A SystemC-AMS library for supporting the design of reconfigurable communication systems

Florian Brame, Christoph Grimm, Markus Damm, Jan Haase
TU Vienna, Institute of Computer Technology

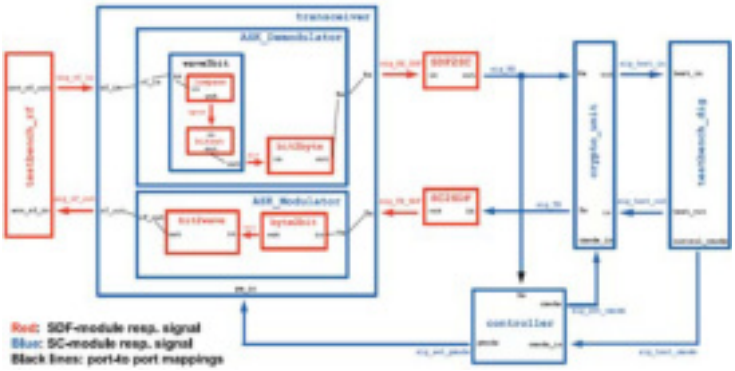
SystemC-AMS provides the designer with a tool that supports the system- and architecture level modelling and simulation of RF-, analog and mixed-signal systems. However, designers usually want libraries that provide a set of complex building blocks that

- Allow them to set up a first running system from a restricted number of blocks.
- Encapsulate high-level IP blocks as a starting point for later refinement.
- Provide some useful help in dimensioning, or selecting structures.

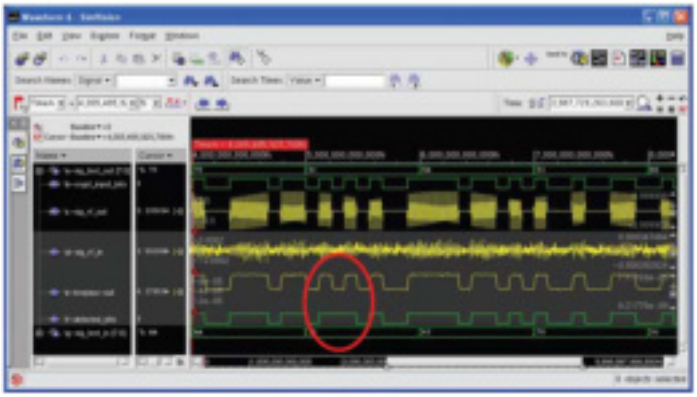
Good examples for such libraries are the libraries that come with or are available for Matlab/Simulink. In the context of the EC FP 6 project ANDRES we are developing a library that supports the design of reconfigurable communication systems. The library specifically supports the designer in selecting appropriate architectures for communication systems that must be reconfigurable to some extent, such as SW defined radio.

The designer can specify a kind of reconfigurability for all building blocks (e.g. by changing parameters, by selecting a different function of a predefined set, by programming a new function, by programming a new function including changes in data rates). For given technology parameters, the library can simulate different implementation variants (or deny, if not appropriate).

As an example we have implemented a simple ASK receiver (see figure).



It contains re-configurable modulators, a controller and combines some typical (still simple) building blocks. The next figure shows the simulation outputs.



The library contains noise sources (gaussian, uniform, …), modulators (passband FM, AM, FSK, ASK, IQ,…), mapper for QPSK, QAM (4,16,…) , demodulators, demapper for QPSK,, QAM (4, 16, …), filters (FIR, IIR; parameterizable), converter for eye diagram to vector format (SVG), IQ constellation, BER calculation, and maybe channel models for the analysis of e.g. sensor networks with multiple transceivers. Where appropriate, the modules use multirate data flow.

Contact:

Univ.Prof. Dr. Christoph Grimm
Technische Universität Wien
Gusshausstraße 27-29/384
1040 Wien
Österreich

Email: grimm@ict.tuwien.ac.at
Tel: +43 (1) 58801 38415

A SystemC-AMS library for supporting the design of reconfigurable communication systems

Institut für
Computertechnik

ICT

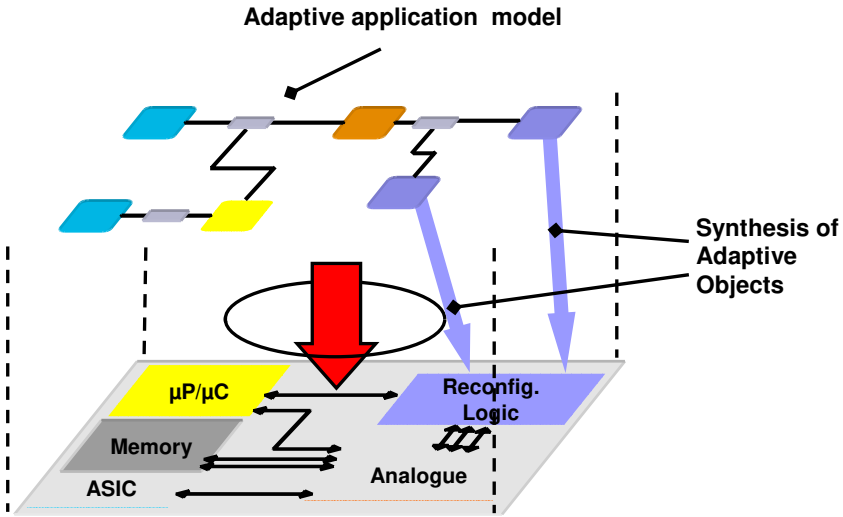
Institute of
Computer Technology

F. Brame, Ch. Grimm, M. Damm, J. Haase

TU Vienna
Institute of Computer Technology
Chair for Embedded Systems

- **The ANDRES project**
- Basic concept for library
- Components of the library
- Example
- Status of implementation & Future work

- **AN**alysis and **D**esign of **R**un-Time **RE**configurable **S**ystems
  - real-world example: SW Defined Radio
  - ANDRES WP1: System partitioning under consideration of requirements for reconfigurability!



- For each process: to what extend is reconfigurability needed?
- How can process be realized? Analog|FPGA|ASIC|SW?

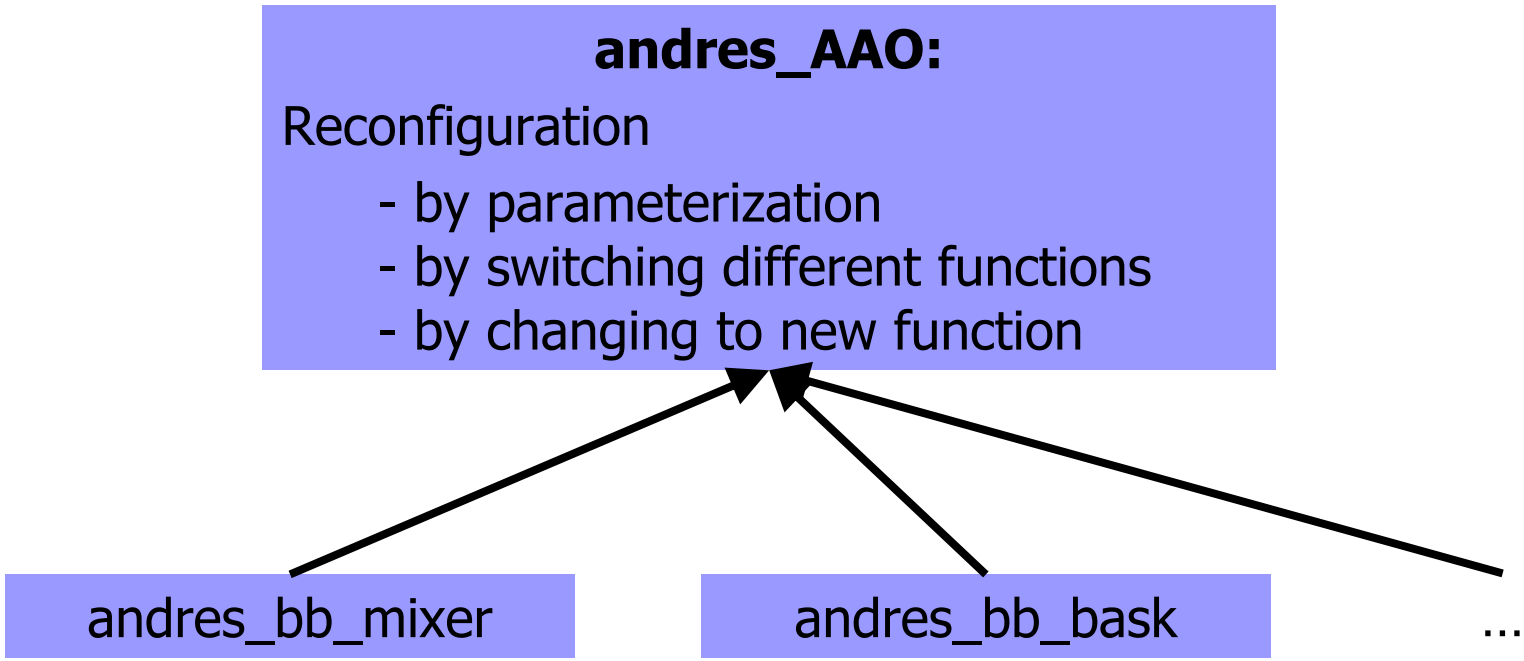
- **System & architecture level design** is done in an **interactive** way by modelling and simulation

- Very useful for modelling/simulation @ architecture level
  - Component libraries à la Matlab/Simulink toolbox
  - For ANDRES WP 1 in Task 1.4:
    1. Components of communication systems
    2. Model properties required for design space exploration

- The ANDRES project
- **Basic concepts for library**
- Components of the library
- Status of implementation & example
- Future work

Specification of Reconfigurability / Adaptivity

- All components inherit an ***abstract adaptive object*** that allows designer the specification of reconfigurability



Architecture level models of FPGA, DSP, ASIC and Analog implementation

Model properties for design space exploration

- Specify architecture properties by paramterization, e.g. Bitwidth
- Specify non-ideal properties by parameterization, e.g. IQ mismatch, noise, distortions, jitter, …
- Provide hierarchical decompostion into building blocks that are AAOs as well
- **Objective:**
  - Keeps library compact
  - Simplifies modification/evaluation cycle in design space exploration
  - **supports refinement of requirements for adaptivity**

- The ANDRES project
- Basic concept for library
- **Components of the library**
- Example
- Status of implementation & Future work

Classification

- Signal sources
- Converters
- Mappers / Demappers
- Modulators / Demodulators
- Analysis tools
- Misc

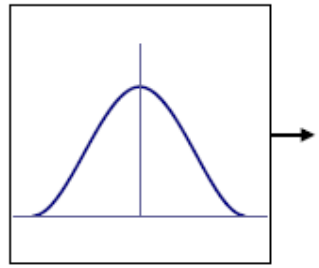
Some signal sources …

- **Gaussian noise**

Class definition:

```
andres_bb_noise_gauss(sc_module_name nm,
      double _variance,
      double _mean) ;
```

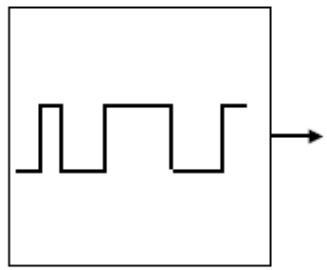
Interfaces: `sca_sdf_out<double> out ;`



- **Uniformly distributed random bits**

Class definition: `andres_bb_rand_bool(sc_module_name nm) ;`

Interfaces: `sca_sdf_out<bool> out ;`

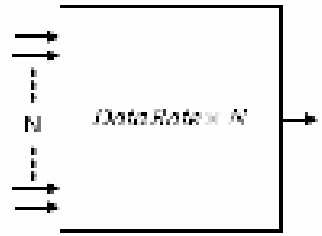


Converters

- Parallel to serial

Class definition: `andres_bb_par2ser(sc_module_name nm);`

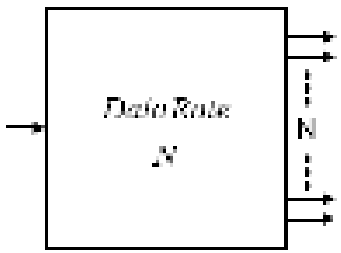
Interfaces:
```
sca_sdf_in<sc_bv<N> > in ;
sca_sdf_out<bool> out ;
```



- Serial to parallel

Class definition: `andres_bb_ser2par(sc_module_name nm);`

Interfaces:
```
sca_sdf_in<bool> in ;
sca_sdf_out<sc_bv<N> > out ;
```

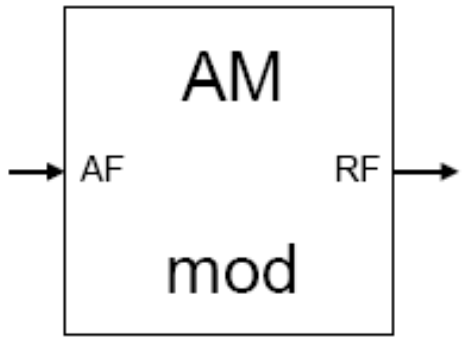


Mappers/Modulators/Demodulators/Demappers

- **Amplitude modulator**

Class definition:

```
andres_bb_mod_am(sc_module_name nm,
      double _freq,
      double _ampl,
      double _offset,
      int _rate) ;
```



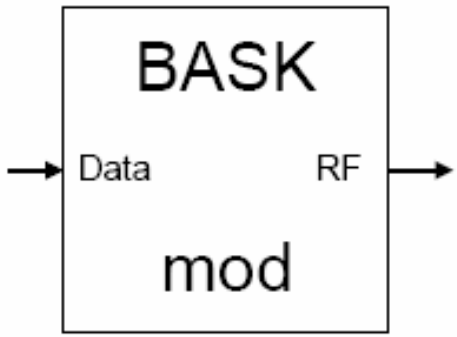
Interfaces: `sca_sdf_in<double> in ;`
sca_sdf_out<double> out ;

Mappers/Modulators/Demodulators/Demappers

- **Binary Amplitude Shift Keeying Modulator**

Class definition:

```
andres_bb_mod_bask(sc_module_name nm,
      double _freq,
      double _ampl1,
      double _ampl0,
      double _phi,
      int _rate) ;
```



Interfaces:

```
sca_sdf_in<bool> in ;
sca_sdf_out<double> out ;
```

Mappers/Modulators/Demodulators/Demappers

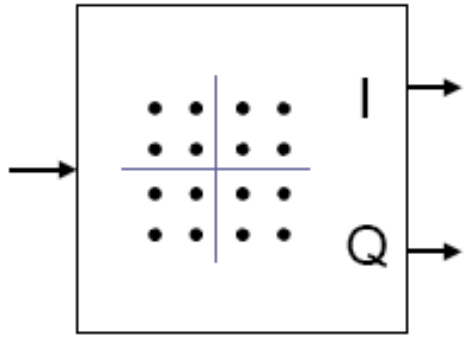
- **IQ mapper (4,16, 64, …)**

Class definition:

```
andres_bb_mod_iq(sc_module_name nm,
      double _freq,
      double _d_ampl,
      double _d_phi,
      int    _rate);
```

Interfaces:

```
sca_sdf_in<double>  i   ;
sca_sdf_in<double>  q ;
sca_sdf_out<double>  out ;
```



Modulators/Demodulators/Mappers/Demappers

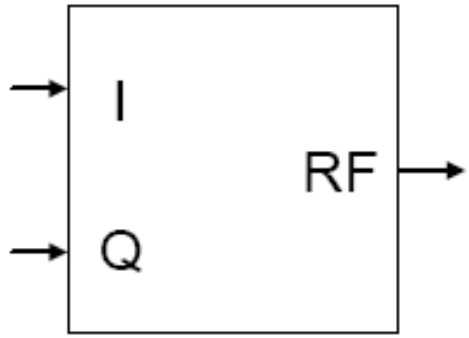
- **IQ modulator**

Class definition:

```
andres_bb_mod_iq(sc_module_name nm,
      double _freq,
      double _d_ampl,
      double _d_phi,
      int    _rate);
```

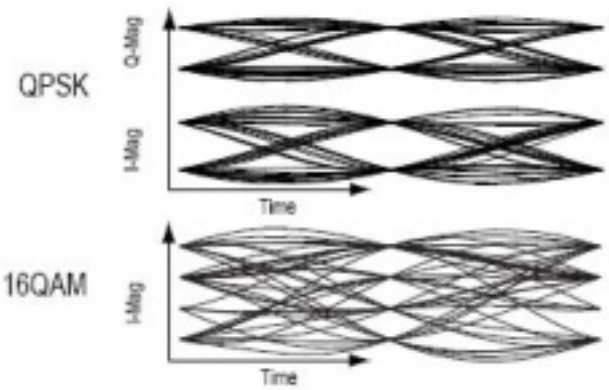
Interfaces:

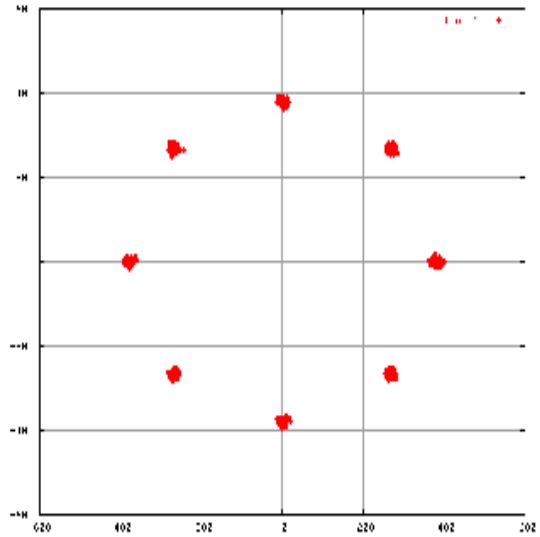
```
sca_sdf_in<double>  i   ;
sca_sdf_in<double>  q ;
sca_sdf_out<double>  out ;
```



Analysis tools

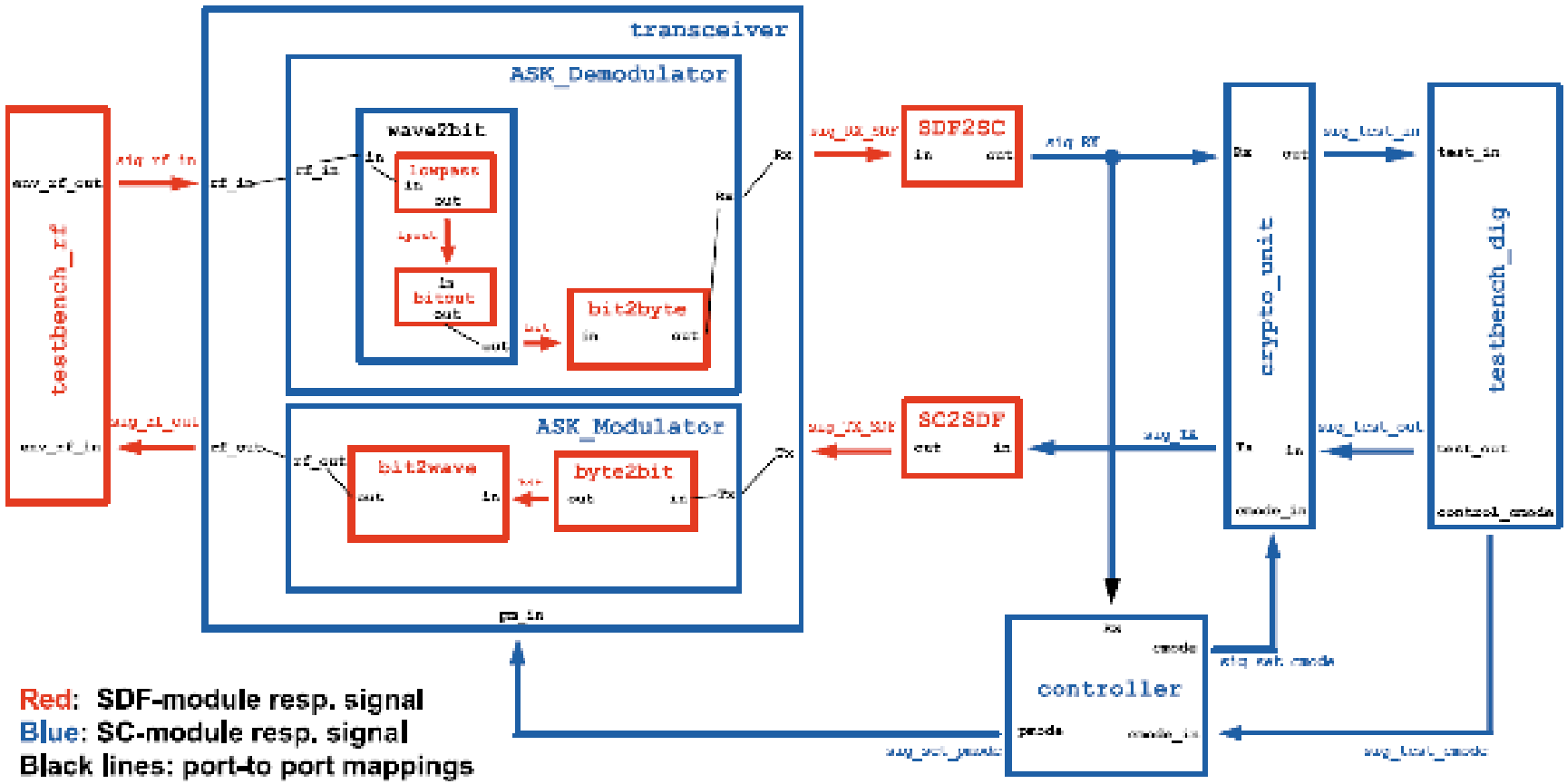
- Tools to analyze signal quality by plotting
  - eye diagrams
  - trellis diagrams
  - constellation diagrams
  - **… as SVG file**



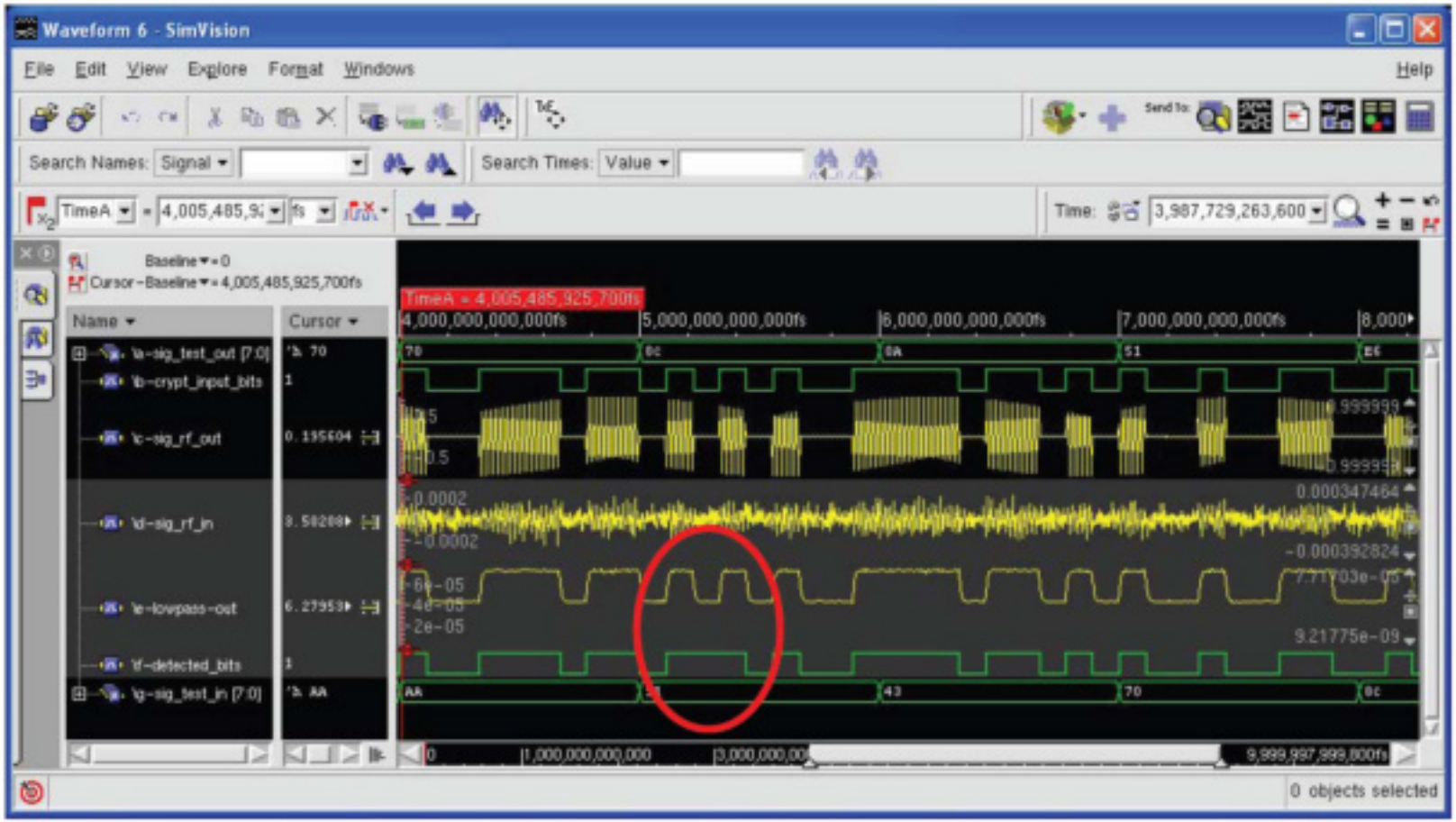


- The ANDRES project
- Basic concept for library
- Components of the library
- **Example**
- Status of implementation & Future work

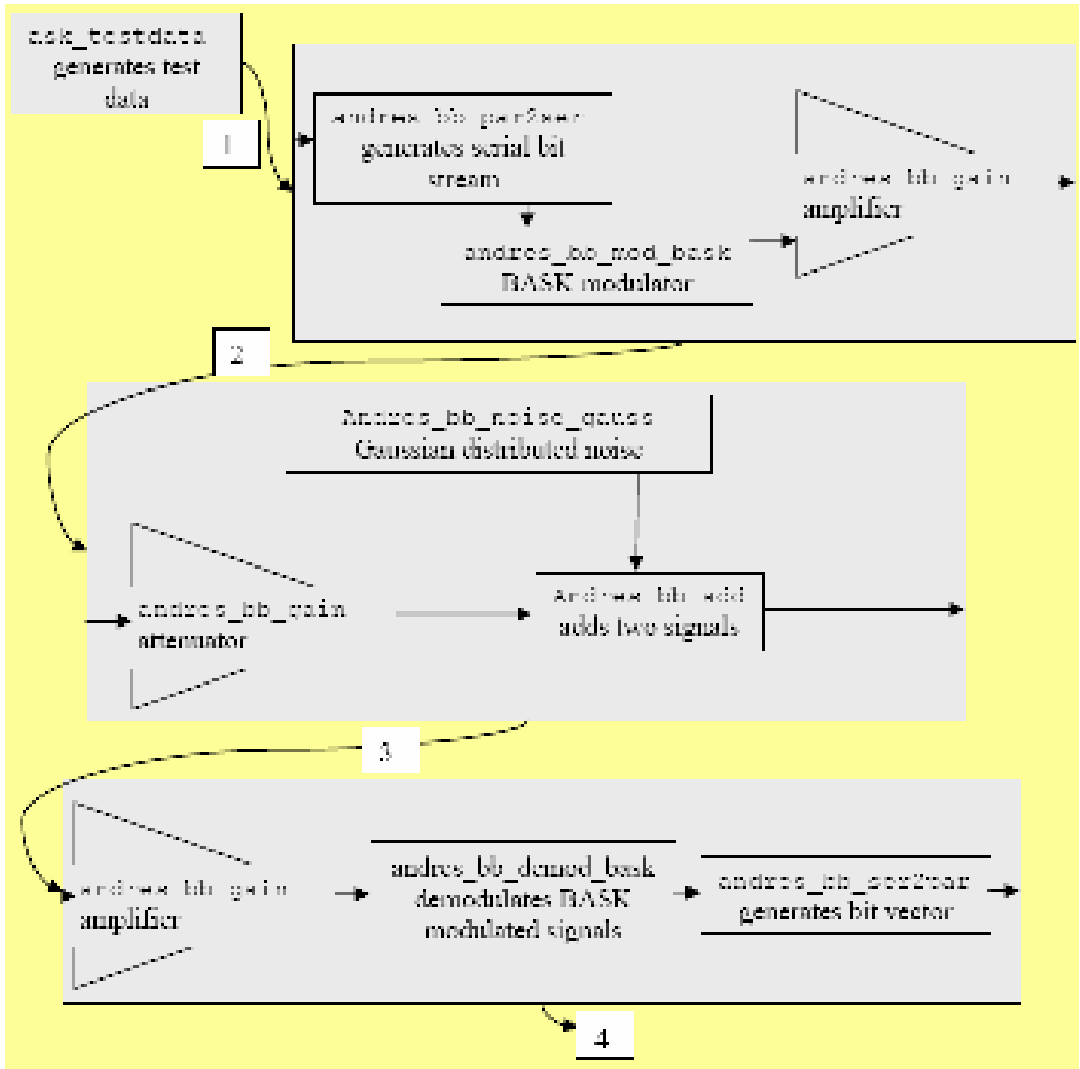
Overview of ASK transceiver system



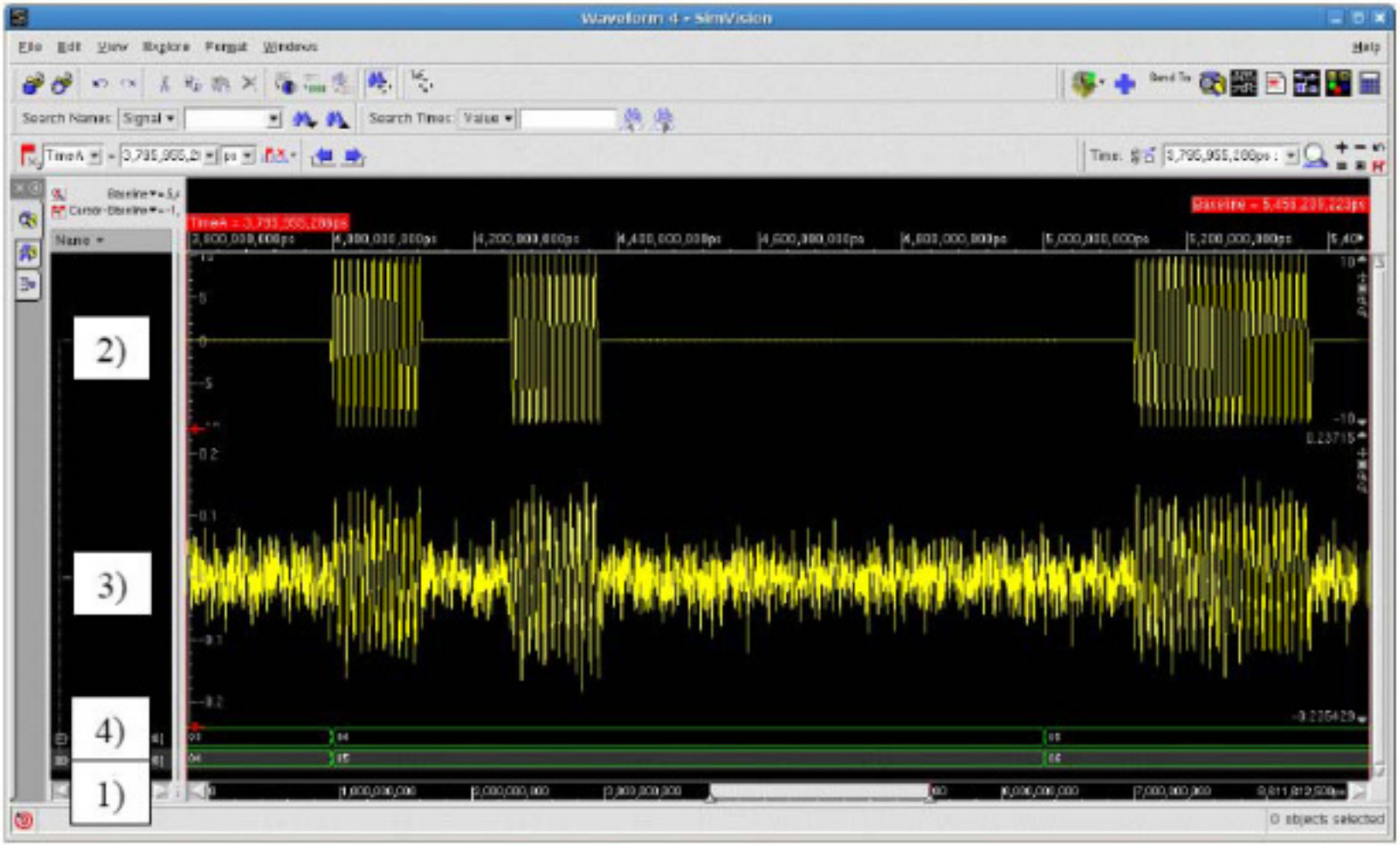
Example …


Waveform 6 - SimVision
File Edit View Explore Format Windows Help
Search Names: Signal
Search Times: Value
Time: 3,987,729,263,600
Baseline = 0
Cursor - Baseline = 4,005,485,925,700fs
Name
Cursor
\a-sig_test_out [7:0]
\b-crypt_input_bits
\c-sig_rf_out
\d-sig_rf_in
\e-lowpass-out
\f-detected_bits
\g-sig_test_in [7:0]
4,000,000,000,000fs
5,000,000,000,000fs
6,000,000,000,000fs
7,000,000,000,000fs
0 objects selected

Model of ASK with ANDRES Building Blocks



Simulation results



- The ANDRES project
- Basic concept for library
- Components of the library
- Example
- **Status of implementation & Future work**

Status of implementation

- First, initial version published to project partners
  - Many, even complex components available
  - Basic architecture level parameters available

- But reconfigurability not yet implemented
  - AAO concept not yet integrated in library components
  - Non-ideal behavior of reconfigurable realizations not yet implemented

Future work

- even more Modulators, Mappers, Tools …
- Integrate AAO concept
- Model non-ideal behavior of different re-configurable implementations in
  - Analog
  - Digital ASIC
  - FPGA
  - Software on DSP
- Evaluate it with real complex example e.g. cognitive radio