A Fast Forward Error Correction Toolbox: Seminary

Adrien Cassagne\textsuperscript{1,2} Olivier Hartmann\textsuperscript{1} Mathieu Léonardon\textsuperscript{1}
Camille Leroux\textsuperscript{1} Christophe Jégo\textsuperscript{1}

\textsuperscript{1}IMS/CNRS, Bordeaux, France

\textsuperscript{2}Inria, Bordeaux Institute of Technology, U. of Bordeaux, LaBRI/CNRS, Bordeaux, France

IMS, March 2018
Plan

1. Introduction
   - Why AFF3CT?

2. State of Play
   - Simulator
   - Toolbox
   - Prototyping
   - Visualization
   - Miscellaneous

3. Simulation
   - What is a Simulation?
   - Launching Simulations

4. Development
   - Source Code Organization
   - Development in AFF3CT
   - My Project with AFF3CT

5. Contribution
   - Source Code Management
   - Add New Feature
   - Repositories
   - Continuous Integration

6. Roadmap and Discussion
   - What’s next?
Why AFF3CT?

- Knowledge capitalization
Why AFF3CT?

- Knowledge capitalization
- Simulation and prototyping
A Fast Forward Error Correction Toolbox!

Why AFF3CT?

- Knowledge capitalization
- Simulation and prototyping
- Reproducibility of the results
A Fast Forward Error Correction Toolbox!

Why AFF3CT?

- Knowledge capitalization
- Simulation and prototyping
- Reproducibility of the results
- Software Defined Radio
Why AFF3CT?

- Knowledge capitalization
- Simulation and prototyping
- Reproducibility of the results
- Software Defined Radio
- Validation of new algorithms
Why AFF3CT?

- Knowledge capitalization
- Simulation and prototyping
- Reproducibility of the results
- Software Defined Radio
- Validation of new algorithms
- Team building
Plan

1 Introduction
   - Why AFF3CT?

2 State of Play
   - Simulator
   - Toolbox
   - Prototyping
   - Visualization
   - Miscellaneous

3 Simulation
   - What is a Simulation?
   - Launching Simulations

4 Development
   - Source Code Organization
   - Development in AFF3CT
   - My Project with AFF3CT

5 Contribution
   - Source Code Management
   - Add New Feature
   - Repositories
   - Continuous Integration

6 Roadmap and Discussion
   - What’s next?
Monte-Carlo simulations

- BER/FER *standard*
- BER/FER *iterative*
Monte-Carlo simulations
- **BER/FER standard**
- **BER/FER iterative**
- Parallel EXIT charts
## List of Supported Channel Codes

<table>
<thead>
<tr>
<th>Channel code</th>
<th>Standard</th>
<th>Decoder</th>
<th>Fixed point</th>
<th>Throughput (Mb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDPC</td>
<td>5G, WiMAX, WiFi, DVB-S2, 10GE, etc.</td>
<td>Sum Product Algorithm (SPA)</td>
<td>No</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min Sum (NMS, OMS)</td>
<td>Yes</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Approximate Min Star (AMS)</td>
<td>No</td>
<td>20</td>
</tr>
<tr>
<td>Polar</td>
<td>5G</td>
<td>Successive Cancellation (SC)</td>
<td>Yes</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Successive Cancellation List (SCL)</td>
<td>Yes</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Soft Cancellation (SCAN)</td>
<td>No</td>
<td>10</td>
</tr>
<tr>
<td>Turbo</td>
<td>LTE (3G, 4G), DVB-RCS, CCSDS, etc.</td>
<td>Turbo BCJR</td>
<td>Yes</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Turbo BCJR + CRC</td>
<td>Yes</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Turbo BCJR + CRC + Flip aNd Check</td>
<td>Yes</td>
<td>100</td>
</tr>
<tr>
<td>BCH</td>
<td>CD, DVD, SSD, DVB-S2, Bitcoin, etc.</td>
<td>Berlekamp-Massey</td>
<td>Yes</td>
<td>100</td>
</tr>
<tr>
<td>Convol.</td>
<td>NASA</td>
<td>BCJR - Maximum A Posteriori (MAP)</td>
<td>No</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BCJR - Linear Approximation</td>
<td>No</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BCJR - Max Approximation</td>
<td>Yes</td>
<td>1000</td>
</tr>
</tbody>
</table>

Codecs come with puncturing patterns, optimized decoders and generic interleavers.
## Codecs

### List of Supported Channel Codes

<table>
<thead>
<tr>
<th>Channel code</th>
<th>Standard</th>
<th>Decoder</th>
<th>Fixed point</th>
<th>Throughput (Mb/s)</th>
</tr>
</thead>
</table>
| **LDPC**     | 5G, WiMAX, WiFi, DVB-S2, 10GE, etc. | Sum Product Algorithm (SPA)  
Min Sum (NMS, OMS)  
Approximate Min Star (AMS) | No          | 5                  |
|              |                              | Successive Cancellation (SC)  
Successive Cancellation List (SCL)  
Soft Cancellation (SCAN) | Yes         | 1000               |
|              | LTE (3G, 4G), DVB-RCS, CCSDS, etc. | Turbo BCJR  
Turbo BCJR + CRC  
Turbo BCJR + CRC + Flip aNd Check | Yes         | 100, 100           |
|              | CD, DVD, SSD, DVB-S2, Bitcoin, etc. | Berlekamp-Massey                      | Yes         | 100               |
| **BCH**      | NASA                         | BCJR - Maximum A Posteriori (MAP)  
BCJR - Linear Approximation  
BCJR - Max Approximation | No, No, Yes | 10, 50, 1000 |
| **Convol.**  | NASA                         | BCJR - Maximum A Posteriori (MAP)  
BCJR - Linear Approximation  
BCJR - Max Approximation | No, No, Yes | 10, 50, 1000 |

Codecs come with puncturing patterns, optimized decoders and generic interleavers.
## List of Supported Modulations/Demodulations

<table>
<thead>
<tr>
<th>Modem</th>
<th>Standard</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-PSK</td>
<td>IEEE 802.16 (WiMAX)</td>
<td>Phase-shift keying</td>
</tr>
<tr>
<td></td>
<td>UMTS (2G, 2G+)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EDGE (8-PSK), ...</td>
<td></td>
</tr>
<tr>
<td>N-QAM</td>
<td>IEEE 802.16 (WiMAX)</td>
<td>Quadrature amplitude modulation</td>
</tr>
<tr>
<td></td>
<td>UMTS (2G, 2G+)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3G, 4G, 5G, ...</td>
<td></td>
</tr>
<tr>
<td>N-PAM</td>
<td>IEEE 802.16 (WiMAX)</td>
<td>Pulse Amplitude Modulation</td>
</tr>
<tr>
<td></td>
<td>UMTS (2G, 2G+)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3G, 4G, 5G, ...</td>
<td></td>
</tr>
<tr>
<td>CPM</td>
<td>GMSK, Bluetooth</td>
<td>Continuous phase modulation</td>
</tr>
<tr>
<td></td>
<td>IEEE 802.11 FHSS</td>
<td>Coded (convolutional-based) modulation</td>
</tr>
<tr>
<td>OOK</td>
<td>IrDA (Infrared)</td>
<td>On-Off Keying</td>
</tr>
<tr>
<td></td>
<td>ISM bands</td>
<td>Used in optical communication systems</td>
</tr>
<tr>
<td>SCMA</td>
<td>-</td>
<td>Sparse Code Multiple Access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multi-user modulation</td>
</tr>
<tr>
<td>User defined</td>
<td>-</td>
<td>Constellation and order can be defined from an external file</td>
</tr>
</tbody>
</table>
## Simulator: Channels

### List of Supported Channels

<table>
<thead>
<tr>
<th>Channel</th>
<th>Mono-user</th>
<th>Multi-user</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWGN</td>
<td>Yes</td>
<td>Yes</td>
<td>Additive White Gaussian Noise</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>Yes</td>
<td>Yes</td>
<td>Flat Rayleigh fading channel</td>
</tr>
<tr>
<td>User defined</td>
<td>Yes</td>
<td>Not yet</td>
<td>User can import noise samples from an external file</td>
</tr>
</tbody>
</table>

*Efforts to optimize the code: Dedicated libraries are integrated.*

- GNU Scientific Library (GSL)
- Intel Math Kernel Library (MKL)
## Simulator: Channels

### List of Supported Channels

<table>
<thead>
<tr>
<th>Channel</th>
<th>Mono-user</th>
<th>Multi-user</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWGN</td>
<td>Yes</td>
<td>Yes</td>
<td>Additive White Gaussian Noise</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>Yes</td>
<td>Yes</td>
<td>Flat Rayleigh fading channel</td>
</tr>
<tr>
<td>User defined</td>
<td>Yes</td>
<td>Not yet</td>
<td>User can import noise samples from an external file</td>
</tr>
</tbody>
</table>

- Takes a **non-negligible part of the simulation time**
- Efforts to optimize the code
- Dedicated libraries are integrated
  - GNU Scientific Library (**GSL**)  
  - Intel Math Kernel Library (**MKL**)
AFF3CT simulator is designed for performance

- Compiled language (C++11)
- Many functions are optimized by hand
  - **SIMD** instructions (SSE, AVX, NEON)
  - Quantized implementations
  - Data layout
- Automatic and higher levels of parallelism
  - **Multi-threaded** (std::thread)
  - **Distributed** (MPI)
AFF3CT simulator is designed for performance

- Compiled language (C++11)
- Many functions are optimized by hand
  - SIMD instructions (SSE, AVX, NEON)
  - Quantized implementations
  - Data layout
- Automatic and higher levels of parallelism
  - Multi-threaded (std::thread)
  - Distributed (MPI)
Limitation

AFF3CT simulator comes with a finite set of possible communication chains.
**Limitation**

AFF3CT simulator comes with a finite set of possible communication chains.

**Solution**

AFF3CT can be compiled as a library or a Toolbox to fit to your specific applications.
AFF3CT simulator comes with a finite set of possible communication chains.

**Solution**

AFF3CT can be compiled as a library or a Toolbox to fit to your specific applications.

- A standard C++11 library
- All the AFF3CT elementary blocks can be reused
- **Easy to use**: take what you need from AFF3CT, leave the rest
AFF3CT has been designed to facilitate prototyping on FPGA
AFF3CT has been designed to facilitate prototyping on FPGA. Dedicated interfaces and implementations for UART and Ethernet.
AFF3CT has been designed to **facilitate prototyping on FPGA**

1. Dedicated interfaces and implementations for UART and Ethernet
2. Non intrusive parsing of the **AFF3CT** outputs into data files
PyBER is a generic visualization tool wrote in Python
PyBER is a generic visualization tool wrote in Python

- Plots BER/FER performance curves in live
PyBER is a generic visualization tool wrote in Python

- Plots BER/FER performance curves in live
- Compares various simulation performances
The AFF3CT project includes a database of simulation outputs.
The **AFF3CT** project includes a database of simulation outputs. Running commands are included to reproduce each simulation.
The AFF3CT project includes a database of simulation outputs. Running commands are included to reproduce each simulation. This database is used for automated testing scripts.
Miscellaneous: MIPP

- **SIMD C++ wrapper**
  - Maximizes code portability
  - *Improves expressiveness* over intrinsics
  - *No dependencies* to other libraries

Programming model close to intrinsics:
- Whenever possible, one statement = one intrinsic
- One variable = one register allocation

Open-source: https://github.com/aff3ct/MIPP
Miscellaneous: MIPP

- **SIMD C++ wrapper**
  - Maximizes code portability
  - **Improves expressiveness** over intrinsics
  - **No dependencies** to other libraries

- **Programming model close to intrinsics**
  - Whenever possible, **one statement = one intrinsic**
  - One variable = one register allocation

Open-source: [https://github.com/aff3ct/MIPP](https://github.com/aff3ct/MIPP)
Miscellaneous: MIPP

- SIMD C++ wrapper
  - Maximizes code portability
  - **Improves expressiveness** over intrinsics
  - **No dependencies** to other libraries
- Programming model **close to intrinsics**
  - Whenever possible, **one statement = one intrinsic**
  - One variable = one register allocation
- Open-source: [https://github.com/aff3ct/MIPP](https://github.com/aff3ct/MIPP)

MIPP MIPP!
Management of fixed point numbers in C++11

\[ y_{s,v} = \min(\max(2^v \cdot y \pm 0.5, -2^{s-1} + 1), 2^{s-1} - 1). \]
Management of fixed point numbers in C++11

\[ y_{s,v} = \min\left(\max\left(2^v \cdot y \pm 0.5, -2^{s-1} + 1\right), 2^{s-1} - 1\right). \]

- Defines a specific type for fixed point numbers: `fixed<S,V>`
  - `S`: the total number of bits
  - `V`: the number of bits for the decimal part
Management of fixed point numbers in C++11

\[ y_{s,v} = \min(\max(2^v \cdot y \pm 0.5, -2^{s-1} + 1), 2^{s-1} - 1). \]

- Defines a specific type for fixed point numbers: `fixed<S,V>`
  - \( S \): the total number of bits
  - \( V \): the number of bits for the decimal part
- Supports **arithmetic** operations (+, -, *, /, %)
- Supports **logic** operations (|, &, ^, ~)
Management of fixed point numbers in C++11

\[ y_{s,v} = \min(\max(2^v \cdot y \pm 0.5, -2^{s-1} + 1), 2^{s-1} - 1). \]

- Defines a specific type for fixed point numbers: `fixed<S,V>`
  - `S`: the total number of bits
  - `V`: the number of bits for the decimal part
- Supports **arithmetic** operations (+, −, *, /, %)
- Supports **logic** operations (|, &, ^, ~)
- Requires to know the quantification (S and V) at compile time
- Operations are optimized for speed
Plan

1 Introduction
   - Why AFF3CT?

2 State of Play
   - Simulator
   - Toolbox
   - Prototyping
   - Visualization
   - Miscellaneous

3 Simulation
   - What is a Simulation?
   - Launching Simulations

4 Development
   - Source Code Organization
   - Development in AFF3CT
   - My Project with AFF3CT

5 Contribution
   - Source Code Management
   - Add New Feature
   - Repositories
   - Continuous Integration

6 Roadmap and Discussion
   - What’s next?
What is a Simulation?

Running a simulation in AFF3CT with the minimal inputs:

$ aff3ct -C POLAR -K 1723 -N 2048 -m 1.0 -M 4.0 -s 1.0$

<table>
<thead>
<tr>
<th>Eb/N0dB</th>
<th>BER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>9.23x10^{-2}</td>
</tr>
<tr>
<td>2.0</td>
<td>7.05x10^{-2}</td>
</tr>
<tr>
<td>3.0</td>
<td>2.43x10^{-2}</td>
</tr>
<tr>
<td>4.0</td>
<td>2.93x10^{-4}</td>
</tr>
</tbody>
</table>
What is a Simulation?

Running a simulation in **AFF3CT** with the minimal inputs:

```
$ aff3ct -C POLAR -K 1723 -N 2048 -m 1.0 -M 4.0 -s 1.0
```

<table>
<thead>
<tr>
<th>Eb/N0dB</th>
<th>BER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>9.23x10^{-2}</td>
</tr>
<tr>
<td>2.0</td>
<td>7.05x10^{-2}</td>
</tr>
<tr>
<td>3.0</td>
<td>2.43x10^{-2}</td>
</tr>
<tr>
<td>4.0</td>
<td>2.93x10^{-4}</td>
</tr>
</tbody>
</table>

# ----------------------------------------------------------------------|
# Bit Error Rate (BER) and Frame Error Rate (FER) depending |
# on the Signal Noise Ratio (SNR) |
# ----------------------------------------------------------------------|
# -------|-------|----------|----------|----------|----------|----------|
# Es/N0 | Eb/N0 | FRA | BE | FE | BER | FER |
# (dB)  | (dB)  |     |    |    |     |     |
# -------|-------|----------|----------|----------|----------|----------|
| 0.25  | 1.00  | 100    | 15910    | 100      | 9.23e-02 | 1.00e+00 |
| 1.25  | 2.00  | 100    | 12151    | 100      | 7.05e-02 | 1.00e+00 |
| 2.25  | 3.00  | 129    | 5392     | 100      | 2.43e-02 | 7.75e-01 |
| 3.25  | 4.00  | 5467   | 2764     | 100      | 2.93e-04 | 1.83e-02 |
The SNR related inputs...

<table>
<thead>
<tr>
<th>C</th>
<th>K</th>
<th>N</th>
<th>m</th>
<th>M</th>
<th>S</th>
<th>Eb/N0 dB</th>
<th>BER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.0</td>
<td></td>
<td>s</td>
<td>9.23x10^-2</td>
<td>9.23x10^-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.0</td>
<td></td>
<td>s</td>
<td>7.05x10^-2</td>
<td>7.05x10^-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3.0</td>
<td></td>
<td>s</td>
<td>2.43x10^-2</td>
<td>2.43x10^-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4.0</td>
<td></td>
<td>s</td>
<td>2.93x10^-4</td>
<td>2.93x10^-4</td>
</tr>
</tbody>
</table>
Channel Code Arguments

... and the code related inputs.

\[ \mathbf{C} = \{ \text{POLAR, LDPC, TURBO, ...} \} \]
Modules and Tasks

Simulation

Source:
Channel:
Codec:
Modem:
Monitor:

generate modulate encode
add_noise

code demodulate check_errors

module task

A. Cassagne, O. Hartmann, M. Léonardon, C. Leroux, C. Jégo
IMS, Inria, U. of Bordeaux
AFF3CT Seminary
Module Arguments

Now, let’s dig into the arguments with help mode (-h) ...

$ aff3ct -C TURBO -K 1024 -m 0 -M 0 -h
Module Arguments

Now, let’s dig into the arguments with help mode (-h) ...

$ aff3ct -C TURBO -K 1024 -m 0 -M 0 -h

Each module or task has its own set of arguments, e.g. the encoder:

Encoder parameter(s):

{R} --enc-info-bits, -K <integer:positive, non-zero>
useful number of bit transmitted (information bits).

--enc-json-path <file [write only]>
path to store the encoder and decoder traces formatted in JSON.

--enc-path <file [read only]>
path to a file containing one or a set of pre-computed codewords, to use with "--enc-type USER".

The {R} tag denotes required argument for a given code and simulation.
Module Arguments

Now, let’s dig into the arguments with help mode (-h) ...

$ aff3ct -C TURBO -K 1024 -m 0 -M 0 -h

Each module or task has its own set of arguments, e.g. the encoder:

Encoder parameter(s):
{R} --enc-info-bits, -K <integer:positive, non-zero>
useful number of bit transmitted (information bits).
--enc-json-path <file [write only]>
path to store the encoder and decoder traces formatted in JSON.
--enc-path <file [read only]>
path to a file containing one or a set of pre-computed codewords, to use
with "--enc-type USER".

The {R} tag denotes required argument for a given code and simulation.

Using the uppercase (-H) argument, advanced arguments are shown, denoted with a {A} tag.

{A} --sim-no-legend
Do not display any legend when launching the simulation.
Each **module** or **task** has its own set of arguments. Still, some of the arguments are common to several **modules** and **tasks**:

- **--xxx-type** is often used to define the type of each **module**: the type of modulation, channel or channel decoder.

  $\texttt{aff3ct -C POLAR -m 1 -M 4 -K 1723 -N 2048 --mdm-type BPSK}$
Each **module** or **task** has its own set of arguments. Still, some of the arguments are common to several **modules** and **tasks**:

- `--xxx-type` is often used to define the type of each **module**: the type of modulation, channel or channel decoder.

  ```bash
  $ aff3ct -C POLAR -m 1 -M 4 -K 1723 -N 2048 --mdm-type BPSK
  ```

- `--xxx-implem` specifies the type of implementation used. The keyword **NAIVE** is often used to denote a readable but unoptimized source code, whereas a **FAST** stands for a source code that is optimized for a high throughput.

  ```bash
  $ aff3ct -C POLAR -m 1 -M 4 -K 1723 -N 2048 --dec-implem FAST
  ```
Debug Arguments

Other arguments allow the user to get advances information about the running simulation:

- `--sim-debug` or `-d` enables the printing of the inputs and outputs of each task,
- `--sim-debug-limit` limits the number of elements displayed in the debug information,
- `--sim-debug-prec` sets the precision of the real values displayed in the debug information,
- `--sim-debug-hex` sets the format of the real values to hexadecimal, which enables for example to extract the exact value of a floating-point number.
Debug Arguments

Here is an example of a debug output frame:

```c
# New communication (n°99)
# Channel_AWGN_LLR::add_noise(const float32 X_N[3084], float32 Y_N[3084])
# {IN}  X_N = [ 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, ...]
# {OUT} Y_N = [ 0.37, 0.06, 2.50, -0.32, 2.49, -0.45, 0.91, 0.77, 2.13, 0.83, ...]
# Returned status: 0
#
# Modem_BPSK::demodulate(const float32 Y_N1[3084], float32 Y_N2[3084])
# {IN}  Y_N1 = [ 0.37, 0.06, 2.50, -0.32, 2.49, -0.45, 0.91, 0.77, 2.13, 0.83, ...]
# {OUT} Y_N2 = [ 0.49, 0.07, 3.32, -0.43, 3.31, -0.59, 1.21, 1.02, 2.83, 1.11, ...]
# Returned status: 0
#
# Decoder_turbo_fast::decode_siho(const float32 Y_N[3084], int32 V_K[1024])
# {IN}  Y_N = [ 0.49, 0.07, 3.32, -0.43, 3.31, -0.59, 1.21, 1.02, 2.83, 1.11, ...]
# {OUT} V_K = [ 0, 0, 0, 0, 1, 0, 0, 0, 1, ...]
# Returned status: 0
#
# Monitor_BFER::check_errors(const int32 U[1024], const int32 V[1024])
# {IN}  U = [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...]
# {IN}  V = [ 0, 0, 0, 0, 1, 0, 0, 0, 1, ...]
# Returned status: 165
```
Statistics Argument

The `--sim-stats` offers interesting arguments concerning the time consumed by each task and the corresponding throughputs and latencies.

<table>
<thead>
<tr>
<th>MODULE</th>
<th>TASK</th>
<th>TIME (s)</th>
<th>PERC (%)</th>
<th>CALLS</th>
<th>TIME (s)</th>
<th>PERC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td>add_noise</td>
<td>0.48</td>
<td>38.70</td>
<td>5533</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>generate</td>
<td>0.35</td>
<td>28.37</td>
<td>5533</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoder</td>
<td>encode</td>
<td>0.27</td>
<td>21.67</td>
<td>5533</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decoder</td>
<td>decode Siho</td>
<td>0.13</td>
<td>18.51</td>
<td>5533</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitor</td>
<td>check_errors</td>
<td>0.80</td>
<td>6.35</td>
<td>5533</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modem</td>
<td>modulate</td>
<td>0.00</td>
<td>0.21</td>
<td>5533</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modem</td>
<td>demodulate</td>
<td>0.00</td>
<td>0.19</td>
<td>5533</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>1.23</td>
<td>106.00</td>
<td>5533</td>
<td>7.75</td>
<td>0.14</td>
</tr>
</tbody>
</table>

Measured throughput:

- Average: 221.37 ± 190.79 (12242.29)
- Minimum: 86.05 ± 79.00 (316.53)
- Maximum: 86.05 ± 79.00 (316.53)

Measured latency:

- Average: 221.37 ± 190.79 (12242.29)
- Minimum: 86.05 ± 79.00 (316.53)
- Maximum: 86.05 ± 79.00 (316.53)
Statistics Argument

The `--sim-stats` offers interesting arguments concerning the time consumed by each task and the corresponding throughputs and latencies.

- The measured latency of each task includes the time needed to read the input socket and write the output socket (single threaded)
The `--sim-stats` offers interesting arguments concerning the time consumed by each task and the corresponding throughputs and latencies.

- The measured latency of each task includes the time needed to read the input socket and write the output socket (*single threaded*).
- The number of bits $N_b$ of a task is the size of its last output socket.
The **--sim-stats** offers interesting arguments concerning the time consumed by each **task** and the corresponding **throughputs** and **latencies**.

- The measured **latency** of each **task** includes the time needed to read the input socket and write the output socket (**single threaded**).
- The number of bits $N_b$ of a **task** is the size of its last output socket.
- **Throughput** $= \frac{N_b}{\text{latency}}$
Multi-threading is a simple way to improve the speed of Monte-Carlo simulations. The number of threads can be manually defined with the \texttt{-t} option (by default AFF3CT runs with all the available threads):

- Simulation on Rahan (2\times Intel® Xeon® E5-2690 v3)

\begin{verbatim}
$ aff3ct -C POLAR -m 1 -M 4 -K 1723 -N 2048 -t 48
\end{verbatim}
Multi-threading is a simple way to improve the speed of Monte-Carlo simulations. The number of threads can be manually defined with the \( -t \) option (by default AFF3CT runs with all the available threads):

- Simulation on Rahan (2\( \times \) Intel\(^\text{®} \) Xeon\(^\text{®} \) E5-2690 v3)
  
  $\text{aff3ct -C POLAR -m 1 -M 4 -K 1723 -N 2048 -t 48}$

For very computational intensive simulations, AFF3CT can also be distributed on multiple servers using the MPI standard, thus increasing the simulation throughput by the number of instances:

- Simulation on OCCIGEN at CINES (4212\( \times \) Intel\(^\text{®} \) Xeon\(^\text{®} \) E5-2690 v3)
  
  $\text{mpirun -np 64 aff3ct -C POLAR -m 1 -M 4 -K 1723 -N 2048 -t 48}$
In order to improve channel coding, it can be useful to **track the erroneous frames** that occurred in a simulation. In **AFF3CT**, it is possible to **dump this erroneous frames in files**, in order to run them again later.
In order to improve channel coding, it can be useful to track the erroneous frames that occurred in a simulation. In AFF3CT, it is possible to dump this erroneous frames in files, in order to run them again later.

The corresponding arguments are the following:

- `--sim-err-trk` dumps the erroneous frames,
- `--sim-err-trk-rev` replays the erroneous frames,
- `--sim-err-trk-path` selects the path of the folder in which the corresponding dump files are stored.
Error Tracking

In order to improve channel coding, it can be useful to track the erroneous frames that occurred in a simulation. In AFF3CT, it is possible to dump this erroneous frames in files, in order to run them again later.

The corresponding arguments are the following:

- `--sim-err-trk` dumps the erroneous frames,
- `--sim-err-trk-rev` replays the erroneous frames,
- `--sim-err-trk-path` selects the path of the folder in which the corresponding dump files are stored.

This is very useful when working on the code error floor!
Some **modules** can deal with multiple frames simultaneously:
Some **modules** can deal with multiple frames simultaneously:

- In order to increase the parallelism with SIMD instructions (MIPP)

  $$\texttt{aff3ct -C POLAR -m 1 -M 4 -K 1723 -N 2048 -F 3 --dec-simd INTER}$$
Some **modules** can deal with multiple frames simultaneously:

In order to increase the parallelism with SIMD instructions (MIPP)

```
$ aff3ct -C POLAR -m 1 -M 4 -K 1723 -N 2048 -F 3 --dec-simd INTER
```

In a multi user context, e.g. multiple access methods like Sparse Code Multiple Access (SCMA)

```
$ aff3ct -C UNCODED -m 0 -M 14 -K 12 -mdm-type SCMA -mdm-ite 6 -F 6
```
Source Code Organization

Folders in the AFF3CT Root

- **ci/** Continuous integration scripts
- **conf/** Input configuration files for the simulator
- **doc/** Documentation related files (Doxygen)
- **lib/** Libraries used by AFF3CT (like MIPP)
- **refs/** Reference curves, simulated results
- **scripts/** Miscellaneous scripts like the debug parser
- **src/** AFF3CT source code
Source Code Organization

Folders in the src/ Directory

Factory/ Manages the command line arguments, builds the objects (Launcher, Simulation, Module and Tools)
Source Code Organization

Folders in the src/ Directory

**Factory/** Manages the command line arguments, builds the objects (Launcher, Simulation, Module and Tools)

**Launcher/** Composes the Simulation with Modules
Source Code Organization

**Folders in the `src/` Directory**

- **Factory/** Manages the command line arguments, builds the objects (Launcher, Simulation, Module and Tools)
- **Launcher/** Composes the Simulation with Modules
- **Simulation/** The BER/FER (std and ite) and EXIT charts simulation chains
Source Code Organization

Folders in the src/ Directory

- **Factory/** Manages the command line arguments, builds the objects (Launcher, Simulation, Module and Tools)
- **Launcher/** Composes the Simulation with Modules
- **Simulation/** The BER/FER (std and ite) and EXIT charts simulation chains
- **Module/** Blocks of the communication chain (Source, Codec, Modem, Channel, ...)

A. Cassagne, O. Hartmann, M. Léonardon, C. Leroux, C. Jégo
Source Code Organization

Folders in the *src/* Directory

- **Factory/** Manages the command line arguments, builds the objects (Launcher, Simulation, Module and Tools)
- **Launcher/** Composes the Simulation with Modules
- **Simulation/** The BER/FER (std and ite) and EXIT charts simulation chains
- **Module/** Blocks of the communication chain (Source, Codec, Modem, Channel, ...)
- **Tools/** Elementary functions that can be reused in all the whole code
Coding Style

- **Indentation with tabulations** (hard tabs)

```cpp
Function register()
{
    if (empty_PORTS)
    {
        msg = "";
        if (empty_ports["user_name"])
        {
            if (empty_ports["user_password"])
            {
                if (empty_ports["user_password"] < empty_ports["user_password_repeat"])
                {
                    if (empty_ports["user_name"] < 0)
                    {
                        if (empty_ports["user_name"] > 1)
                        {
                            if (empty_ports["user_name"] < 1)
                            {
                                if (empty_ports["user_email"] == empty_ports["user_email"])
                                {
                                    if (empty_ports["user_password"] == empty_ports["user_password"])
                                    {
                                        "You are now registered in please login:
                                        number 'location': - 0_0000_00_00"
                                        "Have a good day!
                                        else msg = "User already exist!
                                        }else msg = "Password must be at least 6 characters!
                                        }else msg = "Username must be at least 6 characters!
                                        }else msg = "Password must be at least 6 characters!
                                        }else msg = "Invalid Password!
                                        }else msg = "Invalid Username!
                                        }else msg = "Title is not allowed!
                                        }else msg = "Title is not allowed!
                                        returnRegister();
                                        
```
Coding Style

- **Indentation with tabulations** (hard tabs)
- **Spaces as a separator**
  - Item list: `[1, 2, 3]` instead of `[1,2,3]`
  - Operators: `x += 2` instead of `x+=2`

```cpp
// Example code snippet

int main() {
    if (something() {
        if (do_something()) {
            // Code
        }
    }
    // More code
    return 0;
}
```
Coding Style

- **Indentation with tabulations** (hard tabs)
- **Spaces as a separator**
  - Item list: `[1, 2, 3]` instead of `[1,2,3]`
  - Operators: `x += 2` instead of `x+=2`
- **Maximum of 120 characters per line** of code
Coding Style

- Indentation with tabulations (hard tabs)
- Spaces as a separator
  - Item list: `[1, 2, 3]` instead of `[1,2,3]`
  - Operators: `x += 2` instead of `x+=2`
- Maximum of 120 characters per line of code
- Snake case
  - Variables: `my_variable` instead of `myVariable`
  - Classes: `My_class` instead of `My_Class`
Coding Style

- **Indentation with tabulations (hard tabs)**
- **Spaces as a separator**
  - Item list: `[1, 2, 3]` instead of `[1,2,3]`
  - Operators: `x += 2` instead of `x+=2`
- **Maximum of 120 characters per line** of code
- **Snake case**
  - Variables: `my_variable` instead of `myVariable`
  - Classes: `My_class` instead of `My_Class`
- **File naming conventions**
  - Class file names: `Decoder_repetition.hpp`
  - Function file names: `numerical_integration.h`
  - Compiled source files: always take a `.cpp` extension
  - Template source files: always take a `.hxx` extension
Coding Style

- **Indentation with tabulations** (hard tabs)
- **Spaces as a separator**
  - Item list: `[1, 2, 3]` instead of `[1,2,3]`
  - Operators: `x += 2` instead of `x+=2`
- **Maximum of 120 characters per line** of code
- **Snake case**
  - Variables: `my_variable` instead of `myVariable`
  - Classes: `My_class` instead of `My_Class`
- **File naming conventions**
  - Class file names: `Decoder_repetition.hpp`
  - Function file names: `numerical_integration.h`
  - Compiled source files: always take a `.cpp` extension
  - Template source files: always take a `.hxx` extension
- **Acronyms stay in upper case**: `Decoder_LDPC.cpp`, `Channel_AWGN`
Generating documentation (commands for a Debian-like OS)

$ sudo apt-get install doxygen
$ cd aff3ct_path/doc/
$ doxygen config.txt
$ xdg-open doc/html/index.html
Let’s suppose we want to add a new Modem named 2PSK.
Let’s suppose we want to add a new Modem named 2PSK

1. Identify the type of module we want to add (Source, Codec, Modem, Channel, ...), our new class will inherit from it.
Adding a New Module in AFF3CT

Let’s suppose we want to add a new Modem named 2PSK

1. **Identify the type of module** we want to add (Source, Codec, Modem, Channel, ...), our new class will inherit from it

2. **Add the implementation in AFF3CT**
   - Create a new folder: `src/Module/Modem/2PSK`
   - Create new files: `Modem_2PSK.hpp` and the `Modem_2PSK.cpp`
Adding a New Module in AFF3CT

Let’s suppose we want to add a new **Modem** named **2PSK**

1. **Identify the type of module** we want to add (Source, Codec, Modem, Channel, ...), our new class will inherit from it
2. **Add the implementation in AFF3CT**
   - Create a new folder: `src/Module/Modem/2PSK`
   - Create new files: `Modem_2PSK.hpp` and the `Modem_2PSK.cpp`
3. **Link the new code with the simulator:**
   - Add a new entry in the **Modem** factory
   - File: `src/Factory/Module/Modem/Modem.cpp`
Adding a New Module in AFF3CT

Let’s suppose we want to add a new Modem named 2PSK

1. **Identify the type of module** we want to add (Source, Codec, Modem, Channel, ...), our new class will inherit from it

2. **Add the implementation in AFF3CT**
   - Create a new folder: `src/Module/Modem/2PSK`
   - Create new files: `Modem_2PSK.hpp` and the `Modem_2PSK.cpp`

3. **Link the new code with the simulator:**
   - Add a new entry in the `Modem` factory
   - File: `src/Factory/Module/Modem/Modem.cpp`

4. **Run AFF3CT and simulate** with our new module
Let's create `src/Module/Modem/2PSK/Modem_2PSK.hpp`:

```cpp
#ifndef MODEM_2PSK_HPP_
define MODEM_2PSK_HPP_

#include "Module/Modem/Modem.hpp"

namespace aff3ct{ namespace module{

// B is the type of bits, R and Q are the types of real data
template <typename B = int, typename R = float, typename Q = R>
// inherits from Modem class
class Modem_2PSK : public Modem<B,R,Q>
{
    // class constructor and destructor
    public:
    Modem_2PSK(const int N, const R sigma) : Modem<B,R,Q>(N, sigma, 1) {}
    virtual ~Modem_2PSK() {}

    // specific modem interface (= modem tasks)
    protected:
    void _modulate(const B *X_N1, R *X_N2, const int frame_id);
    void _filter(const R *Y_N1, R *Y_N2, const int frame_id);
    void _demodulate(const Q *Y_N1, Q *Y_N2, const int frame_id);
};

#endif /* MODEM_2PSK_HPP_ */
```
Let’s create src/Module/Modem/2PSK/Modem_2PSK.cpp:

```cpp
#include "Modem_2PSK.hpp"

using namespace aff3ct::module;

template <typename B, typename R, typename Q>
void Modem_2PSK<B,R,Q>::_modulate(const B *X_N1, R *X_N2, const int frame_id){
    for (auto i = 0; i < this->N; i++)
        X_N2[i] = (X_N1[i] == 1) ? -1 : +1;
}

template <typename B,typename R, typename Q>
void Modem_2PSK<B,R,Q>::_filter(const R *Y_N1, R *Y_N2, const int frame_id){
    std::copy(Y_N1, Y_N1 + this->N_mod, Y_N2); // no filtering here so copy
}

template <typename B, typename R, typename Q>
void Modem_2PSK<B,R,Q>::_demodulate(const Q *Y_N1, Q *Y_N2, const int frame_id){
    for (auto i = 0; i < this->N_fil; i++)
        Y_N2[i] = 2 * Y_N1[i] / (this->sigma * this->sigma);
}
```
Example: Adding a Modulator, Modem_2PSK.cpp

Let's create src/Module/Modem/2PSK/Modem_2PSK.cpp:

```cpp
#include "Modem_2PSK.hpp"

using namespace aff3ct::module;

template <typename B, typename R, typename Q>
void Modem_2PSK<B,R,Q>::_modulate(const B *X_N1, R *X_N2, const int frame_id){
    for (auto i = 0; i < this->N; i++)
        X_N2[i] = (X_N1[i] == 1) ? -1 : +1;
}

template <typename B,typename R, typename Q>
void Modem_2PSK<B,R,Q>::_filter(const R *Y_N1, R *Y_N2, const int frame_id){
    std::copy(Y_N1, Y_N1 + this->N_mod, Y_N2); // no filtering here so copy
}

template <typename B, typename R, typename Q>
void Modem_2PSK<B,R,Q>::_demodulate(const Q *Y_N1, Q *Y_N2, const int frame_id){
    for (auto i = 0; i < this->N_fil; i++)
        Y_N2[i] = 2 * Y_N1[i] / (this->sigma * this->sigma);
}
```

The **Modem** parent class contains the current sigma value (**this->sigma**) as well as the frame lengths (**this->N**, **this->N_mod**, **this->N_fil**).
Example: Adding a Modulator to the Factory

In the src/Factory/Module/Modem/Modem.cpp file:

```cpp
#include "Module/Modem/2PSK/Modem_2PSK.hpp"

void Modem::parameters::get_description(tools::Argument_map_info &args) const {
    args.add(
        {p+"-type"},
        tools::Text(tools::Including_set("...", "2PSK")),
        "type of the modulation to use in the simulation.");

    template <typename B, typename R, typename Q, tools::proto_max<Q> MAX>
    module::Modem<B,R,Q>* Modem::parameters::_build() const {
        // ...
        else if (this->type == "2PSK")
            return new module::Modem_2PSK<B,R,Q>(this->N, this->sigma);
        // ...
    }
```
Example: Adding a Modulator to the Factory

In the src/Factory/Module/Modem/Modem.cpp file:

```cpp
// ...
int Modem
::get_buffer_size_after_modulation(const std::string &type,
    const int N,
    const int bps,
    const int upf,
    const int cpm_L,
    const int cpm_p)
{
    // ...
    else if (type == "2PSK") return N;
    // ...
}

int Modem
::get_buffer_size_after_filtering(const std::string &type,
    const int N,
    const int bps,
    const int cpm_L,
    const int cpm_p)
{
    // ...
    else if (type == "2PSK") return N;
    // ...
}
// ...
```
Example: Run the New Modem

Compile and Run

$ cd aff3ct_path/build/
$ cmake .
$ make -j4

Live AFF3CT simulation on PyBER

A. Cassagne, O. Hartmann, M. Léonardon, C. Leroux, C. Jégo
Example: Run the New Modem

Compile and Run

$ cd aff3ct_path/build/
$ cmake .
$ make -j4
$ ./bin/aff3ct -C UNCODED -m 0 -M 4 -K 2048 --mdm-type 2PSK --sim-pyber "My modem" > pyber_path/data/test.txt
Example: Run the New Modem

Compile and Run

$ cd aff3ct_path/build/
$ cmake .
$ make -j4
$ ./bin/aff3ct -C UNCODED -m 0 -M 4 -K 2048 --mdm-type 2PSK --sim-pyber "My modem" > pyber_path/data/test.txt

Live AFF3CT simulation on PyBER
Example: Run the New Modem

Compile and Run

$ cd aff3ct_path/build/
$ cmake .
$ make -j4
$ ./bin/aff3ct -C UNCODED -m 0 -M 4 -K 2048 --mdm-type 2PSK --sim-pyber "My modem" > pyber_path/data/test.txt

Live AFF3CT simulation on PyBER
Example: Run the New Modem

Compile and Run

$ cd aff3ct_path/build/
$ cmake .
$ make -j4
$ ./bin/aff3ct -C UNCODED -m 0 -M 4 -K 2048 --mdm-type 2PSK --sim-pyber "My modem" > pyber_path/data/test.txt

Live AFF3CT simulation on PyBER
Example: Run the New Modem

Compile and Run

$ cd aff3ct_path/build/
$ cmake .
$ make -j4
$ ./bin/aff3ct -C UNCODED -m 0 -M 4 -K 2048 --mdm-type 2PSK --sim-pyber "My modem" > pyber_path/data/test.txt

Live AFF3CT simulation on PyBER
My Project with AFF3CT

Use AFF3CT as a **Toolbox / library** for your projects

- **Low level features**
  - **Code-related**: Alist/QC readers, Polar functions API, frozen bits generators, Galois Field generator, ...
  - **Miscellaneous**: Matrix operations, sparse matrices, binary trees, ...

A. Cassagne, O. Hartmann, M. Léonardon, C. Leroux, C. Jégo

AFF3CT Seminary
My Project with AFF3CT

Use AFF3CT as a **Toolbox** / **library** for your projects

- **Low level features**
  - **Code-related**: Alist/QC readers, Polar functions API, frozen bits generators, Galois Field generator, ...
  - **Miscellaneous**: Matrix operations, sparse matrices, binary trees, ...

- **High level features**
  - **Factories**: Command line arguments management, automatic objects instantiation, ...
  - **Modules**: Sources, CRC, encoders, decoders, modems, channels, ...
#include <aff3ct.hpp>
using namespace aff3ct;

// allocate the module objects
# include <aff3ct.hpp>
using namespace aff3ct;

// allocate the module objects
module::Source_random<> source (K );
#include <aff3ct.hpp>
using namespace aff3ct;

// allocate the module objects
module::Source_random<> source(K);
module::Encoder_repetition_sys<> encoder(K, N);
# include <aff3ct.hpp>
using namespace aff3ct;

// allocate the module objects
module::Source_random<> source ( K );
module::Encoder_repetition_sys<> encoder(K, N);
module::Modem_BPSK<> modem ( N );
My Project with **AFF3CT**: Modules Allocation

```cpp
#include <aff3ct.hpp>
using namespace aff3ct;

// allocate the module objects
module::Source_random<> source (K );
module::Encoder_repetition_sys<> encoder(K, N);
module::Modem_BPSK<> modem (N );
module::Channel_AWGN_LLR<> channel(N );
```

A. Cassagne, O. Hartmann, M. Léonardon, C. Leroux, C. Jégo
AFF3CT Seminary
IMS, Inria, U. of Bordeaux
My Project with **AFF3CT**: Modules Allocation

```cpp
#include <aff3ct.hpp>
using namespace aff3ct;

// allocate the module objects
module::Source_random<> source (K);
module::Encoder_repetition_sys<> encoder(K, N);
module::Modem_BPSK<> modem (N);
module::Channel_AWGN_LLR<> channel(N);
module::Decoder_repetition_std<> decoder(K, N);
```
My Project with AFF3CT: Modules Allocation

```cpp
#include <aff3ct.hpp>
using namespace aff3ct;

// allocate the module objects
module::Source_random<> source(K);
module::Encoder_repetition_sys<> encoder(K, N);
module::Modem_BPSK<> modem(N);
module::Channel_AWGN_LLRLR<> channel(N);
module::Decoder_repetition_std<> decoder(K, N);
module::Monitor_BFER<> monitor(K, E);
```
#include <aff3ct.hpp>
using namespace aff3ct;

// allocate the module objects
module::Source_random<>
moder (K );
module::Encoder_repetition_sys<>
encoder(K, N);
module::Modem_BPSK<>
modem (N );
module::Channel_AWGN_LLR<>
channel(N );
module::Decoder_repetition_std<>
decoder(K, N);
module::Monitor_BFER<>
monitor(K, E);
using namespace aff3ct::module;

// bind the sockets over the tasks
using namespace aff3ct::module;

// bind the sockets over the tasks
encoder[enc::sck::encode ::U_K ].bind( source [src::sck::generate ::U_K ] );
using namespace aff3ct::module;

// bind the sockets over the tasks
encoder[enc::sck::encode ::U_K].bind( source [src::sck::generate ::U_K ] );
modem [mdm::sck::modulate ::X_N1].bind( encoder[enc::sck::encode ::X_N ] );
My Project with AFF3CT: Sockets Binding

```
using namespace aff3ct::module;

// bind the sockets over the tasks
encoder[enc::sck::encode ::U_K].bind( source [src::sck::generate ::U_K ]);
modem [mdm::sck::modulate ::X_N1].bind( encoder[enc::sck::encode ::X_N ]);
channel[chn::sck::add_noise ::X_N ].bind( modem [mdm::sck::modulate ::X_N2] );
```
My Project with AFF3CT: Sockets Binding

```
using namespace aff3ct::module;

// bind the sockets over the tasks
encoder[enc::sck::encode ::U_K].bind( source [src::sck::generate ::U_K ] );
modem [mdm::sck::modulate ::X_N1].bind( encoder[enc::sck::encode ::X_N ] );
channel[chn::sck::add_noise ::X_N ].bind( modem [mdm::sck::modulate ::X_N2] );
modem [mdm::sck::demodulate ::Y_N1].bind( channel[chn::sck::add_noise ::Y_N ] );
```
using namespace aff3ct::module;

// bind the sockets over the tasks
encoder[enc::sck::encode ::U_K].bind( source [src::sck::generate ::U_K ] );
modem [mdm::sck::modulate ::X_N1].bind( encoder[enc::sck::encode ::X_N ] );
channel[chn::sck::add_noise ::X_N].bind( modem [mdm::sck::modulate ::X_N2] );
modem [mdm::sck::demodulate ::Y_N1].bind( channel[chn::sck::add_noise ::Y_N ] );
decoder[dec::sck::decode_siho ::Y_N].bind( modem [mdm::sck::demodulate ::Y_N2] );
using namespace aff3ct::module;

// bind the sockets over the tasks
encoder[enc::sck::encode ::U_K].bind( source [src::sck::generate ::U_K ] );
modem [mdm::sck::modulate ::X_N1].bind( encoder[enc::sck::encode ::X_N ] );
channel[chn::sck::add_noise ::X_N ].bind( modem [mdm::sck::modulate ::X_N2] );
modem [mdm::sck::demodulate ::Y_N1].bind( channel[chn::sck::add_noise ::Y_N ] );
decoder[dec::sck::decode_siho ::Y_N ].bind( modem [mdm::sck::demodulate ::Y_N2] );
monitor[mnt::sck::check_errors::U ].bind( source [src::sck::generate ::U_K ] );
using namespace aff3ct::module;

// bind the sockets over the tasks
encoder[enc::sck::encode ::U_K ].bind( source [src::sck::generate ::U_K ] );
modem [mdm::sck::modulate ::X_N1].bind( encoder[enc::sck::encode ::X_N ] );
channel[chn::sck::add_noise ::X_N ].bind( modem [mdm::sck::modulate ::X_N2] );
modem [mdm::sck::demodulate ::Y_N1].bind( channel[chn::sck::add_noise ::Y_N ] );
decoder[dec::sck::decode_siho ::Y_N ].bind( modem [mdm::sck::demodulate ::Y_N2] );
monitor[mnt::sck::check_errors::U ].bind( source [src::sck::generate ::U_K ] );
monitor[mnt::sck::check_errors::V ].bind( decoder[dec::sck::decode_siho::V_K ] );
using namespace aff3ct::module;

// bind the sockets over the tasks
encoder[enc::sck::encode ::U_K ].bind( source [src::sck::generate ::U_K ] );
modem [mdm::sck::modulate ::X_N1].bind( encoder[enc::sck::encode ::X_N ] );
channel[chn::sck::add_noise ::X_N ].bind( modem [mdm::sck::modulate ::X_N2] );
modem [mdm::sck::demodulate ::Y_N1].bind( channel[chn::sck::add_noise ::Y_N ] );
decoder[dec::sck::decode_siho ::Y_N ].bind( modem [mdm::sck::demodulate ::Y_N2] );
monitor[mnt::sck::check_errors::U ].bind( source [src::sck::generate ::U_K ] );
monitor[mnt::sck::check_errors::V ].bind( decoder[dec::sck::decode_siho::V_K ] );
My Project with AFF3CT: Tasks Execution

```cpp
// the simulation loop
while (!monitor.fe_limit_achieved()) {
```

A. Cassagne, O. Hartmann, M. Léonardon, C. Leroux, C. Jégo

IMS, Inria, U. of Bordeaux

AFF3CT Seminary
My Project with **AFF3CT**: Tasks Execution

```cpp
// the simulation loop
while (!monitor.fe_limit_achieved()) {
    source[src::tsk::generate].exec();
}
```
My Project with AFF3CT: Tasks Execution

```cpp
// the simulation loop
while (!monitor.fe_limit_achieved()) {
    source[src::tsk::generate].exec();
    encoder[enc::tsk::encode].exec();
}
```

Green: module
- task
- input socket
- output socket
My Project with AFF3CT: Tasks Execution

```cpp
// the simulation loop
while (!monitor.fe_limit_achieved()) {
    source [src::tsk::generate ].exec();
    encoder[enc::tsk::encode ].exec();
    modem [mdm::tsk::modulate ].exec();
}
```
My Project with **AFF3CT**: Tasks Execution

```cpp
1 // the simulation loop
2 while (!monitor.fe_limit_achieved()) {
3    source [src::tsk::generate ].exec();
4    encoder[enc::tsk::encode ].exec();
5    modem [mdm::tsk::modulate ].exec();
6    channel[chn::tsk::add_noise ].exec();
7 }
```
// the simulation loop
while (!monitor.fe_limit_achieved()) {
    source [src::tsk::generate ].exec();
    encoder[enc::tsk::encode ].exec();
    modem [mdm::tsk::modulate ].exec();
    channel[chn::tsk::add_noise ].exec();
    modem [mdm::tsk::demodulate ].exec();
}
// the simulation loop
while (!monitor.fe_limit_achieved()) {
    source [src::tsk::generate ].exec();
    encoder[enc::tsk::encode ].exec();
    modem [mdm::tsk::modulate].exec();
    channel[chn::tsk::add_noise ].exec();
    modem [mdm::tsk::demodulate].exec();
    decoder[dec::tsk::decode_siho].exec();
}

A. Cassagne, O. Hartmann, M. Léonardon, C. Leroux, C. Jégo
IMS, Inria, U. of Bordeaux
AFF3CT Seminary
My Project with AFF3CT: Tasks Execution

```cpp
// the simulation loop
while (!monitor.fe_limit_achieved()) {
    source [src::tsk::generate ].exec();
    encoder[enc::tsk::encode ].exec();
    modem [mdm::tsk::modulate ].exec();
    channel[chn::tsk::add_noise ].exec();
    modem [mdm::tsk::demodulate ].exec();
    decoder[dec::tsk::decode_siho ].exec();
    monitor[mnt::tsk::check_errors].exec();
}
```
My Project with AFF3CT: Tasks Execution

```cpp
// the simulation loop
while (!monitor.fe_limit_achieved()) {
    source [src::tsk::generate ].exec();
    encoder[enc::tsk::encode ].exec();
    modem [mdm::tsk::modulate ].exec();
    channel[chn::tsk::add_noise ].exec();
    modem [mdm::tsk::demodulate ].exec();
    decoder[dec::tsk::decode_siho ].exec();
    monitor[mnt::tsk::check_errors].exec();
}
```
Plan

1 Introduction
   - Why AFF3CT?

2 State of Play
   - Simulator
   - Toolbox
   - Prototyping
   - Visualization
   - Miscellaneous

3 Simulation
   - What is a Simulation?
   - Launching Simulations

4 Development
   - Source Code Organization
   - Development in AFF3CT
   - My Project with AFF3CT

5 Contribution
   - Source Code Management
   - Add New Feature
   - Repositories
   - Continuous Integration

6 Roadmap and Discussion
   - What’s next?
Motivation

Sharing source code between team members can be very complicated without the appropriate tools.
Source Code Management: **Git**

- **Git centralizes the source code** in a repository
- **Git manages the transactions** between the developers
Source Code Management: **Git**

- **Git centralizes the source code** in a repository
- **Git manages the transactions** between the developers
- Each developer:
  - Has a local repository (**git clone** or **git init**)
  - Can save new features in its local repository (**git commit**)
  - Can update its local repository from the centralized one (**git pull**)
  - Can update the centralized repository from its local one (**git push**)
Two special and **public** branches: **master** and **development**
- **master**: stable branch for releases, conservative interfaces
- **development**: integration of the new features
Two special and **public** branches: **master** and **development**
- **master**: stable branch for releases, conservative interfaces
- **development**: integration of the new features

New feature = new **feature** branch
- **feature** branches are **private**
Source Code Management: **Git Branches**

- Two special and **public** branches: **master** and **development**
  - **master**: stable branch for releases, conservative interfaces
  - **development**: integration of the new features
- New feature = new **feature** branch
  - **feature** branches are **private**
- **feature** branches are **merged** in the **development** branch
  - To keep a **feature** branch **private**, never **merge** it in a **public** branch
**Source Code Management: GitHub and GitLab**

**GitLab:**
- Private instance of the AFF3CT Git repository
- Contains public and private source code
- `master`, `development` and `feature` branches
Source Code Management: GitHub and GitLab

**GitLab:**
- Private instance of the AFF3CT Git repository
- Contains public and private source code
- **master**, **development** and **feature** branches

**GitHub:**
- Public instance of the AFF3CT Git repository
- Contains only public source code
- Only **master** and **development** branches
Source Code Management: GitHub and GitLab

GitLab:
- Private instance of the AFF3CT Git repository
- Contains public and private source code
- master, development and feature branches

GitHub:
- Public instance of the AFF3CT Git repository
- Contains only public source code
- Only master and development branches

GitHub and GitLab are two instances of a Git server
- Graphical user interface (UI) for Git
Source Code Management: Add New Feature

- Create branches from `master` or `development`

```bash
$ git checkout development
$ git checkout -b new_feature_branch

Make some work and commit
$ git add touched_files
$ git commit -m "Nice message."

Update the `master` or `development` branch
$ git checkout development
$ git pull origin development

Merge `development` in the `feature` branch
$ git checkout new_feature_branch
$ git merge development
$ git push origin new_feature_branch

Merge feature in the `development` branch
$ git checkout development
$ git merge new_feature_branch
$ git push origin development
```
Source Code Management: Add New Feature

- Create branches from **master** or **development**
  
  
  $ git checkout development

  develop

  $ git checkout -b new_feature_branch

  Make some work and commit
  
  $ git add touched_files
  $ git commit -m "Nice message."

  Update the **master** or **development** branch
  
  $ git checkout development
  $ git pull origin development

  Merge **development** in the **feature** branch
  
  $ git checkout new_feature_branch
  $ git merge development
  $ git push origin new_feature_branch

  Merge **feature** in the **development** branch
  
  $ git checkout development
  $ git merge new_feature_branch
  $ git push origin development
Create branches from `master` or `development`

```sh
$ git checkout development
$ git checkout -b new_feature_branch
```

Make some work and commit

```sh
$ git add touched_files
$ git commit -m "Nice message."
```

Update the `master` or `development` branch

```sh
$ git checkout development
$ git pull origin development
```

Merge `development` in the feature branch

```sh
$ git checkout new_feature_branch
$ git merge development
$ git push origin new_feature_branch
```

Merge `feature` in the `development` branch

```sh
$ git checkout development
$ git merge new_feature_branch
$ git push origin development
```
Source Code Management: Add New Feature

- Create branches from **master** or **development**
  ```bash
  $ git checkout development
  $ git checkout -b new_feature_branch
  ```

- Make some work and commit
  ```bash
  $ git add touched_files
  ```
Source Code Management: Add New Feature

- Create branches from `master` or `development`
  
  ```
  $ git checkout development
  $ git checkout -b new_feature_branch
  ```

- Make some work and commit
  
  ```
  $ git add touched_files
  $ git commit -m "Nice message."
  ```

- Update the `master` or `development` branch
  
  ```
  $ git checkout development
  $ git pull origin development
  ```

- Merge `development` in the `feature` branch
  
  ```
  $ git checkout new_feature_branch
  $ git merge development
  $ git push origin new_feature_branch
  ```

- Merge `feature` in the `development` branch
  
  ```
  $ git checkout development
  $ git merge new_feature_branch
  $ git push origin development
  ```
Source Code Management: Add New Feature

- Create branches from master or development
  
  
  ```
  $ git checkout development
  $ git checkout -b new_feature_branch
  ```

- Make some work and commit
  
  ```
  $ git add touched_files
  $ git commit -m "Nice message."
  ```

- Update the master or development branch
  
  ```
  $ git checkout development
  ```
Source Code Management: Add New Feature

- Create branches from `master` or `development`
  
  ```bash
  $ git checkout development
  $ git checkout -b new_feature_branch
  ```

- Make some work and commit
  
  ```bash
  $ git add touched_files
  $ git commit -m "Nice message."
  ```

- Update the `master` or `development` branch
  
  ```bash
  $ git checkout development
  $ git pull origin development
  ```
Source Code Management: **Add New Feature**

- Create branches from **master** or **development**
  
  ```
  $ git checkout development
  $ git checkout -b new_feature_branch
  ```

- Make some work and commit
  
  ```
  $ git add touched_files
  $ git commit -m "Nice message."
  ```

- Update the **master** or **development** branch
  
  ```
  $ git checkout development
  $ git pull origin development
  ```
Source Code Management: **Add New Feature**

- Create branches from *master* or *development*
  
  ```
  $ git checkout development
  $ git checkout -b new_feature_branch
  ```

- Make some work and commit
  
  ```
  $ git add touched_files
  $ git commit -m "Nice message."
  ```

- Update the *master* or *development* branch
  
  ```
  $ git checkout development
  $ git pull origin development
  ```

- Merge *development* in the *feature* branch
  
  ```
  $ git checkout new_feature_branch
  $ git merge development
  $ git push origin new_feature_branch
  ```

- Merge *feature* in the *development* branch
  
  ```
  $ git checkout development
  $ git merge new_feature_branch
  $ git push origin development
  ```
Source Code Management: Add New Feature

- Create branches from `master` or `development`
  
  ```
  $ git checkout development
  $ git checkout -b new_feature_branch
  ```

- Make some work and commit
  
  ```
  $ git add touched_files
  $ git commit -m "Nice message."
  ```

- Update the `master` or `development` branch
  
  ```
  $ git checkout development
  $ git pull origin development
  ```

- Merge `development` in the `feature` branch
  
  ```
  $ git checkout new_feature_branch
  $ git merge development
  $ git push origin new_feature_branch
  ```
Source Code Management: **Add New Feature**

- Create branches from *master* or *development*
  
  $ git checkout development
  $ git checkout -b new_feature_branch

- Make some work and commit
  
  $ git add touched_files
  $ git commit -m "Nice message."

- Update the *master* or *development* branch
  
  $ git checkout development
  $ git pull origin development

- Merge *development* in the *feature* branch
  
  $ git checkout new_feature_branch
  $ git merge development
  $ git push origin new_feature_branch

---

A. Cassagne, O. Hartmann, M. Léonardon, C. Leroux, C. Jégo

AFF3CT Seminary
Source Code Management: Add New Feature

- Create branches from `master` or `development`
  
  - $ git checkout development
  - $ git checkout -b new_feature_branch

- Make some work and commit
  
  - $ git add touched_files
  - $ git commit -m "Nice message."

- Update the `master` or `development` branch
  
  - $ git checkout development
  - $ git pull origin development

- Merge `development` in the `feature` branch
  
  - $ git checkout new_feature_branch
  - $ git merge development
  - $ git push origin new_feature_branch

- Merge `feature` in the `development` branch
  
  - $ git checkout development
**Source Code Management: Add New Feature**

- Create branches from `master` or `development`
  
  ```
  $ git checkout development
  $ git checkout -b new_feature_branch
  ```

- Make some work and commit
  
  ```
  $ git add touched_files
  $ git commit -m "Nice message."
  ```

- Update the `master` or `development` branch
  
  ```
  $ git checkout development
  $ git pull origin development
  ```

- Merge `development` in the `feature` branch
  
  ```
  $ git checkout new_feature_branch
  $ git merge development
  $ git push origin new_feature_branch
  ```

- Merge `feature` in the `development` branch
  
  ```
  $ git checkout development
  $ git merge new_feature_branch
  $ git push origin development
  ```
Create branches from `master` or `development`

- $ git checkout development
- $ git checkout -b new_feature_branch

Make some work and commit

- $ git add touched_files
- $ git commit -m "Nice message."

Update the `master` or `development` branch

- $ git checkout development
- $ git pull origin development

Merge `development` in the `feature` branch

- $ git checkout new_feature_branch
- $ git merge development
- $ git push origin new_feature_branch

Merge `feature` in the `development` branch

- $ git checkout development
- $ git merge new_feature_branch
- $ git push origin development
Source Code Management: Add New Feature

- Create branches from `master` or `development`
  
  ```bash
  $ git checkout development
  $ git checkout -b new_feature_branch
  ```

- Make some work and commit
  
  ```bash
  $ git add touched_files
  $ git commit -m "Nice message."
  ```

- Update the `master` or `development` branch
  
  ```bash
  $ git checkout development
  $ git pull origin development
  ```

- Merge `development` in the `feature` branch
  
  ```bash
  $ git checkout new_feature_branch
  $ git merge development
  $ git push origin new_feature_branch
  ```

- Merge `feature` in the `development` branch

Make a merge request from GitLab!
Repositories

AFF3CT is structured in many Git repositories.

- repository: a specific Git repository

configuration_files: a set of AFF3CT input configuration files (like H-matrices)

error_rate_references: a database of simulated results with AFF3CT PyBER

my_project_with_aff3ct: a database of examples to use AFF3CT as a Toolbox/Library

MIPP: a SIMD library used in AFF3CT source code
Repositories

AFF3CT is structured in many Git repositories.

- repository: a specific Git repository

  - aff3ct: the main AFF3CT repository (source code)
AFF3CT is structured in many Git repositories.

- **repository**: a specific Git repository

- **aff3ct**: the main AFF3CT repository (source code)
- **configuration_files**: a set of AFF3CT input configuration files (like $H$ matrices)
Repositories

AFF3CT is structured in many Git repositories.

- **repository**: a specific Git repository

- **aff3ct**: the main AFF3CT repository (source code)
- **configuration_files**: a set of AFF3CT input configuration files (like $H$ matrices)
- **error_rate_references**: a database of simulated results with AFF3CT
Repositories

AFF3CT is structured in many Git repositories.

- repository: a specific Git repository

- **aff3ct**: the main AFF3CT repository (source code)
- **configuration_files**: a set of AFF3CT input configuration files (like $H$ matrices)
- **error_rate_references**: a database of simulated results with AFF3CT
- **PyBER**: a graphical application to visualize the AFF3CT simulated BER/FER results
AFF3CT is structured in many Git repositories.

- repository: a specific Git repository

- **aff3ct**: the main AFF3CT repository (source code)
- **configuration_files**: a set of AFF3CT input configuration files (like $H$ matrices)
- **error_rate_references**: a database of simulated results with AFF3CT
- **PyBER**: a graphical application to visualize the AFF3CT simulated BER/FER results
- **my_project_with_aff3ct**: a database of examples to use AFF3CT as a Toolbox/Library
Repositories

AFF3CT is structured in many Git repositories.

- **repository**: a specific Git repository

- **aff3ct**: the main AFF3CT repository (source code)
- **configuration_files**: a set of AFF3CT input configuration files (like $H$ matrices)
- **error_rate_references**: a database of simulated results with AFF3CT
- **PyBER**: a graphical application to visualize the AFF3CT simulated BER/FER results
- **my_project_with_aff3ct**: a database of examples to use AFF3CT as a Toolbox/Library
- **MIPP**: a SIMD library used in AFF3CT source code
Repositories

**AFF3CT** is structured in many Git repositories.

- **repository**: a specific Git repository
- **group**: set of repositories

- **aff3ct/aff3ct**: the main **AFF3CT** repository (source code)
- **aff3ct/configuration_files**: a set of **AFF3CT** input configuration files (like $H$ matrices)
- **aff3ct/error_rate_references**: a database of simulated results with **AFF3CT**
- **aff3ct/PyBER**: a graphical application to visualize the **AFF3CT** simulated BER/FER results
- **aff3ct/my_project_with_aff3ct**: a database of examples to use **AFF3CT** as a Toolbox/Library
- **aff3ct/MIPP**: a SIMD library used in **AFF3CT** source code
Submodules

There are some dependencies between the repositories of the AFF3CT group (called git submodule).
Submodules

There are some dependencies between the repositories of the AFF3CT group (called git submodule).

- **aff3ct**
  - configuration_files
  - error_rate_references
  - MIPP
Submodules

There are some dependencies between the repositories of the AFF3CT group (called git submodule).

- **aff3ct**
  - configuration_files
  - error_rate_references
  - MIPP

- **PyBER**
  - error_rate_references
Submodules

There are some dependencies between the repositories of the AFF3CT group (called `git submodule`).

- **aff3ct**
  - configuration_files
  - error_rate_references
  - MIPP

- **PyBER**
  - error_rate_references

- **my_project_with_aff3ct**
  - aff3ct
    - configuration_files
    - error_rate_references
    - MIPP
There are some dependencies between the repositories of the AFF3CT group (called git submodule).

- **aff3ct/aff3ct**
  - aff3ct/configuration_files
  - aff3ct/error_rate_references
  - aff3ct/MIPP

- **aff3ct/PyBER**
  - aff3ct/error_rate_references

- **aff3ct/my_project_with_aff3ct**
  - aff3ct/aff3ct
    - aff3ct/configuration_files
    - aff3ct/error_rate_references
    - aff3ct/MIPP
Problem

A source code can easily be broken, side effects are very common in software development.
Continuous Integration

Problem
A source code can easily be broken, side effects are very common in software development.

Proposal
Since our results are reproducible, we propose to test the code after each modification.

Tests are automated: after each git push on GitLab
The developer has the responsibility to add the tests covering its features.
Continuous Integration

Problem
A source code can easily be broken, side effects are very common in software development.

Proposal
Since our results are reproducible, we propose to test the code after each modifications.

- **Tests are automated**: after each `git push` on GitLab
- The developer has **the responsibility** to add the tests covering its features
The tests are run for each branches (even on the feature branches)
Continuous Integration: Focus on a Specific Pipeline

One series of tests is decomposed in a 3-stage pipeline
Continuous Integration: Focus on a Specific Pipeline

One series of tests is decomposed in a 3-stage pipeline

- **Analysis**: checks if the code syntax is valid (with cppcheck)
Continuous Integration: Focus on a Specific Pipeline

One series of tests is decomposed in a **3-stage pipeline**

- **Analysis**: checks if the code syntax is valid (with `cppcheck`)
- **Build**: compiles the code on various systems and on various targets
Continuous Integration: Focus on a Specific Pipeline

One series of tests is decomposed in a 3-stage pipeline

- **Analysis**: checks if the code syntax is valid (with cppcheck)
- **Build**: compiles the code on various systems and on various targets
- **Test**: runs simulations to check the regressions
  - Validates simulations from the `error_rate_references` repository
  - Each developer is invited to add simulation results in this repository
Roadmap and Discussion

- **Wraps** AFF3CT for other languages (Python, MATLAB)
Roadmap and Discussion

- **Wraps** AFF3CT for other languages (Python, MATLAB)
- **Automatic parallelism** with `my_project_with_aff3ct`
Roadmap and Discussion

- **Wraps** AFF3CT for other languages (Python, MATLAB)
- **Automatic parallelism** with `my_project_with_aff3ct`
- **Graphical User Interface**
  - Simplify the command line utilization
  - Creation of new communication chains
Roadmap and Discussion

- **Wraps** AFF3CT for other languages (Python, MATLAB)
- **Automatic parallelism** with `my_project_with_aff3ct`
- **Graphical User Interface**
  - Simplify the command line utilization
  - Creation of new communication chains
- **And you, what do you think?**