **MSPS**
- **DAC**
- **ADC**
- **DUC**
- **FPGA**
- **DAC/ADC chip**

**Symbols:**

- **Tx** — Transmit
- **Rx** — Receive
- **DUC** — Digital Up Converter
- **FIFO** — First In, First Out buffer
- **HBF** — Half Band Filter
- **MSPS** — Mega Samples per Second

---

**Notes:**

- 03.06.2011
- Dipl.-Ing. Michael Schwall
- Model-based Waveform Design for Heterogeneous SDR Platforms with Simulink
- Prof. Dr.rer.nat. Friedrich K. Jondral

---

**Figure:**

- Diagram of the USRP mainboard assembly and its components.
The Software Defined Radio Platform

**USRP Daughterboard**

- **RFX2400**
- **Industrial Scientific Medical (ISM) frequency band coverage**
- **Transmit side:**
  - Frequency range: 2.3 GHz - 2.9 GHz
  - Power: 50 mW (17 dBm)
- **Receive side:**
  - Frequency range: 2.3 GHz - 2.9 GHz
The Software Defined Radio Platform

- Simulink-USRP Plug-in
  - Open source project [SU11]
  - Download: http://www.cel.kit.edu/english/downloads.php

- Enables the connection between the USRP and the PC
  - Simulink-USRP API

- Development of custom FPGA configurations
  - Bitfile Generator
Simulink-USRP API

- Application Programming Interface that enables the connection between the USRP and Simulink
- The C-code is derived from the GNU Radio [Blo04] project
- The plug-in provides primarily a sink and a source block to transfer IQ data between the USRP and the PC
- Simulink USRP GPP library features:

Displays general information

Reads from the USRP

- usrp_source
- Set USB Firmware
- Set Bitsream

Writes to the USRP

- usrp_sink
- Set FPGA Bitstream
Simulink-USRP API

- The sink and source block can be used to parameterize the corresponding transmit or receive side:

 becomes obsolete when using custom FPGA configuration
The Software Defined Radio Platform

- **Bitfile Generator**
  - The Bitfile Generator is used to generate a custom configuration for the Altera Cyclone FPGA on the USRP mainboard.
  - Transmit and receive chain for side A can be modeled in Simulink with the inbuilt HDL-Coder.

- **Limitations:**
  - Only a subset of entire Simulink library is available (HDL library).
  - Sample-based processing.
  - Fixed-point arithmetic.

- **Simulink USRP FPGA library features:**
The Software Defined Radio Platform

- **Bitfile Generator**
  
  - **Build process:**
    1. Simulink HDL-Coder produces FPGA independent Verilog or VHDL code
    2. The code is embedded as a custom logic into a template project
    3. The technology-mapped netlist is generated by the HDL-Coder
    4. The place-and-route is performed by proprietary software from the FPGA vendor (e.g. Altera Quartus)

- The entire process is initiated and managed by the Bitfile Generator
Case Study: Digital Communication Link
Case Study: Digital Communication Link

The case study will cover all steps from a waveform specification to machine code.


Case Study: Digital Communication Link

- Computation Independent Model

- Waveform specification:
  - Single carrier transmission
  - QPSK modulation
  - Data rate: 100 kbit/s
    - Bit duration: \( T_B = 10 \ \mu s \)
    - Symbol duration: \( T_S = 20 \ \mu s \)
  - Pulse-shaping: Root Raised-Cosine (RRC) filter with a roll-off factor \( \alpha = 0.35 \)
  - Carrier frequency 2.41 GHz (ISM-band)

QPSK — Quadrature Phase Shift Keying

I — In-phase
Q — Quadrature
Case Study: Digital Communication Link

Platform Independent Model

Transmitter

Channel

Receiver
Case Study: Digital Communication Link

Sample-based and frame-based processing

Sample-based
- process one sample
- Interrupt Service Routine

Frame-based
- process multiple samples sequential

Sample-based processing
- Low processing latency
- High process overhead due to ISRs after every sample

Frame-based processing
- Fixed process overhead is distributed across many samples
- Common format in real-time systems and communications
- Increases latency
Case Study: Digital Communication Link

- Integrating C/C++ Code into Simulink

C/C++ code is embedded into an S-function (manually or with the Legacy Code Tool)

S-functions are specifically constructed Simulink functions written in C/C++

To simulate an S-function in the PIM, it has to be compiled to an Matlab Executable (MEX) file
Case Study: Digital Communication Link

- Integrating C/C++ Code into Simulink
- Overall S-function structure

```c
/* Define; Include; Set global variables */

static void mdlInitializeSizes(SimStruct *S)
{
    /* Port initialization */
}

static void mdlOutputs(SimStruct *S, int_T tid)
{
    /* Signal processing */
}

static void mdlTerminate(SimStruct *S)
{
    /* Free memory */
}
```
Case Study: Digital Communication Link

- Integrating C/C++ Code into Simulink
  - Generate a Matlab Executable with the `mex` command

  \[ \text{mex flags mySfunction.c} \rightarrow \text{mySfunction.mexw32} \]

- Embed mex file into model

  ![S-Function Block Parameters](image)

  S-Function
  - User-definable block. Blocks can be written in C, C++, and Fortran and must conform to S-function standards. The variables y, u, and flag are automatically passed to the S-Function by Simulink. You can specify additional parameters in the S-Function parameter field. If the S-Function block requires additional source files for the Real-Time Workshop build process, specify the filenames in the 'S-Function modules' field. Enter the filenames only; do not use extensions or full pathnames, e.g., enter 'arc and', not 'arc and.lib'.

  Parameters
  - S-Function name: mySFunction
  - S-Function parameters:
  - S-Function modules: *
Case Study: Digital Communication Link

Channel:
- For the evaluation of the synchronization algorithms in the receiver, a channel must be simulated
- Actually not part of the PIM
- The channel behaves like a virtual frontend and comprises
  - Additive White Gaussian Noise \( n(t) \)
  - Phase offset \( \Phi \)
  - Frequency shift \( \Delta f \)
  - Time drift \( \varepsilon \)
- The received signal can be described as:

\[
 r(k/N) = s(k/N - \varepsilon) \cdot e^{j(2\pi \Delta f k/N + \phi)} + n(k/N)
\]
Case Study: Digital Communication Link

- Frequency synchronization
  - Symbol-time-based method to compensate the frequency shift of a phase modulated base band signal \([MD97]\)

1. The received signal is multiplied with a delayed copy of itself to remove the constant phase offset

2. The resulting signal is exponentiated by the constellation order \(M\) (QPSK: \(M=4\))
   → Due to the phase shift modulation, the calculation of the frequency offset becomes independent of the signal's information

\[
\Delta \hat{f} = \frac{1}{2\pi T M} \cdot \arg \left( \sum_{k=0}^{L-1} (r(k) \cdot r^*(k-1))^M \right)
\]

3. After compensation:
   \[
r'(k/N) = s(k/N - \epsilon) \cdot e^{j\phi} + n(k/N)
   \]
Case Study: Digital Communication Link

- Timing synchronization
  - The feed-forward method Squaring Timing Recovery [OM88] is performed on the oversampled signal

1. The absolute value of the signal is squared

2. The frequency component at a multiple of the symbol rate is calculated

\[
\bar{c} = -\frac{T N}{2\pi} \arg \left( \sum_{k=0}^{LN-1} |r'(k/N)|^2 \exp \left( -j \frac{2\pi k}{N} \right) \right)
\]

3. After compensation:

\[
r''(k) = s(k) \cdot e^{j\phi} + n(k)
\]
Case Study: Digital Communication Link

- Timing synchronization
- Spectrum of the signal: \(|r'(k/N)|^2\)

![Graph showing spectral analysis](Image)

- Phase of the peak keeps timing offset information
- Symbol rate

Magnitude-squared, dB

Frequency (kHz)
Case Study: Digital Communication Link

- Phase synchronization
  - The remaining phase offset of the signal is determined and adjusted with a Costas loop [MD97]

\[ \hat{\phi}(k + 1) = \hat{\phi}(k) + \xi(k) \]

\[ \xi(k) = \xi(k - 1) + \gamma (1 + p)e(k) - \gamma e(k - 1) \]

- After compensation:

\[ r'''(k) = s(k) + n(k) \]
Case Study: Digital Communication Link

Platform Independent Model

Transmitter

Channel

Receiver
Case Study: Digital Communication Link

Platform Specific Model

- Platform specific aspects have to be identified:

  - Two programmable digital signal processing units:
    - GPP
    - FPGA

  → Waveform segmentation

- Data rate conversion between the waveform and the ADC/DAC has to be considered

\[
\frac{1}{T_S} = 50 \text{ kHz} \quad f_{\text{DAC}} = 32 \text{ MHz} \quad f_{\text{ADC}} = 64 \text{ MHz}
\]

→ Interpolation and decimation
Case Study: Digital Communication Link

Waveform segmentation

Transmitter

Channel

Receiver
Case Study: Digital Communication Link

Waveform segmentation

Transmitter
- GPP
- PIM transmitter (except of pulse shaping)

Receiver
- GPP
- PIM receiver

FPGA
- usrp_sink
- Pulse shaping and interpolation

FPGA
- usrp_source
- Decimation

Over the air

→ 4 Simulink models
Case Study: Digital Communication Link

Transmitter

- Adding `usrp_sink`, `usrp_helper`, `Set Firmware` and `Set Bitstream`
- No further modification concerning data types necessary

Sample rate 50 kHz
Case Study: Digital Communication Link

- Transmitter
  - Modeling with Simulink HDL library
    - Subset of Simulink library
    - Only sample-based processing and real fixed-point arithmetic

- Interpolation factor:
  \[ I = \frac{f_{DAC}}{N \cdot 1/T_S} = \frac{32 \text{ MHz}}{8 \cdot 50 \text{ kHz}} = 80 \]

In-phase → [Pulse shaping]

Quadrature → [CIC [Hog81] interpolation stage]

\[ I_1 = 10 \quad I_2 = 8 \]

Sample rate 50 kHz → Sample rate 32 MHz
**Case Study: Digital Communication Link**

- **Receiver**
  - **Decimation factor:**

\[
D = \frac{f_{ADC}}{N \cdot 1/T_S} = \frac{64 \text{ MHz}}{8 \cdot 50 \text{ kHz}} = 160
\]

**CIC decimation stage**

- **In-phase**
- **Quadrature**

Sample rate 64 MHz → $D_1 = 20$ → $D_2 = 8$ → Sample rate 400 kHz

- In-phase
- Quadrature

Sample rate 64 MHz

$D_1 = 20$

$D_2 = 8$

Sample rate 400 kHz
Case Study: Digital Communication Link

Receiver

- Adding *usrp_source, usrp_helper, Set Firmware* and *Set Bitstream*
- No further modification concerning data types necessary
- An Automatic Gain Control (AGC) is inserted to normalize the signal’s amplitude

Sample rate 400 kHz

```
Sample rate 400 kHz
```

Extension

```
Extension
```

PSM

GPP
Case Study: Digital Communication Link

- The configuration of the FPGA’s PSM is generated using the Bitfile Generator and Altera Quartus [QUA11].

- The custom configuration (raw binary file, *.rbf) has to be selected in the appropriate GPP model with the `Set Bitstream` block.

- The custom configuration is uploaded to the FPGA whenever executing the GPP model.
Case Study: Digital Communication Link

- The GPP model can be executed in two different ways

  - Execution within Simulink:
    - Normal execution
      - Interpreting the model code
    - Using the Accelerator or Rapid Accelerator mode
      - Interpreting the model code with the use of precompiled functions
    - Provides most Simulink visualization, benchmark and debug features during runtime

  - Execution using the Real-Time Workshop:
    - C-code is automatically generated
    - Code is compiled using an installed compiler (all compiler and linker options are available)
    - Start of the executable outside Simulink
Benchmarks, Waveform Implementations and Conclusion
Comparison between hand written and generated code on a GPP

- Implementation: FIR filter, Processing time is normalized to filter coefficients

![Comparison graph]

**Benchmarks**

- Intel Core 2 Duo CPU (P8400)
- 2.28 GHz clock frequency

**Compiler**

- LCC – Local C Compiler
- VS – Visual Studio compiler (Microsoft)
- IPS – Intel Parallel Studio
Comparison between hand written and generated code on a GPP

Implementation: Fast Fourier transform

- Intel Core 2 Duo CPU (P8400)
- 2.28 GHz clock frequency

- Fastest Fourier Transform in the West [FFT11]
- Optimized software library for computing the discrete Fourier transform
Comparison between hand written and generated code on a DSP

- Implementation: Fast Fourier transform

**Benchmarks**

<table>
<thead>
<tr>
<th>Processing time</th>
<th>Cycles/1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 s</td>
<td>594000</td>
</tr>
<tr>
<td>100 ms</td>
<td>5940</td>
</tr>
<tr>
<td>10 ms</td>
<td>594</td>
</tr>
<tr>
<td>1 ms</td>
<td>594</td>
</tr>
<tr>
<td>100 μs</td>
<td>59.4</td>
</tr>
<tr>
<td>10 μs</td>
<td>5.94</td>
</tr>
</tbody>
</table>

**DSP**
- Texas Instruments C64x+
- Fixed-point processor
- 594 MHz clock frequency

**Optimized library**
- Texas Instruments C64x DSP library
- Embedded in Simulink → Target Support Package
- Algorithms: FFT, FIR, ...
Comparison between fixed and floating point implementation on a DSP

- Implementation: FIR filter, Processing time is normalized to filter coefficients

**Benchmarks**

- **Code**
  - Hand written
  - Generated
  - Optimized

**Cycles**

- **DSP**
  - Texas Instruments C64x+
  - Fixed-point processor
  - 594 MHz clock frequency

**Optimized library**

- Texas Instruments C64x DSP library
- Embedded in Simulink → Target Support Package
- Algorithms: FFT, FIR, …
Comparison between hand written and generated code on a FPGA

- Implementation: CIC filter

Benchmarks

FPGAs

- Altera Cyclone EP1C12: 12060 Logic Elements (LE)
- XILINX Virtex-4 XC4VSX35: 15360 slices and 192 multiplier
Conclusion:

- Simulink generates efficient C/C++ and HDL code
- Execution can be accelerated by
  - … using optimized libraries (e.g. FFTW, TI’s DSP library)
  - … considering the processor specific aspects (e.g. fixed-point arithmetic) when modeling the PSM
  - … choosing the right compiler
Waveform Implementations

- Introducing the SFF SDR DP
  - Vendor: Lyrtech [Lyr11]
  - Stand-alone platform

- 3 Modules:
  - RF-Module
  - Data-Conversion Module
  - Digital Signal Processing Module
Comparison
Waveform Implementations

- **AM/FM waveform**

Ported to:
Waveform Implementations

- TETRA waveform

Ported to:
Waveform Implementations

- WLAN similar to IEEE 802.11g

Ported to:

IEEE 802.11a-1999, PHY-Layer

Communication Scenario
- Self-Exercise
- Interpretation

IEEE Std 802.11a-1999 96c
- f_c = 20000000 Hz
- f_s = 50.068 s
- f_s = 315000 kHz
- f_s = 650000 Hz
- Frame: #/frame: 750000
- Status: 13000.3889
- Bit/s: 555.5556

Transmitter

Channel

Receiver
Waveform Implementations

For teaching purposes
Conclusion

Pros:

- Simulink is suited for the model-based waveform design approach
- Waveform reusability is achieved by the Platform Independent Model
- Platform specific aspects are considered in the Platform Specific Model
- The automatic generated code is efficient. Optimized libraries can be integrated
- Several hardware vendors offer tools and libraries for Simulink
Conclusion

Cons:

- Existing platform APIs have to be integrated into the Simulink environment (e.g. the Simulink USRP API)
- Simulink is not for free
Literature


Literature


Thank you for your attention!

Q&A