A Fast Forward Error Correction Toolbox: Seminary

Adrien Cassagne[1,2] Olivier Hartmann[1] Mathieu Léonardon[1]
Camille Leroux[1] Christophe Jégo[1]

[1]IMS/CNRS, Bordeaux, France

[2]Inria, Bordeaux Institute of Technology, U. of Bordeaux, LaBRI/CNRS, Bordeaux, France

IMS, March 2018

Plan

A Fast Forward Error Correction Toolbox!

Why AFF3CT?

- Knowledge capitalization

A Fast Forward Error Correction Toolbox!

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

A Fast Forward Error Correction Toolbox!

Why AFF3CT?

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

A Fast Forward Error Correction Toolbox!

Why AFF3CT?

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

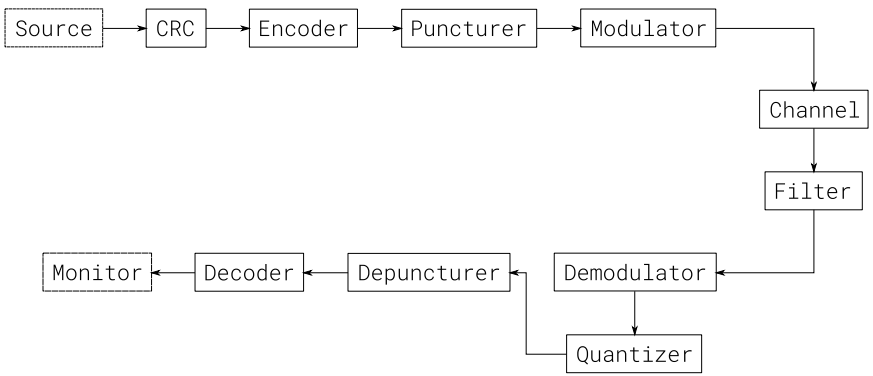
Plan

Simulator

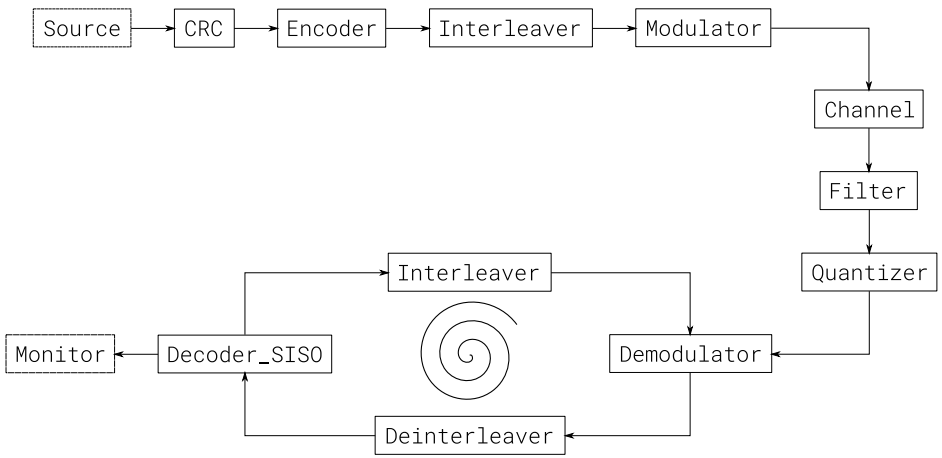
Monte-Carlo simulations

- BER/FER *standard*
- BER/FER *iterative*

Simulation_BFER



Simulation_BFERI

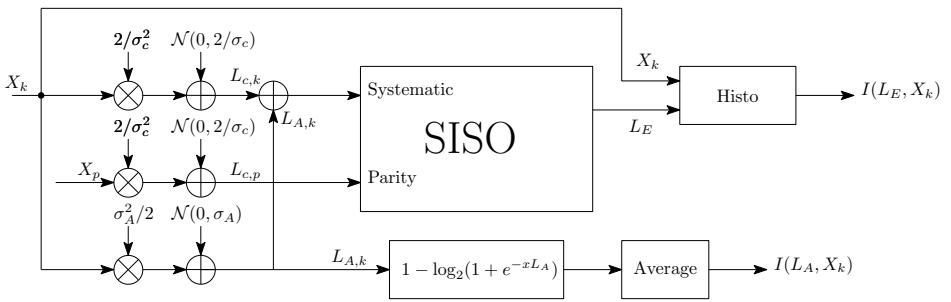


Simulator

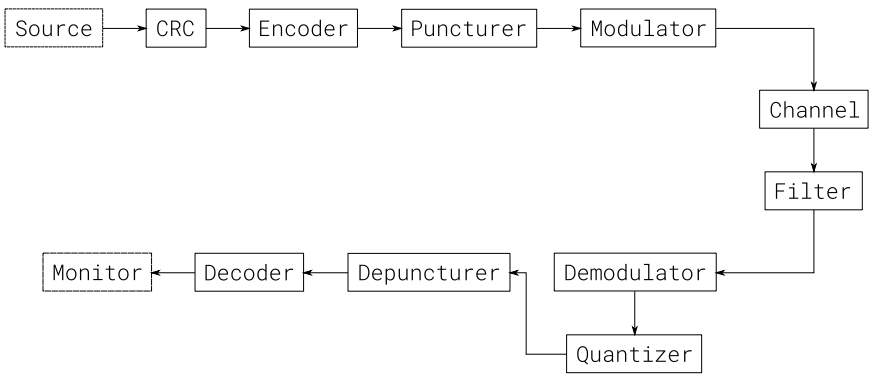
Monte-Carlo simulations

- BER/FER *standard*
- BER/FER *iterative*
- Parallel EXIT charts

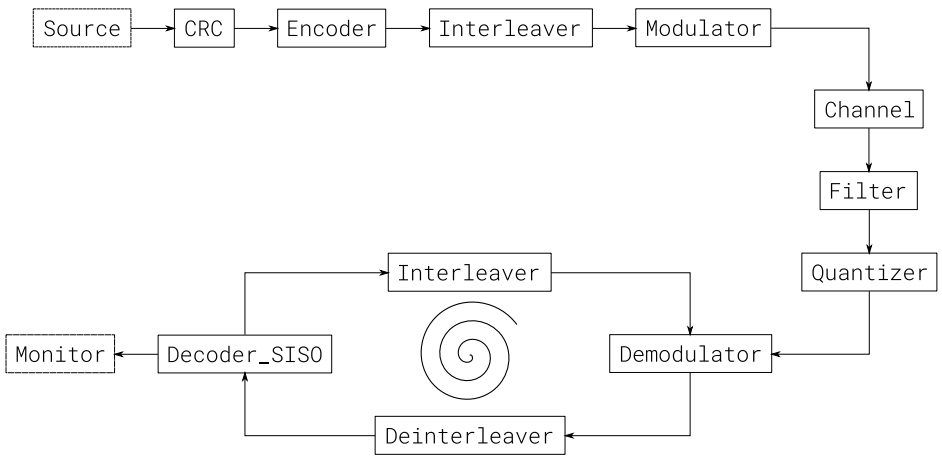
Parallel EXIT



Simulation_BFER



Simulation_BFERI



Simulator: Codecs

List of Supported Channel Codes

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

Simulator: Codecs

List of Supported Channel Codes

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

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

Simulator: Modems

List of Supported Modulations/Demodulations

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

Simulator: Channels

List of Supported Channels

Channel	Mono-user	Multi-user	Characteristics
AWGN	Yes	Yes	Additive White Gaussian Noise
Rayleigh	Yes	Yes	Flat Rayleigh fading channel
User defined	Yes	Not yet	User can import noise samples from an external file

Simulator: Channels

List of Supported Channels

Channel	Mono-user	Multi-user	Characteristics
AWGN	Yes	Yes	Additive White Gaussian Noise
Rayleigh	Yes	Yes	Flat Rayleigh fading channel
User defined	Yes	Not yet	User can import noise samples from an external file

- 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**)

Simulator: Performance

`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`)

Simulator: Performance

`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`)

Toolbox / Library

Limitation

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

Toolbox / Library

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.

Toolbox / Library

Limitation

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

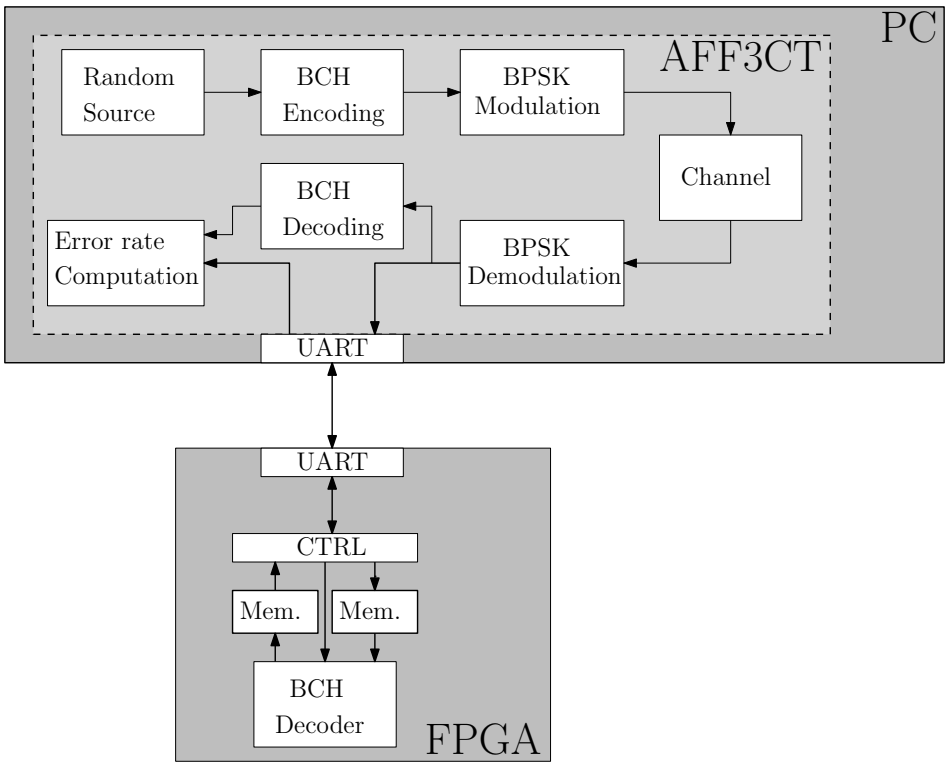
Solution

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

- 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

Prototyping / Hardware in the Loop

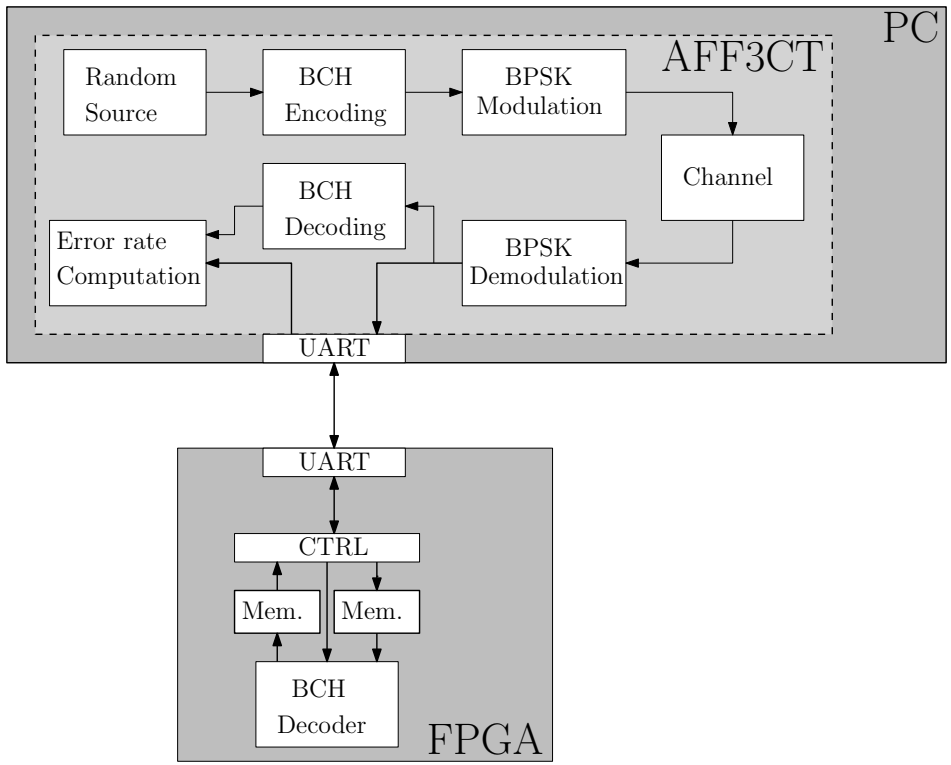
AFF3CT has been designed to **facilitate prototyping on FPGA**



Prototyping / Hardware in the Loop

AFF3CT has been designed to **facilitate prototyping on FPGA**

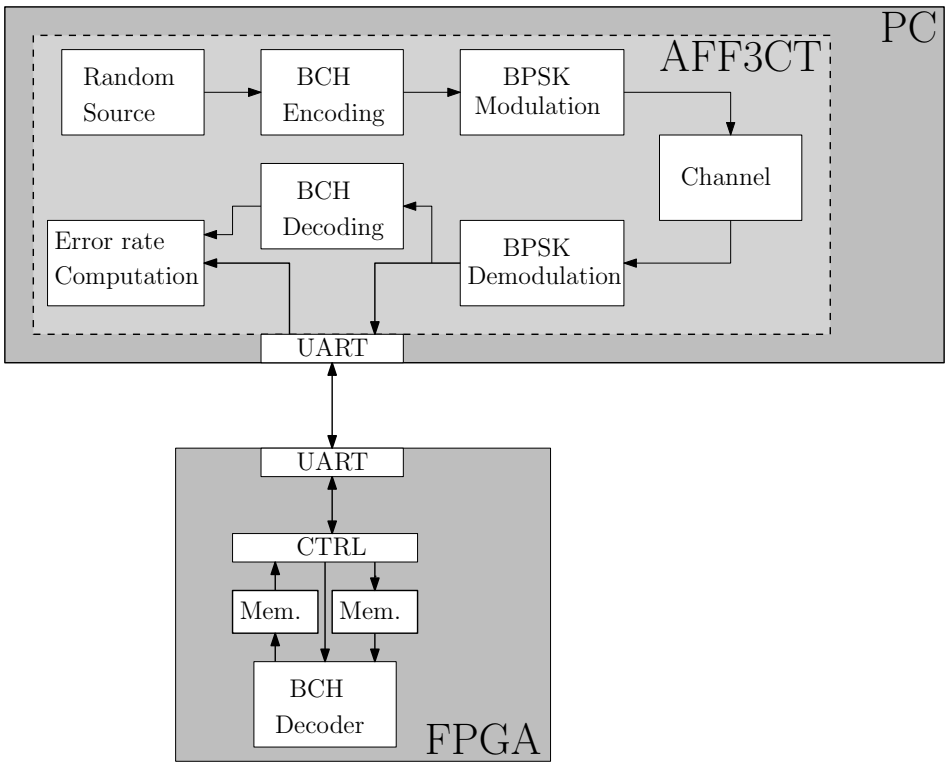
1. Dedicated interfaces and implementations for UART and Ethernet



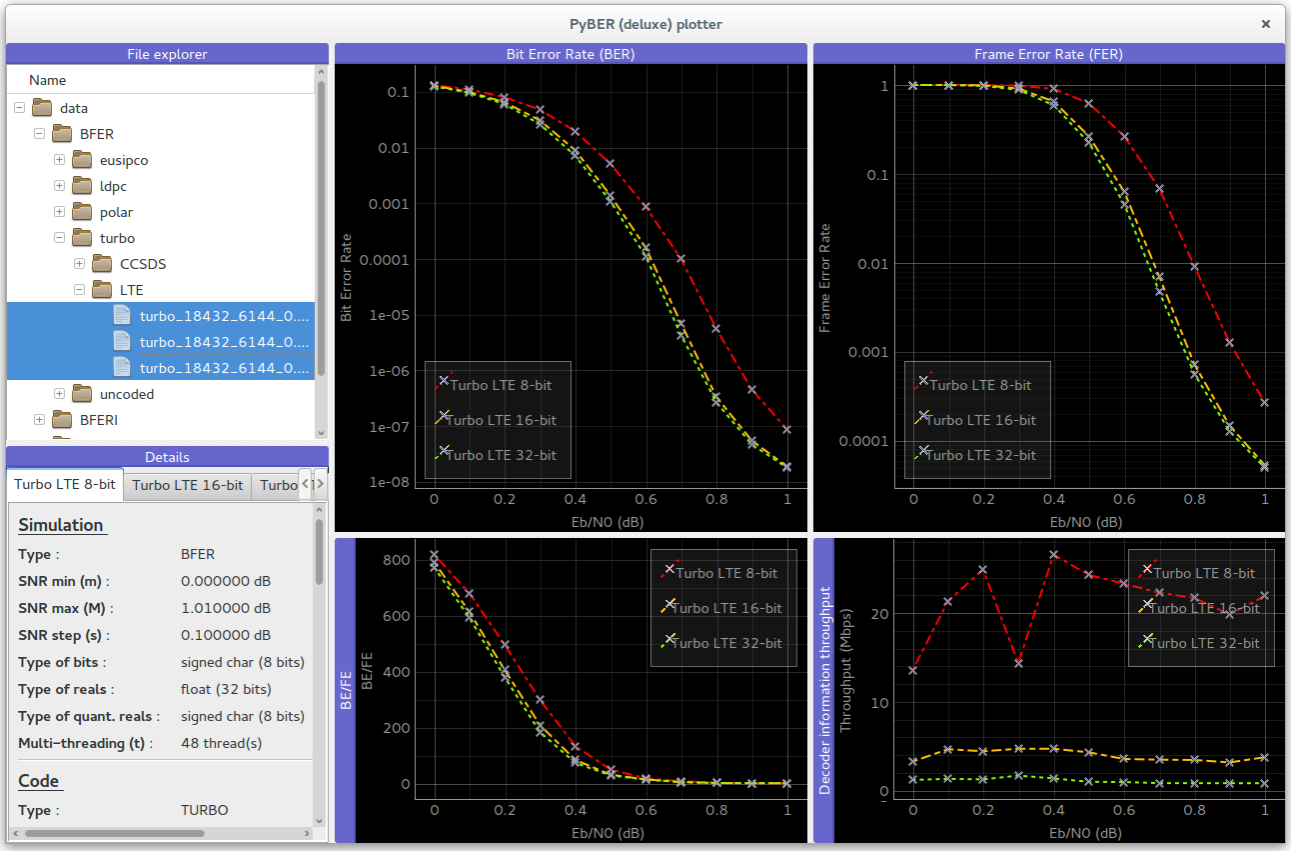
Prototyping / Hardware in the Loop

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

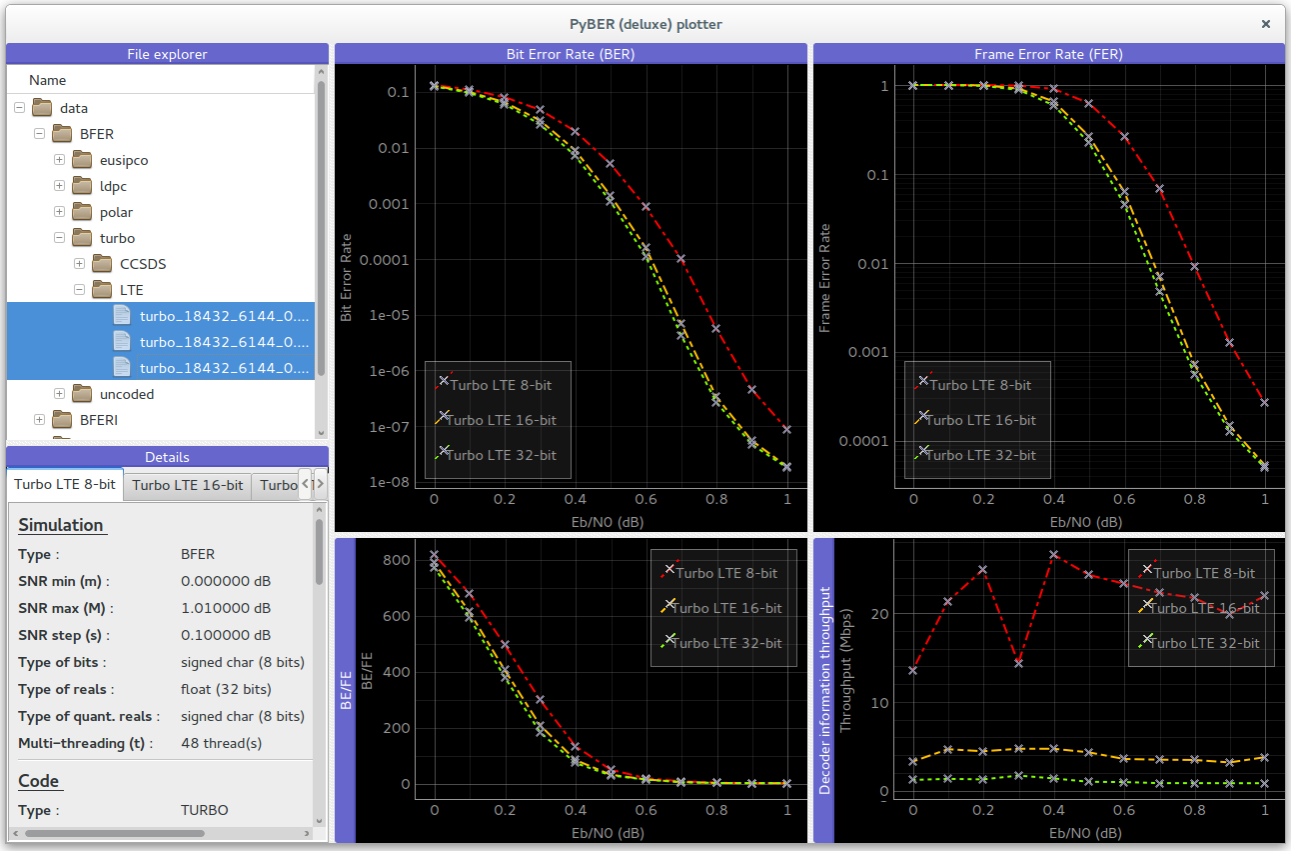


PyBER is **a generic visualization tool** wrote in Python



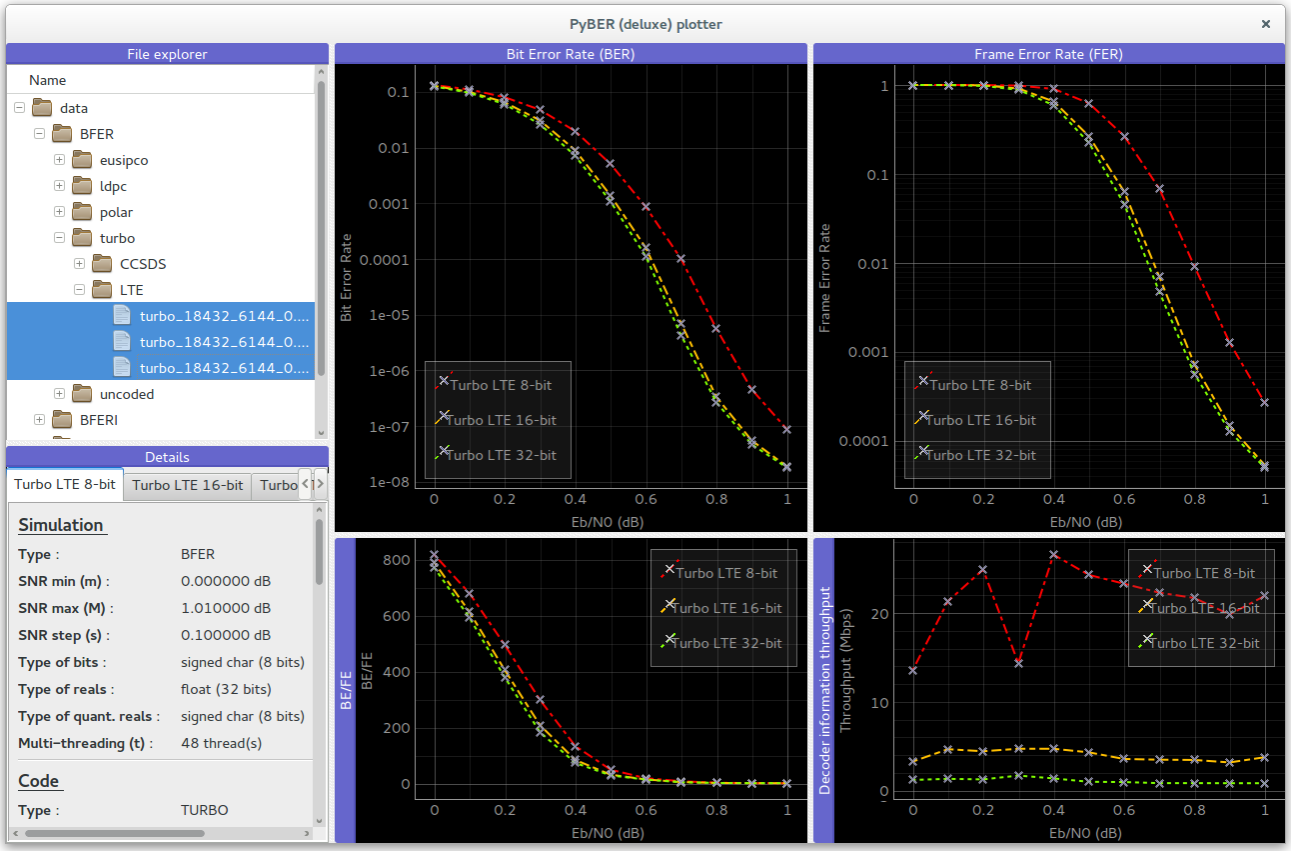
PyBER is **a generic visualization tool** wrote in Python

- **Plots BER/FER performance** curves in live



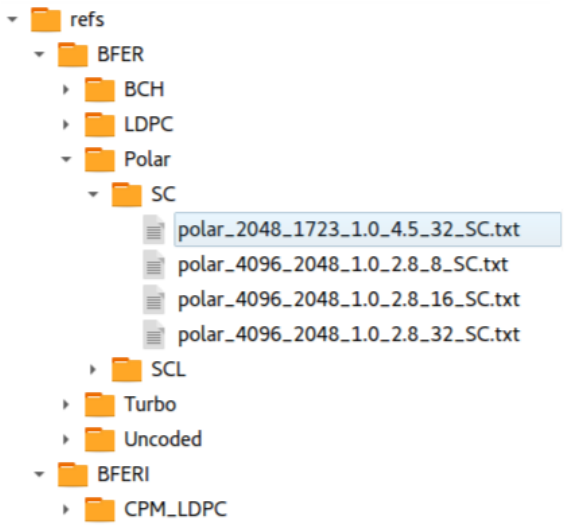
PyBER is **a generic visualization tool** wrote in Python

- **Plots BER/FER performance** curves in live
- **Compares** various simulation performances



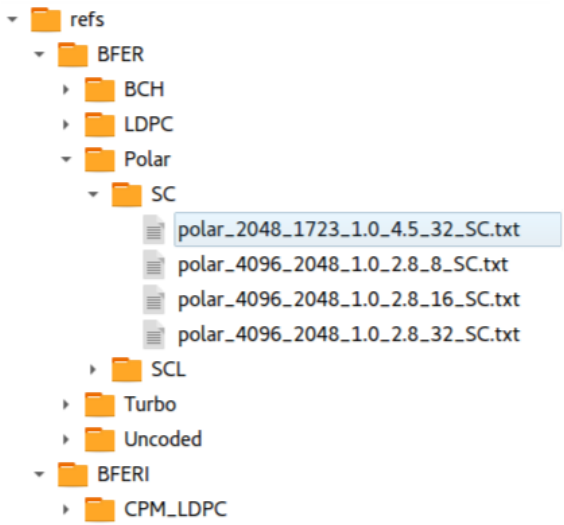
References

- The **AFF3CT** project includes a database of simulation outputs



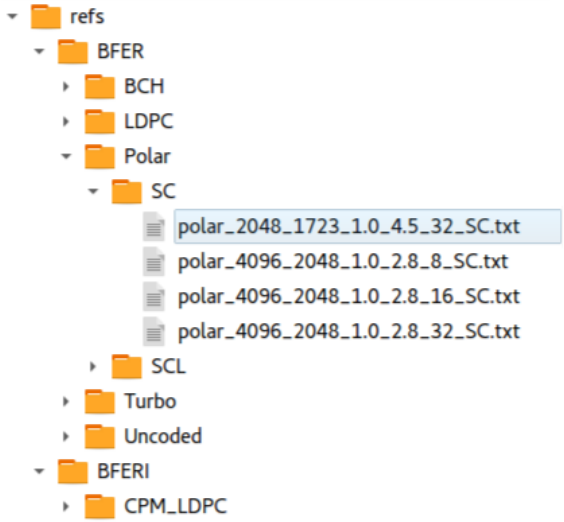
References

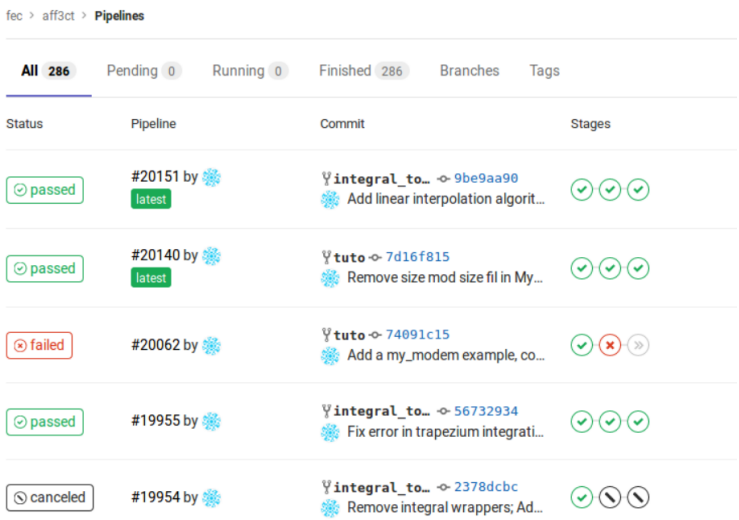
- The **AFF3CT** project includes a database of simulation outputs
- Running commands are included to reproduce each simulation



References

- 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

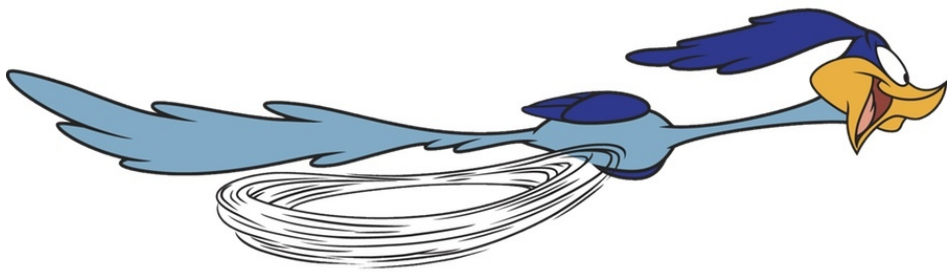
- SIMD C++ **wrapper**
  - Maximizes code portability
  - **Improves expressiveness** over intrinsics
  - **No dependencies** to other libraries

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

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



MIPP MIPP!

Miscellaneous: Fixed Point API

Management of fixed point numbers in C++11

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

Miscellaneous: Fixed Point API

Management of fixed point numbers in C++11

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

- 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

Miscellaneous: Fixed Point API

Management of fixed point numbers in C++11

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

- 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 (|, &, ^, ~)

Miscellaneous: Fixed Point API

Management of fixed point numbers in C++11

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

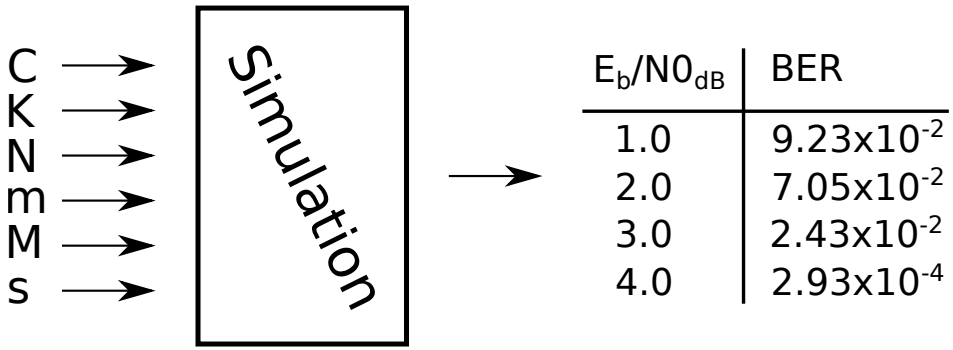
- Defines a specific type for fixed point numbers: `fixed<S,V>`
  - `S`: the total number of bits
  - `V`: the number of bits for the decimal part
- Supports **arithmetic** operations (+, -, *, /, %)
- Supports **logic** operations (|, &, ^, ~)
- Requires to know the quantification (`S` and `V`) at compile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

Plan

What is a Simulation?

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

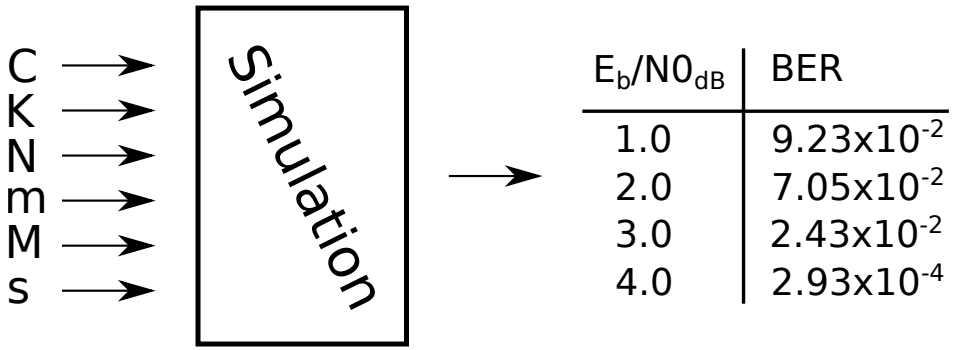
```
$ aff3ct -C POLAR -K 1723 -N 2048 -m 1.0 -M 4.0 -s 1.0
```



What is a Simulation?

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

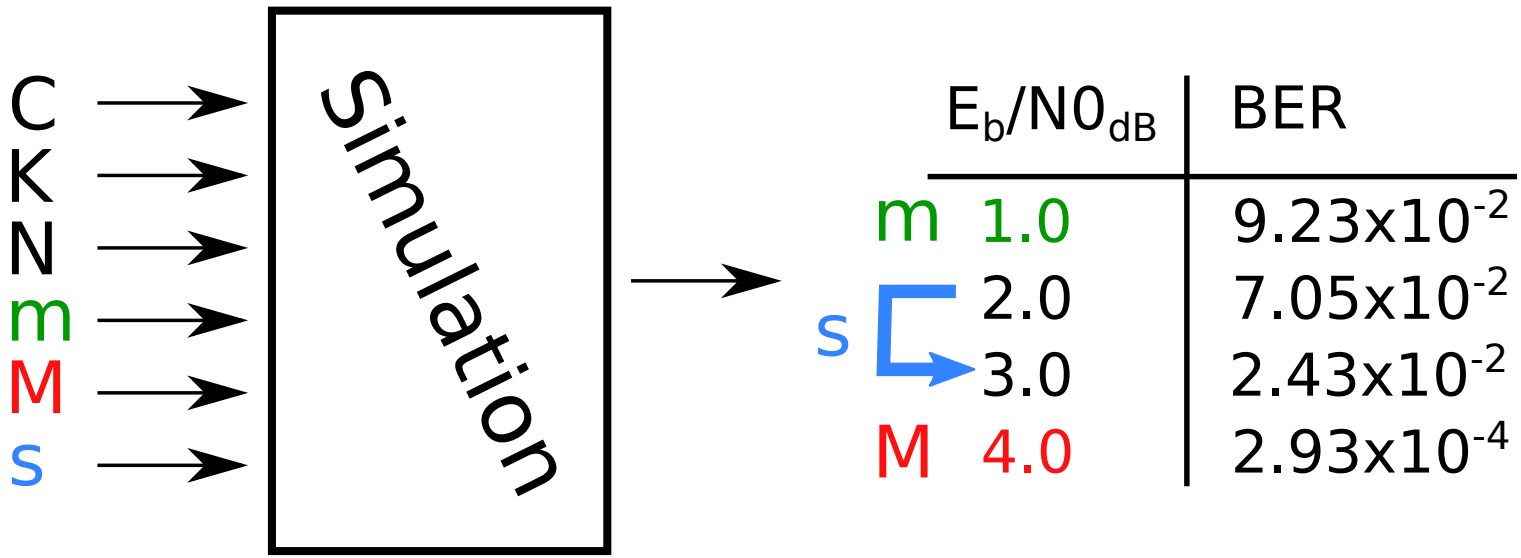
```
$ aff3ct -C POLAR -K 1723 -N 2048 -m 1.0 -M 4.0 -s 1.0
```




```
# ----------------------------------------------------------------------------|
#       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 |
```

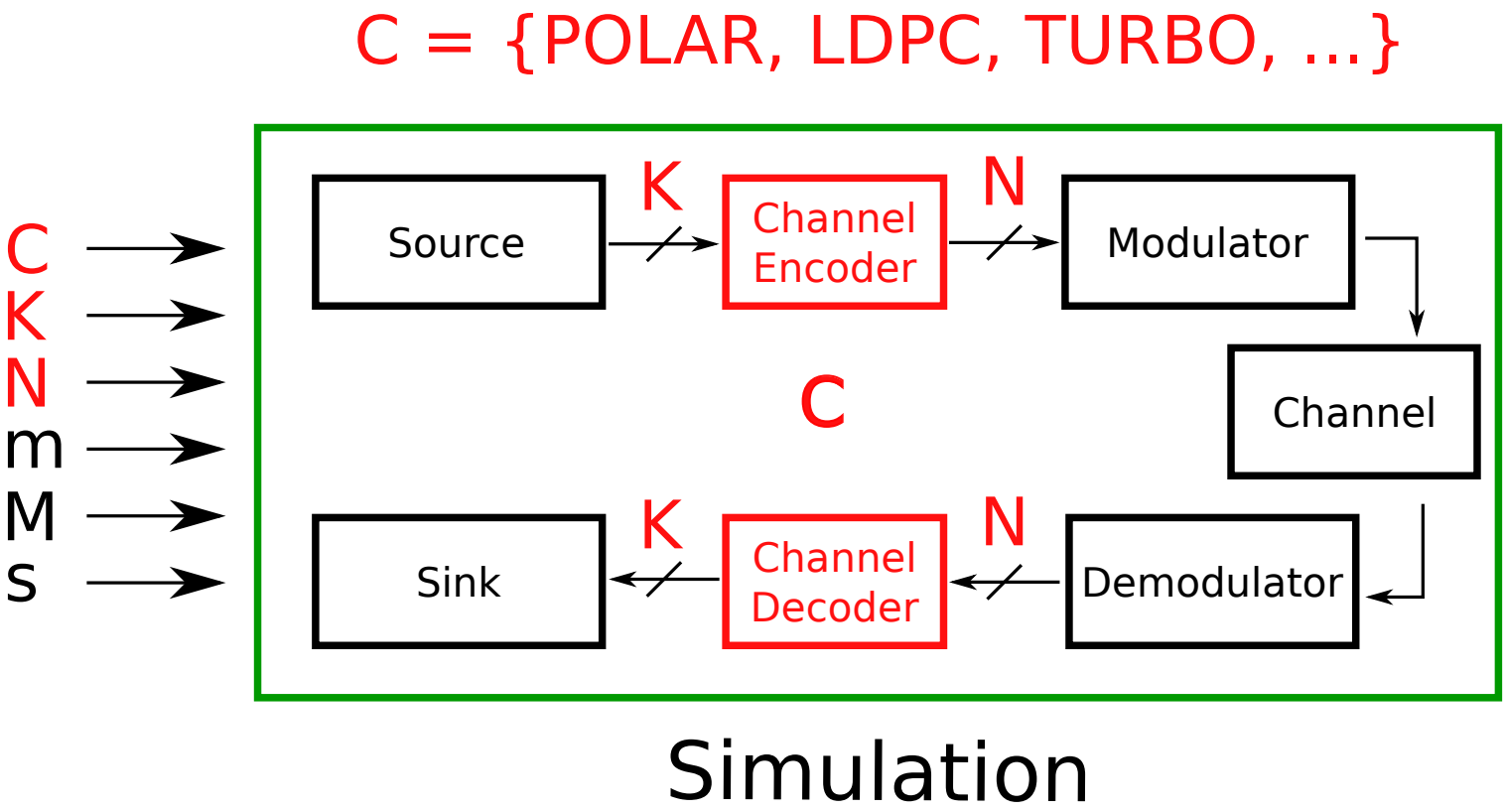
E_b/N_0 Arguments

The SNR related inputs...

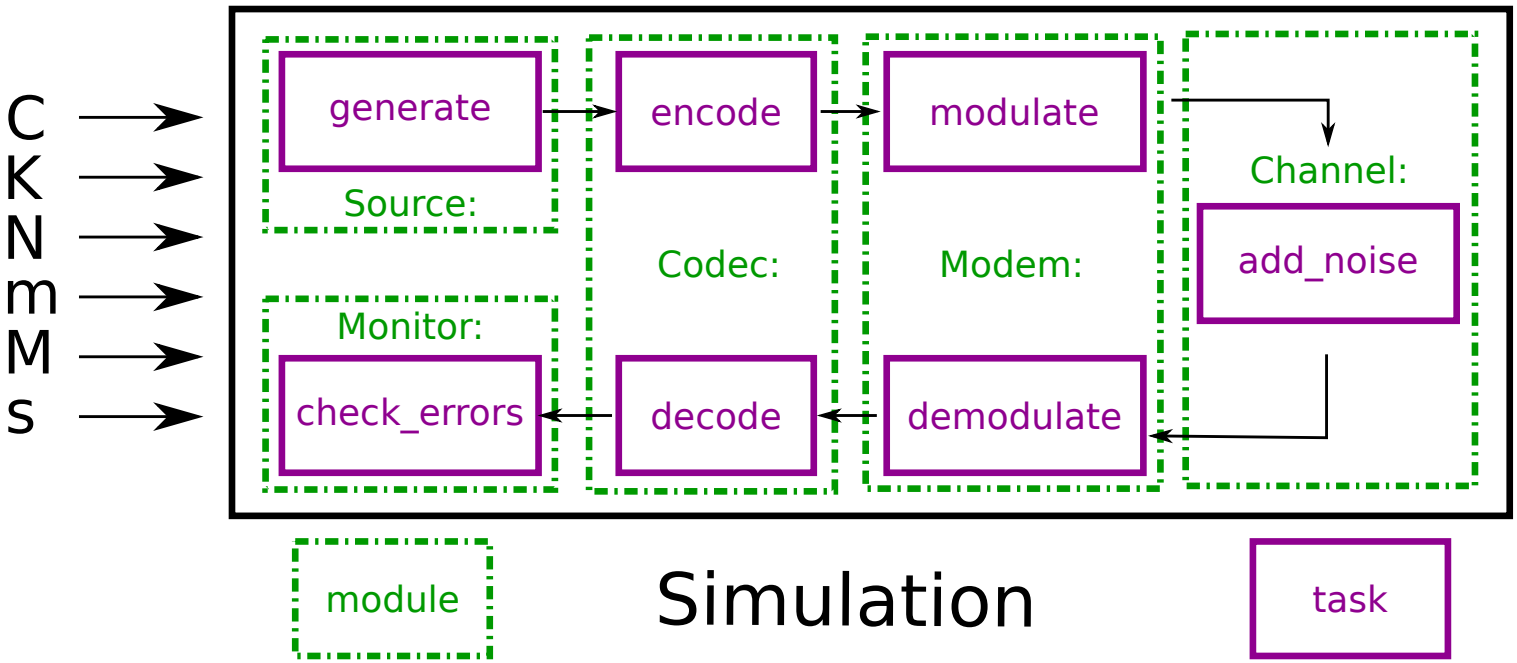


Channel Code Arguments

... and the code related inputs.



Modules and Tasks



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 formated 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 formated 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.
```

Tasks and Modules Options

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.

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

Tasks and Modules Options

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.

```
$ 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.

```
$ 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:

```
# ------------------------------
# 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,     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,     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.

```
#
# ----------------------------------------||---------------------------||-------------------------------||-------------------------------
#       Statistics for the given task     ||      Basic statistics     ||   Measured throughput         ||   Measured latency
#    ('*' = any, '-' = same as previous)  ||        on the task        ||   considering the last socket  ||   considering the last socket
# ----------------------------------------||---------------------------||-------------------------------||-------------------------------
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#      MODULE |             TASK |   TIMER ||    CALLS |     TIME |   PERC ||  AVERAGE |  MINIMUM |  MAXIMUM ||  AVERAGE |  MINIMUM |  MAXIMUM
#             |                  |         ||          |      (s) |    (%) ||  (Mb/s)  |  (Mb/s)  |  (Mb/s)  ||    (us)  |    (us)  |    (us)
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#     Channel |        add_noise |       * ||     5533 |     0.48 |  38.70 ||    23.80 |     0.31 |    25.93 ||    86.05 |    79.00 |  6613.54
#      Source |         generate |       * ||     5533 |     0.35 |  28.37 ||    27.31 |     5.44 |    37.58 ||    63.08 |    45.85 |   316.53
#     Encoder |           encode |       * ||     5533 |     0.27 |  21.67 ||    42.51 |     6.55 |    57.17 ||    48.18 |    35.82 |   312.65
#     Decoder |      decode_siho |       * ||     5533 |     0.13 |  10.51 ||    73.74 |     0.35 |    91.15 ||    23.36 |    18.90 |  4973.41
#     Monitor |     check_errors |       * ||     5533 |     0.00 |   0.35 ||  2212.37 |   312.36 |  3065.84 ||     0.78 |     0.56 |     5.52
#       Modem |         modulate |       * ||     5533 |     0.00 |   0.21 ||  4287.62 |   146.28 |  5902.02 ||     0.48 |     0.35 |    14.00
#       Modem |       demodulate |       * ||     5533 |     0.00 |   0.19 ||  4742.44 |   308.06 |  6649.35 ||     0.43 |     0.31 |     6.65
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#       TOTAL |                * |       * ||     5533 |     1.23 | 100.00 ||     7.75 |     0.14 |     9.53 ||   222.37 |   180.79 | 12242.29
#
```

Statistics Argument

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

```
#
# ----------------------------------------||---------------------------||-------------------------------||-----------------------------
#       Statistics for the given task     ||      Basic statistics     ||    Measured throughput        ||    Measured latency
#    ('*' = any, '-' = same as previous)  ||         on the task       ||  considering the last socket  ||  considering the last socket
# ----------------------------------------||---------------------------||-------------------------------||-----------------------------
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#      MODULE |             TASK |   TIMER ||    CALLS |     TIME |   PERC ||  AVERAGE |  MINIMUM |  MAXIMUM ||  AVERAGE |  MINIMUM |  MAXIMUM
#             |                  |         ||          |      (s) |    (%) ||  (Mb/s)  |  (Mb/s)  |  (Mb/s)  ||    (us)  |    (us)  |    (us)
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#     Channel |        add_noise |       * ||     5533 |     0.48 |  38.70 ||    23.80 |     0.31 |    25.93 ||    86.05 |    79.00 |  6613.54
#      Source |         generate |       * ||     5533 |     0.35 |  28.37 ||    27.31 |     5.44 |    37.58 ||    63.08 |    45.85 |   316.53
#     Encoder |           encode |       * ||     5533 |     0.27 |  21.67 ||    42.51 |     6.55 |    57.17 ||    48.18 |    35.82 |   312.65
#     Decoder |      decode_siho |       * ||     5533 |     0.13 |  10.51 ||    73.74 |     0.35 |    91.15 ||    23.36 |    18.90 |  4973.41
#     Monitor |     check_errors |       * ||     5533 |     0.00 |   0.35 ||  2212.37 |   312.36 |  3065.84 ||     0.78 |     0.56 |     5.52
#       Modem |         modulate |       * ||     5533 |     0.00 |   0.21 ||  4287.62 |   146.28 |  5902.02 ||     0.48 |     0.35 |    14.00
#       Modem |       demodulate |       * ||     5533 |     0.00 |   0.19 ||  4742.44 |   308.06 |  6649.35 ||     0.43 |     0.31 |     6.65
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#       TOTAL |                * |       * ||     5533 |     1.23 | 100.00 ||     7.75 |     0.14 |     9.53 ||   222.37 |   180.79 | 12242.29
#
```

- The measured latency of each **task** includes the time needed to read the input socket and write the output socket (**single threaded**)

Statistics Argument

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

```
#
# ----------------------------------------||---------------------------||-------------------------------||-----------------------------
#       Statistics for the given task     ||      Basic statistics     ||   Measured throughput         ||   Measured latency
#    ('*' = any, '-' = same as previous)  ||        on the task        ||  considering the last socket  ||  considering the last socket
# ----------------------------------------||---------------------------||-------------------------------||-----------------------------
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#      MODULE |             TASK |   TIMER ||    CALLS |     TIME |   PERC ||  AVERAGE |  MINIMUM |  MAXIMUM ||  AVERAGE |  MINIMUM |  MAXIMUM
#             |                  |         ||          |      (s) |    (%) ||  (Mb/s)  |  (Mb/s)  |  (Mb/s)  ||    (us)  |    (us)  |    (us)
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#     Channel |        add_noise |       * ||     5533 |     0.48 |  38.70 ||    23.80 |     0.31 |    25.93 ||    86.05 |    79.00 |  6613.54
#      Source |         generate |       * ||     5533 |     0.35 |  28.37 ||    27.31 |     5.44 |    37.58 ||    63.08 |    45.85 |   316.53
#     Encoder |           encode |       * ||     5533 |     0.27 |  21.67 ||    42.51 |     6.55 |    57.17 ||    48.18 |    35.82 |   312.65
#     Decoder |      decode_siho |       * ||     5533 |     0.13 |  10.51 ||    73.74 |     0.35 |    91.15 ||    23.36 |    18.90 |  4973.41
#     Monitor |     check_errors |       * ||     5533 |     0.00 |   0.35 ||  2212.37 |   312.36 |  3065.84 ||     0.78 |     0.56 |     5.52
#       Modem |         modulate |       * ||     5533 |     0.00 |   0.21 ||  4287.62 |   146.28 |  5902.02 ||     0.48 |     0.35 |    14.00
#       Modem |       demodulate |       * ||     5533 |     0.00 |   0.19 ||  4742.44 |   308.06 |  6649.35 ||     0.43 |     0.31 |     6.65
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#       TOTAL |                * |       * ||     5533 |     1.23 | 100.00 ||     7.75 |     0.14 |     9.53 ||   222.37 |   180.79 | 12242.29
#
```

- 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

Statistics Argument

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

```
#
# ----------------------------------------||---------------------------||-------------------------------||-----------------------------
#       Statistics for the given task     ||      Basic statistics     ||      Measured throughput       ||      Measured latency
#    ('*' = any, '-' = same as previous)  ||        on the task        ||   considering the last socket  ||   considering the last socket
# ----------------------------------------||---------------------------||-------------------------------||-----------------------------
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#      MODULE |             TASK |   TIMER ||    CALLS |     TIME |   PERC ||  AVERAGE |  MINIMUM |  MAXIMUM ||  AVERAGE |  MINIMUM |  MAXIMUM
#             |                  |         ||          |      (s) |    (%) ||  (Mb/s)  |  (Mb/s)  |  (Mb/s)  ||    (us)  |    (us)  |    (us)
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#     Channel |        add_noise |       * ||     5533 |     0.48 |  38.70 ||    23.80 |     0.31 |    25.93 ||    86.05 |    79.00 |  6613.54
#      Source |         generate |       * ||     5533 |     0.35 |  28.37 ||    27.31 |     5.44 |    37.58 ||    63.08 |    45.85 |   316.53
#     Encoder |           encode |       * ||     5533 |     0.27 |  21.67 ||    42.51 |     6.55 |    57.17 ||    48.18 |    35.82 |   312.65
#     Decoder |      decode_siho |       * ||     5533 |     0.13 |  10.51 ||    73.74 |     0.35 |    91.15 ||    23.36 |    18.90 |  4973.41
#     Monitor |     check_errors |       * ||     5533 |     0.00 |   0.35 ||  2212.37 |   312.36 |  3065.84 ||     0.78 |     0.56 |     5.52
#       Modem |         modulate |       * ||     5533 |     0.00 |   0.21 ||  4287.62 |   146.28 |  5902.02 ||     0.48 |     0.35 |    14.00
#       Modem |       demodulate |       * ||     5533 |     0.00 |   0.19 ||  4742.44 |   308.06 |  6649.35 ||     0.43 |     0.31 |     6.65
# ------------|------------------|---------||----------|----------|--------||----------|----------|----------||----------|----------|----------
#       TOTAL |                * |       * ||     5533 |     1.23 | 100.00 ||     7.75 |     0.14 |     9.53 ||   222.37 |   180.79 | 12242.29
#
```

- 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
- $Throughput = \frac{N_b}{latency}$

Parallelism

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× Intel® Xeon® E5-2690 v3)

```
$ aff3ct -C POLAR -m 1 -M 4 -K 1723 -N 2048 -t 48
```

Parallelism

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× Intel® Xeon® E5-2690 v3)

```
$ 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× Intel® Xeon® E5-2690 v3)

```
$ mpirun -np 64 aff3ct -C POLAR -m 1 -M 4 -K 1723 -N 2048 -t 48
```

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.

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.

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.

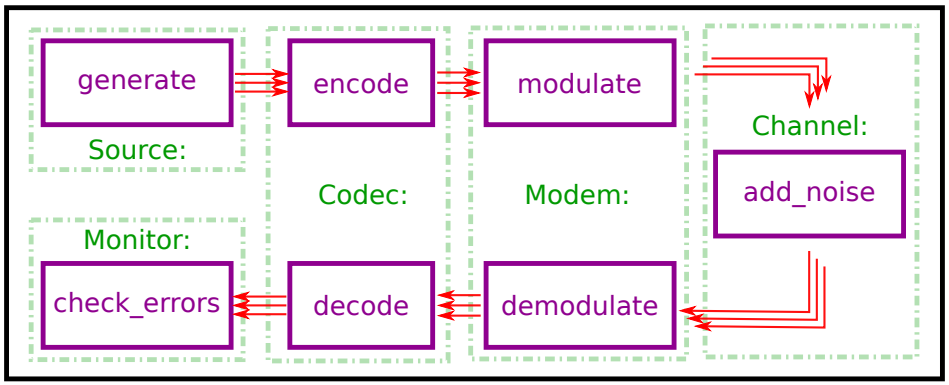
This is very useful when working on the code error floor!

Inter Frame

Some **modules** can deal with multiple frames simultaneously:

Inter Frame

Some **modules** can deal with multiple frames simultaneously:

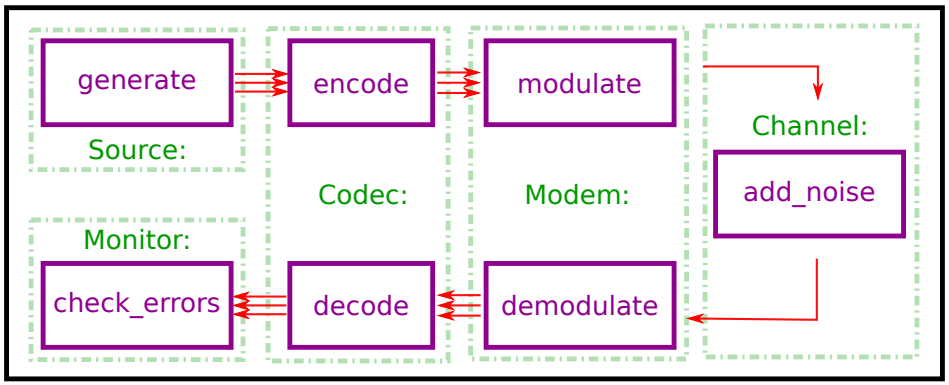


- In order to increase the parallelism with SIMD instructions (`MIPP`)

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

Inter Frame

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
```

Plan

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`, ...)

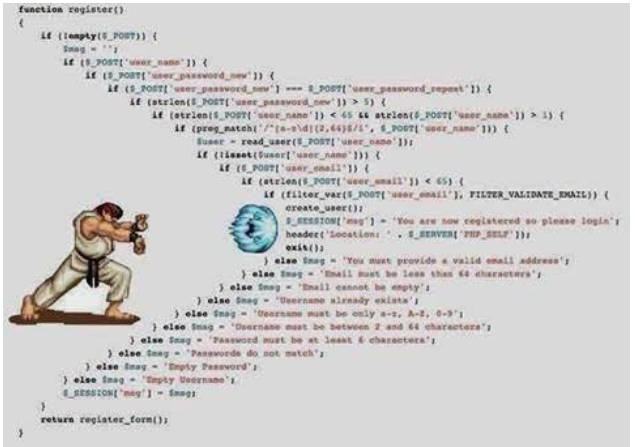
Source Code Organization

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

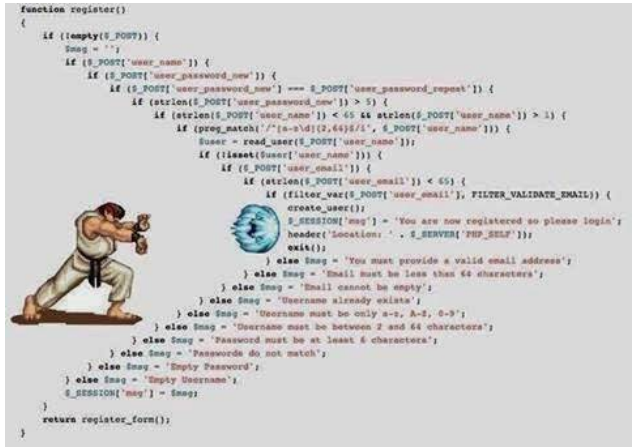
Coding Style

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



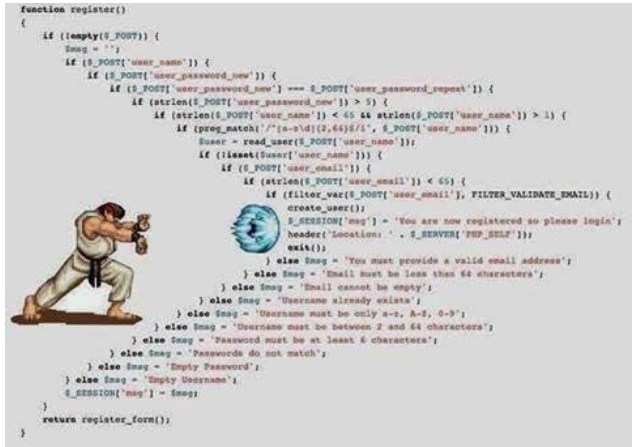
Coding Style

- **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`

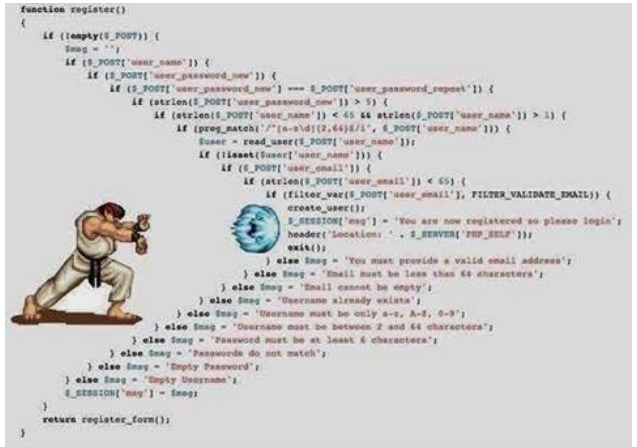


Coding Style

- **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

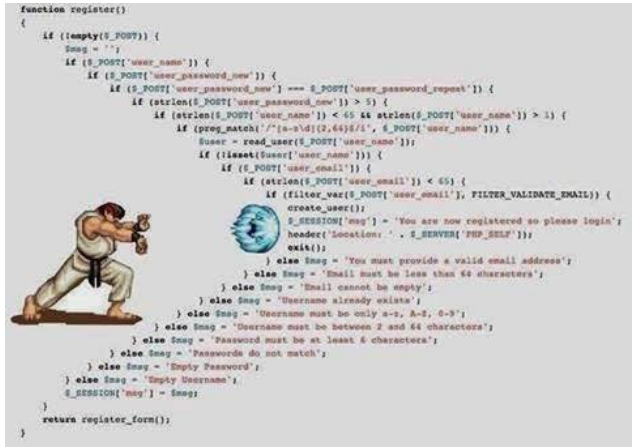


Coding Style



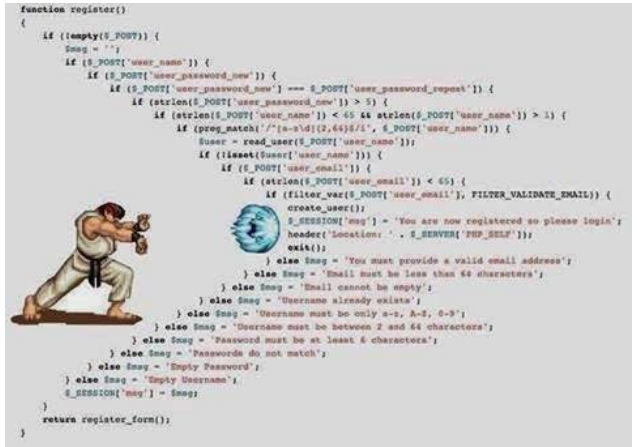
- **Indentation with tabulations** (hard tabs)
- **Spaces as a separator**
  - Item list: `[1, 2, 3]` instead of `[1,2,3]`
  - Operators: `x += 2` instead of `x+=2`
- Maximum of **120 characters per line** of code
- **Snake case**
  - Variables: `my_variable` instead of `myVariable`
  - Classes: `My_class`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

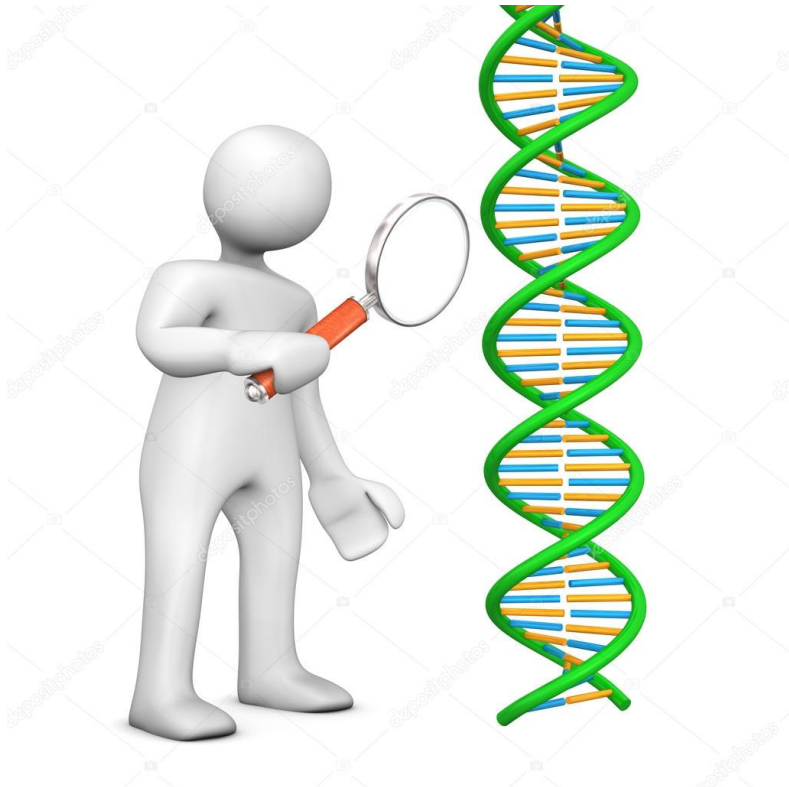


- **Indentation with tabulations** (hard tabs)
- **Spaces as a separator**
  - Item list: `[1, 2, 3]` instead of `[1,2,3]`
  - Operators: `x += 2` instead of `x+=2`
- Maximum of **120 characters per line** of code
- **Snake case**
  - Variables: `my_variable` instead of `myVariable`
  - Classes: `My_class`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

Documentation: Doxygen

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
```



Documentation: Generation with Doxygen

AFF3CT

A Fast Forward Error Correction Toolbox!

Main Page | Related Pages | Classes | Files | Search

Class List | Class Index | Class Hierarchy | Class Members

Class Index

A | B | C | D | E | F | G | H | I | L | M | N | O | P | Q | R | S | T | U

A

AList (aff3ct::tools)
API_polar (aff3ct::tools)
API_polar_dynamic_inter (aff3ct::tools)
API_polar_dynamic_inter_8bit_bitpacking (aff3ct::tools)
API_polar_dynamic_intra (aff3ct::tools)
API_polar_dynamic_seq (aff3ct::tools)
API_polar_static_inter (aff3ct::tools)
API_polar_static_inter_8bit_bitpacking (aff3ct::tools)
API_polar_static_intra_16bit (aff3ct::tools)
API_polar_static_intra_32bit (aff3ct::tools)
API_polar_static_intra_8bit (aff3ct::tools)
API_polar_static_seq (aff3ct::tools)
Argument_handler (aff3ct::tools)
Argument_info (aff3ct::tools)
Argument_map_group (aff3ct::tools)
Argument_map_info (aff3ct::tools)
Argument_map_value (aff3ct::tools)
Argument_range (aff3ct::tools)
Argument_type (aff3ct::tools)
Argument_type_limited (aff3ct::tools)
Argument_type_limited< T > (aff3ct::tools)
Argument_type_limited_T (aff3ct::tools)

B

Barrier (aff3ct::tools)
BCH (aff3ct::launcher)
BCH_polynomial_generator (aff3ct::tools)
BFER (aff3ct::factory)
BFER (aff3ct::simulation)
BFER_ite (aff3ct::launcher)
BFER_ite (aff3ct::simulation)

Decoder_BCH (aff3ct::factory)
Decoder_BCH (aff3ct::module)
Decoder_BCH_genius (aff3ct::module)
Decoder_BCH_std (aff3ct::module)
Decoder_chase_std (aff3ct::module)
Decoder_HIHO (aff3ct::module)
Decoder_LDPC (aff3ct::factory)
Decoder_LDPC_BP (aff3ct::module)
Decoder_LDPC_BP_flooding (aff3ct::module)
Decoder_LDPC_BP_flooding_approximate_min_star (aff3ct::module)
Decoder_LDPC_BP_flooding_Gallager_A (aff3ct::module)
Decoder_LDPC_BP_flooding_log_sum_product (aff3ct::module)
Decoder_LDPC_BP_flooding_offset_normalize_min_sum (aff3ct::module)
Decoder_LDPC_BP_flooding_sum_product (aff3ct::module)
Decoder_LDPC_BP_layered (aff3ct::module)
Decoder_LDPC_BP_layered_approximate_min_star (aff3ct::module)
Decoder_LDPC_BP_layered_log_sum_product (aff3ct::module)
Decoder_LDPC_BP_layered_offset_normalize_min_sum (aff3ct::module)
Decoder_LDPC_BP_layered_ONMS_inter (aff3ct::module)
Decoder_LDPC_BP_layered_sum_product (aff3ct::module)
Decoder_maximum_likelihood (aff3ct::module)
Decoder_maximum_likelihood_naive (aff3ct::module)
Decoder_maximum_likelihood_std (aff3ct::module)
Decoder_NO (aff3ct::factory)
Decoder_NO (aff3ct::module)
Decoder_polar (aff3ct::factory)
Decoder_polar_ASCL_fast_CA_sys (aff3ct::module)
Decoder_polar_ASCL_MEM_fast_CA_sys (aff3ct::module)
Decoder_polar_SC_fast_sys (aff3ct::module)
Decoder_polar_SC_naive (aff3ct::module)
Decoder_polar_SC_naive_sys (aff3ct::module)
Decoder_polar_SCAN_naive (aff3ct::module)
Decoder_polar_SCAN_naive_sys (aff3ct::module)

Encode
Encode
Encode
Encode
Encode
Encode
Encode
Encode
Encode
Encode
Encode
Encode
Examp
Excludi
EXIT (a
EXIT (a
EXIT (a

F

Factory
File_sy
File_ty
Flip_an
Flip_an
Flip_an
Flip_an
Folder_
Frame_
Frozen
Frozen

AFF3CT

A Fast Forward Error Correction Toolbox!

Main Page | Related Pages | Classes | Files | Search

Class List | Class Index | Class Hierarchy | Class Members

aff3ct | module | Module

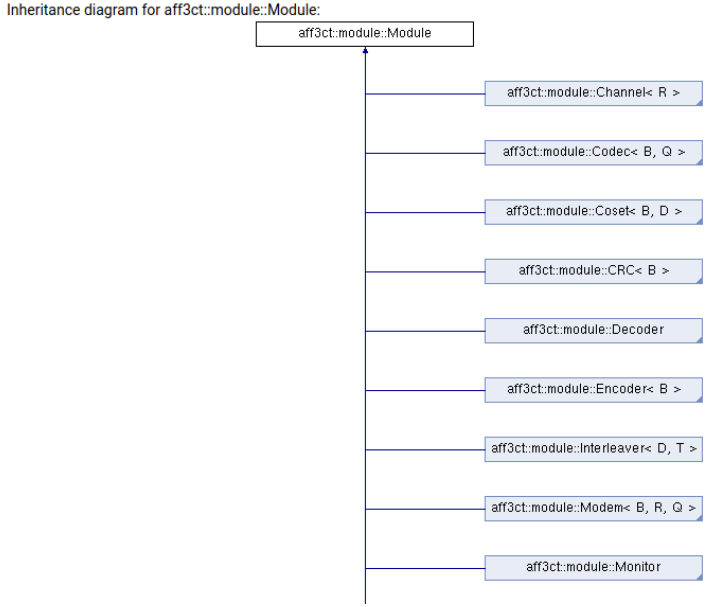
aff3ct::module::Module Class Reference

Public Member Functions | Public Attributes | Protected Member Functions | Protected Attributes | List of all members

A Module is an abstract concept. Basically, all the objects used in a Simulation are a Module. More...

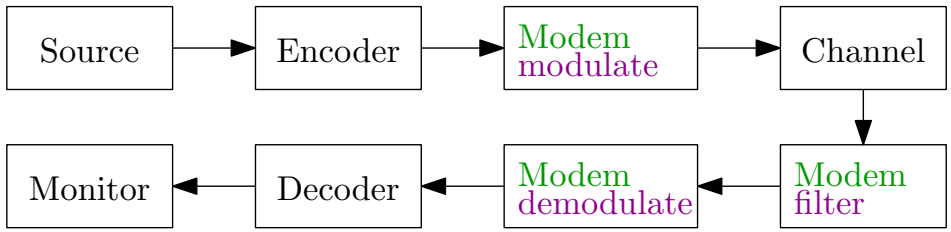
`#include <Module.hpp>`

Inheritance diagram for aff3ct::module::Module:



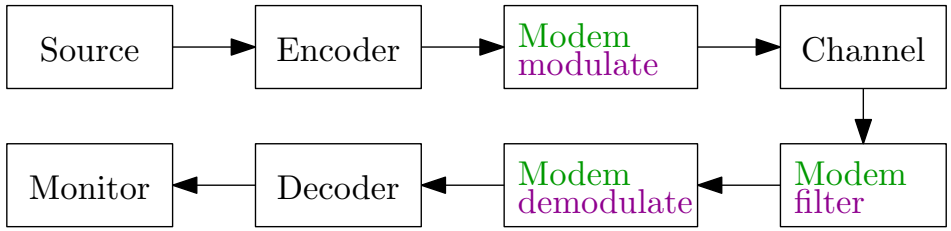
Adding a New Module in AFF3CT

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



Adding a New Module in AFF3CT

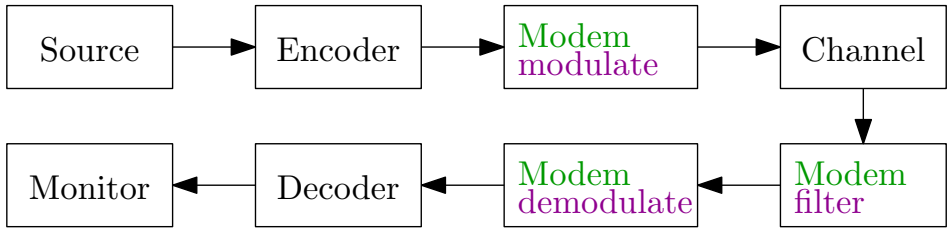
Let's suppose we want to add a new `Modem` named `2PSK`



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

Adding a New Module in AFF3CT

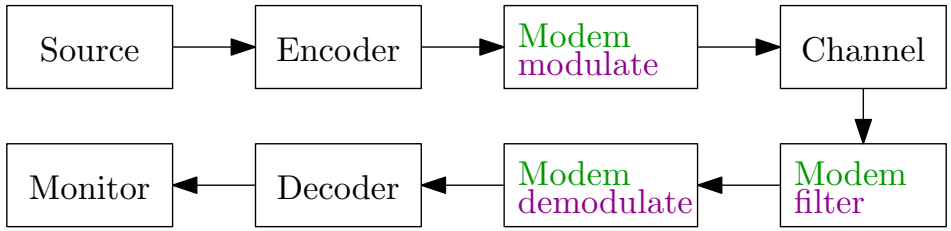
Let's suppose we want to add a new Modem named 2PSK



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

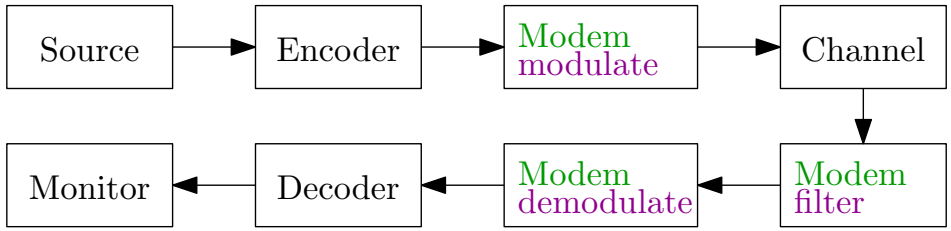
Let's suppose we want to add a new Modem named 2PSK



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**

Example: Adding a Modulator, Modem_2PSK.hpp

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

```
 1 #ifndef MODEM_2PSK_HPP_
 2 #define MODEM_2PSK_HPP_
 3
 4 #include "Module/Modem/Modem.hpp"
 5
 6 namespace aff3ct{ namespace module{
 7
 8 // B is the type of bits, R and Q are the types of real data
 9 template <typename B = int, typename R = float, typename Q = R>
10 // inherits from Modem class
11 class Modem_2PSK : public Modem<B,R,Q>
12 {
13   // class constructor and destructor
14   public:
15   Modem_2PSK(const int N, const R sigma) : Modem<B,R,Q>(N, sigma, 1) {}
16   virtual ~Modem_2PSK() {}
17
18   // specific modem interface (= modem tasks)
19   protected:
20   void    _modulate(const B *X_N1, R *X_N2, const int frame_id);
21   void      _filter(const R *Y_N1, R *Y_N2, const int frame_id);
22   void _demodulate(const Q *Y_N1, Q *Y_N2, const int frame_id);
23 };
24 }}
25 #endif /* MODEM_2PSK_HPP_ */
```

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);
}
```

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:

```
1  // ...
2  #include "Module/Modem/2PSK/Modem_2PSK.hpp"
3  // ...
4  void Modem::parameters
5  ::get_description(tools::Argument_map_info &args) const
6  {
7    // ...
8    args.add(
9      {p+"-type"},
10     tools::Text(tools::Including_set("...", "2PSK")),
11     "type of the modulation to use in the simulation.");
12   // ...
13 }
14 // ...
15 template <typename B, typename R, typename Q, tools::proto_max<Q> MAX>
16 module::Modem<B,R,Q>* Modem::parameters
17 ::_build() const
18 {
19   // ...
20   else if (this->type == "2PSK")
21     return new module::Modem_2PSK<B,R,Q>(this->N, this->sigma);
22   // ...
23 }
24 // ...
```

Example: Adding a Modulator to the Factory

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

```
1  // ...
2  int Modem
3  ::get_buffer_size_after_modulation(const std::string &type,
4                                     const int          N,
5                                     const int          bps,
6                                     const int          upf,
7                                     const int          cpm_L,
8                                     const int          cpm_p)
9  {
10   // ...
11   else if (type == "2PSK") return N;
12   // ...
13 }
14
15 int Modem
16 ::get_buffer_size_after_filtering(const std::string &type,
17                                    const int          N,
18                                    const int          bps,
19                                    const int          cpm_L,
20                                    const int          cpm_p)
21 {
22   // ...
23   else if (type == "2PSK") return N;
24   // ...
25 }
26 // ...
```

Example: Run the New Modem

Compile and Run

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

Example: Run the New Modem

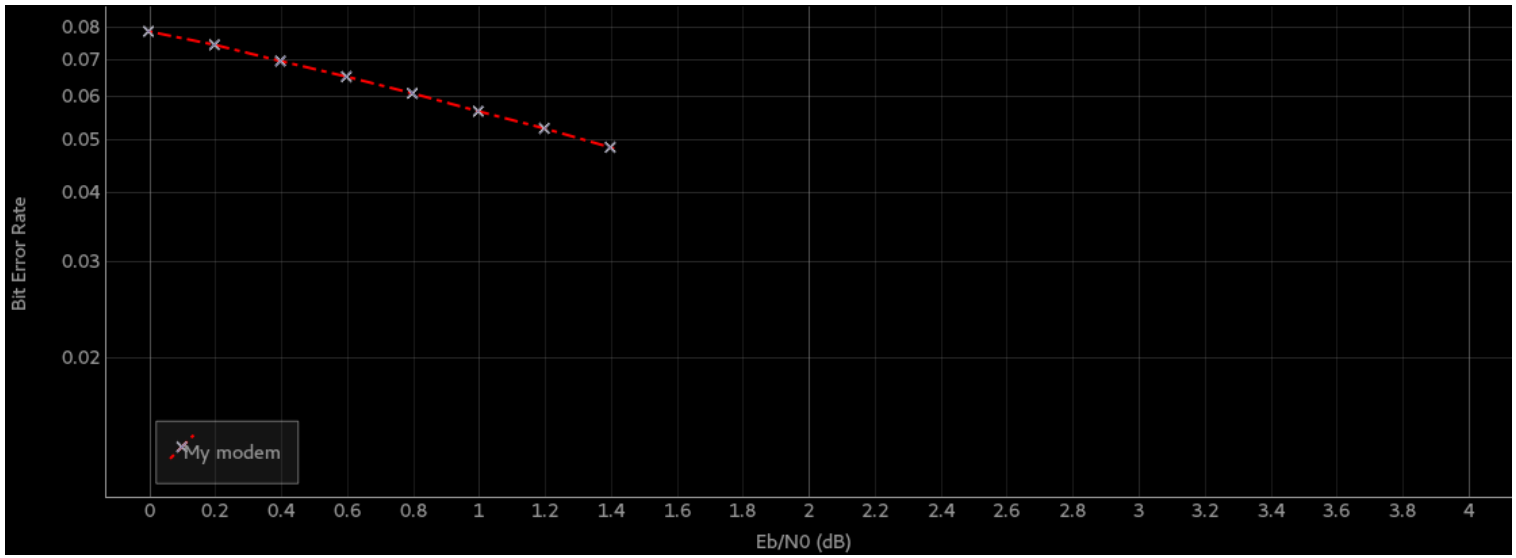
Compile and Run

```
$ 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
```

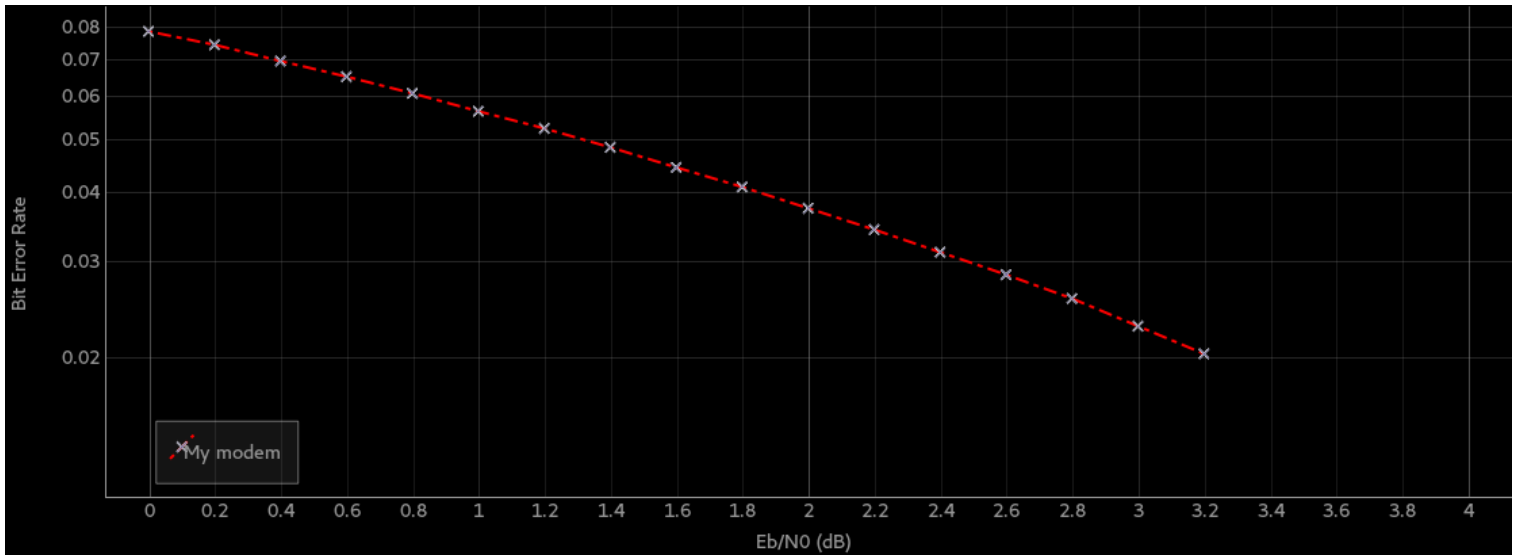


Live AFF3CT simulation on PyBER

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
```

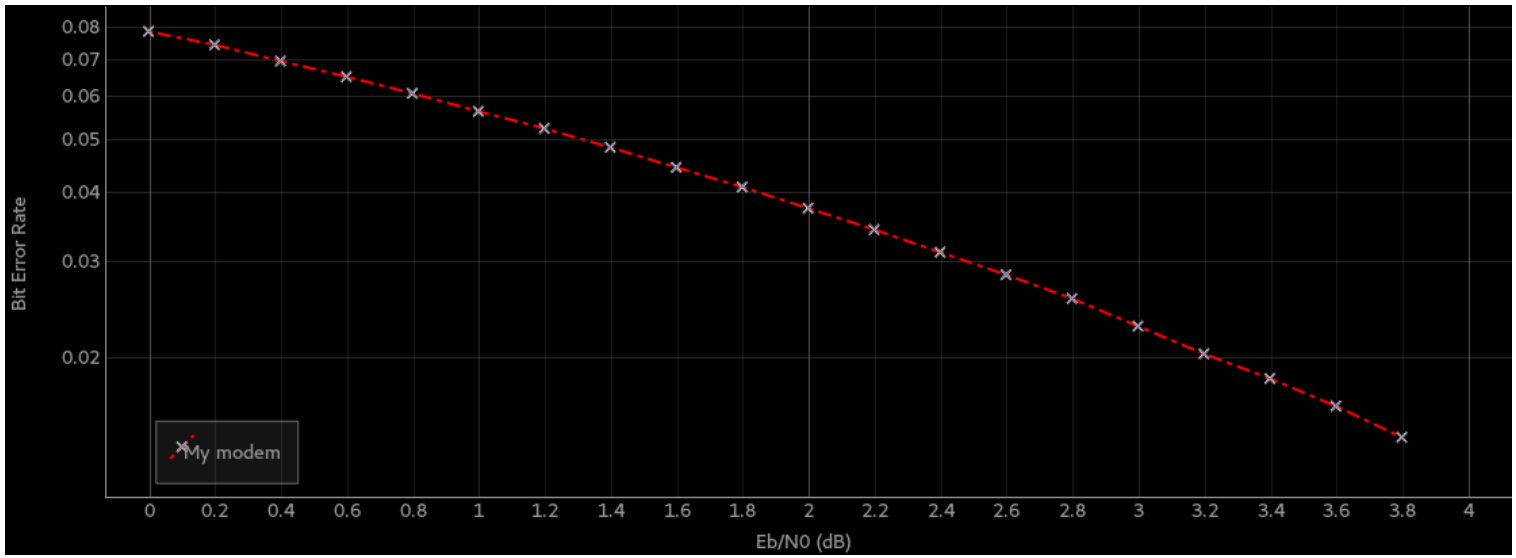


Live AFF3CT simulation on PyBER

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
```

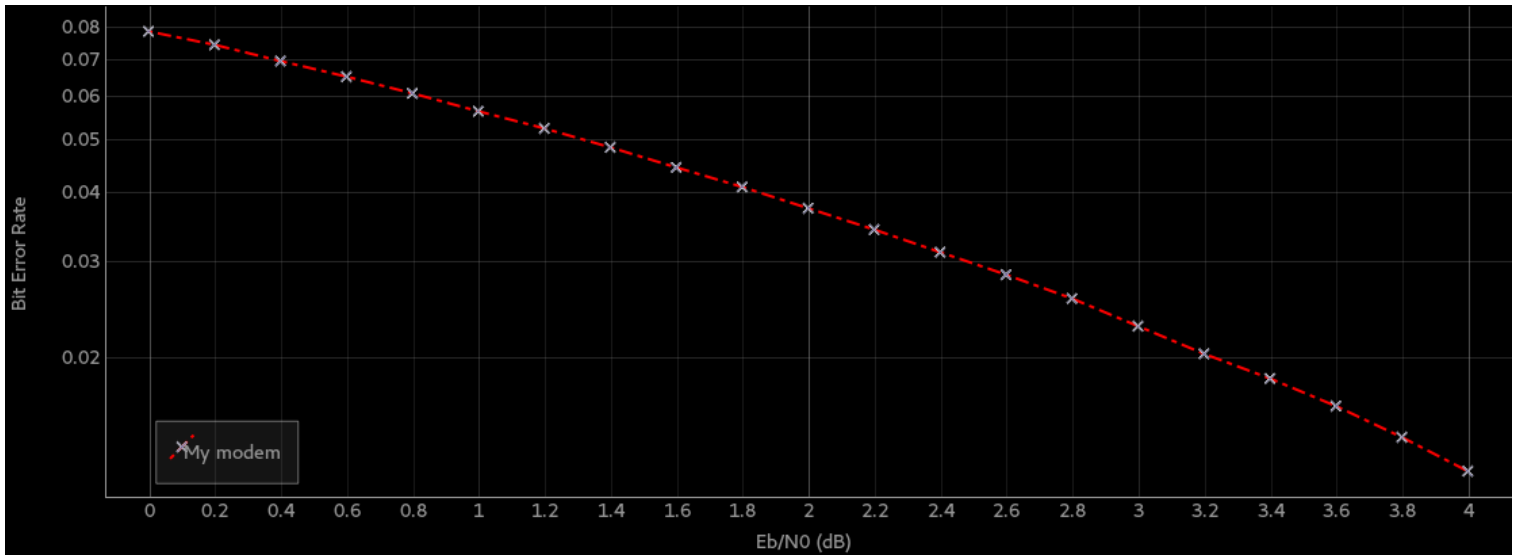


Live AFF3CT simulation on PyBER

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

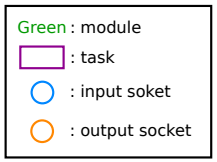
- Low level features
  - **Code-related**: Alist/QC readers, Polar functions API, frozen bits generators, Galois Field generator, ...
  - **Miscellaneous**: Matrix operations, sparse matrices, binary trees, ...

My Project with AFF3CT

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

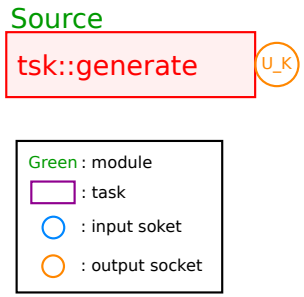
- Low level features
  - **Code-related**: Alist/QC readers, Polar functions API, frozen bits generators, Galois Field generator, ...
  - **Miscellaneous**: Matrix operations, sparse matrices, binary trees, ...
- High level features
  - **Factories**: Command line arguments management, automatic objects instantiation, ...
  - **Modules**: Sources, CRC, encoders, decoders, modems, channels, ...

My Project with AFF3CT: Modules Allocation



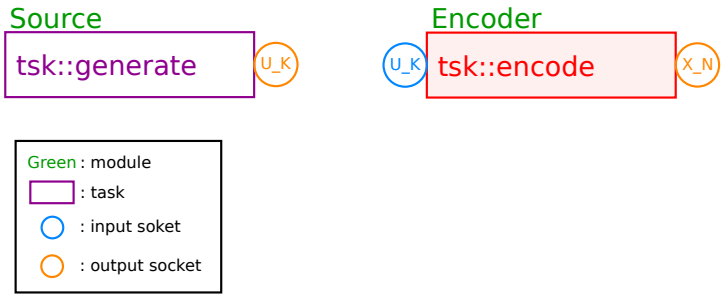

```
1  #include <aff3ct.hpp>
2  using namespace aff3ct;
3
4  // allocate the module objects
5
6
7
8
9
10
```

My Project with AFF3CT: Modules Allocation



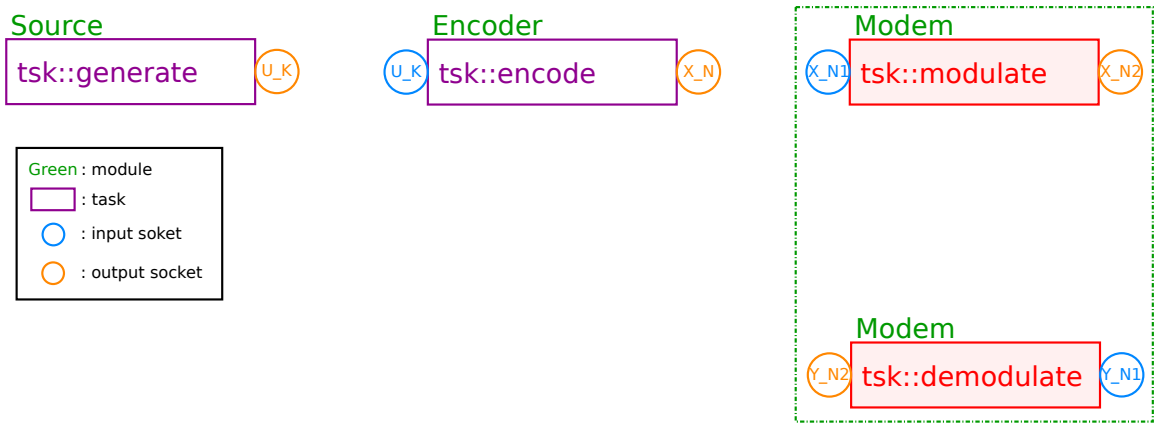

```
1  #include <aff3ct.hpp>
2  using namespace aff3ct;
3
4  // allocate the module objects
5  module::Source_random<>          source (K   );
6
7
8
9
10
```

My Project with AFF3CT: Modules Allocation



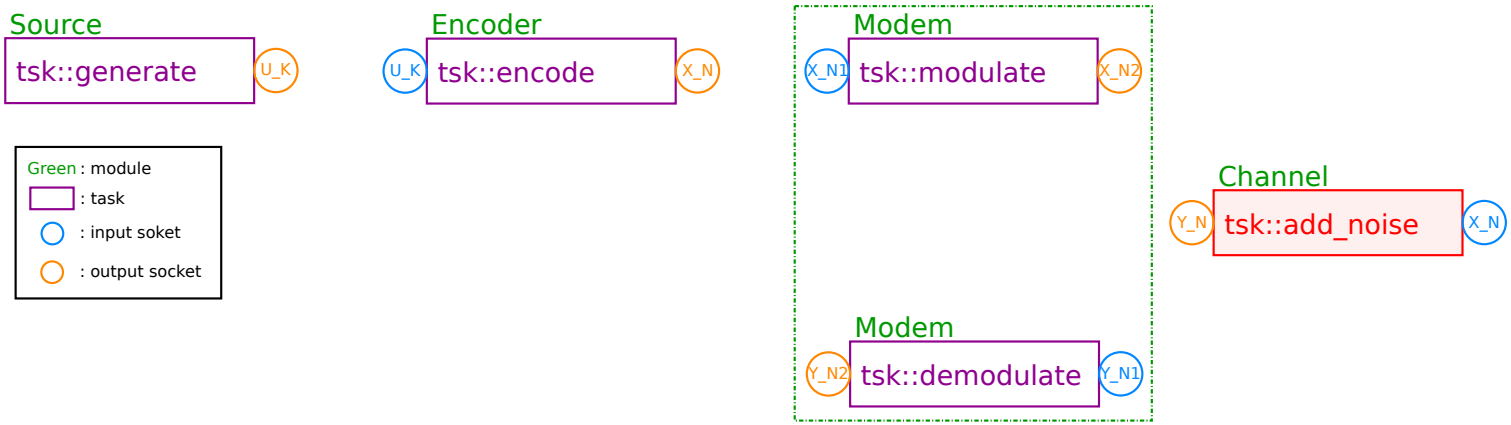

```
1  #include <aff3ct.hpp>
2  using namespace aff3ct;
3
4  // allocate the module objects
5  module::Source_random<>          source (K   );
6  module::Encoder_repetition_sys<> encoder(K, N);
7
8
9
10
```

My Project with AFF3CT: Modules Allocation



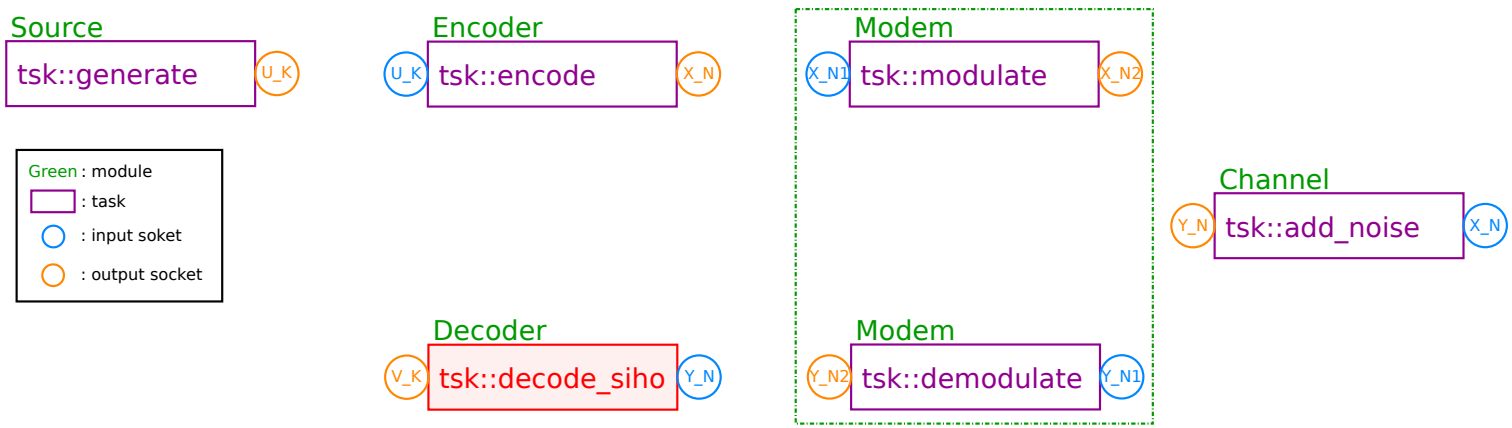

```
1  #include <aff3ct.hpp>
2  using namespace aff3ct;
3
4  // allocate the module objects
5  module::Source_random<>          source (K   );
6  module::Encoder_repetition_sys<> encoder(K, N);
7  module::Modem_BPSK<>             modem  (N   );
8
9
10
```

My Project with AFF3CT: Modules Allocation



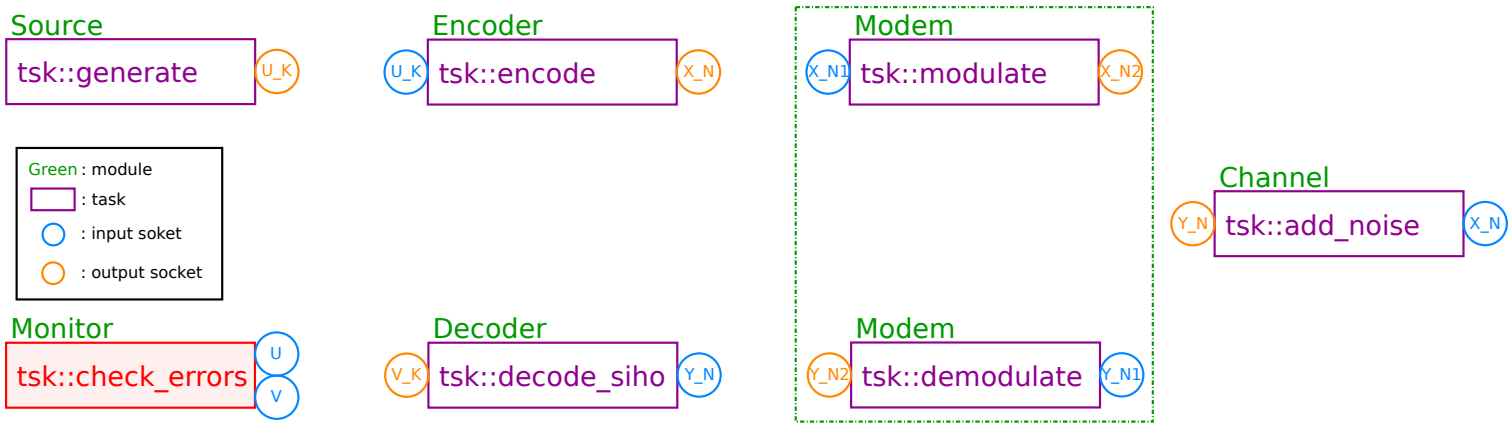

```
1  #include <aff3ct.hpp>
2  using namespace aff3ct;
3
4  // allocate the module objects
5  module::Source_random<>          source (K   );
6  module::Encoder_repetition_sys<> encoder(K, N);
7  module::Modem_BPSK<>             modem  (N   );
8  module::Channel_AWGN_LLR<>       channel(N   );
9
10
```

My Project with AFF3CT: Modules Allocation



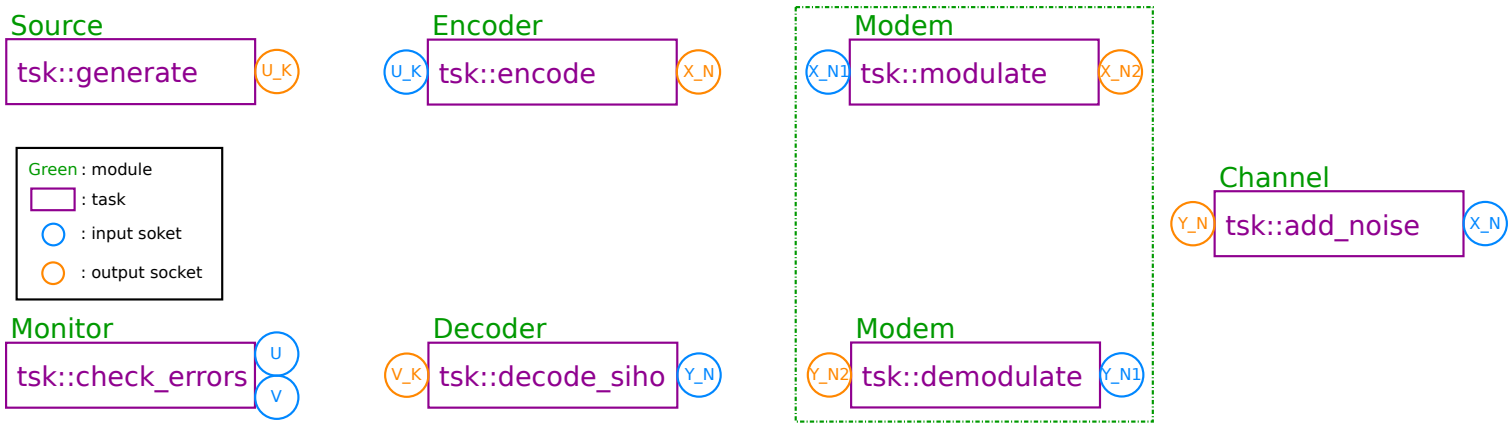

```
1  #include <aff3ct.hpp>
2  using namespace aff3ct;
3
4  // allocate the module objects
5  module::Source_random<>          source (K   );
6  module::Encoder_repetition_sys<> encoder(K, N);
7  module::Modem_BPSK<>             modem  (N   );
8  module::Channel_AWGN_LLR<>       channel(N   );
9  module::Decoder_repetition_std<> decoder(K, N);
10
```

My Project with AFF3CT: Modules Allocation



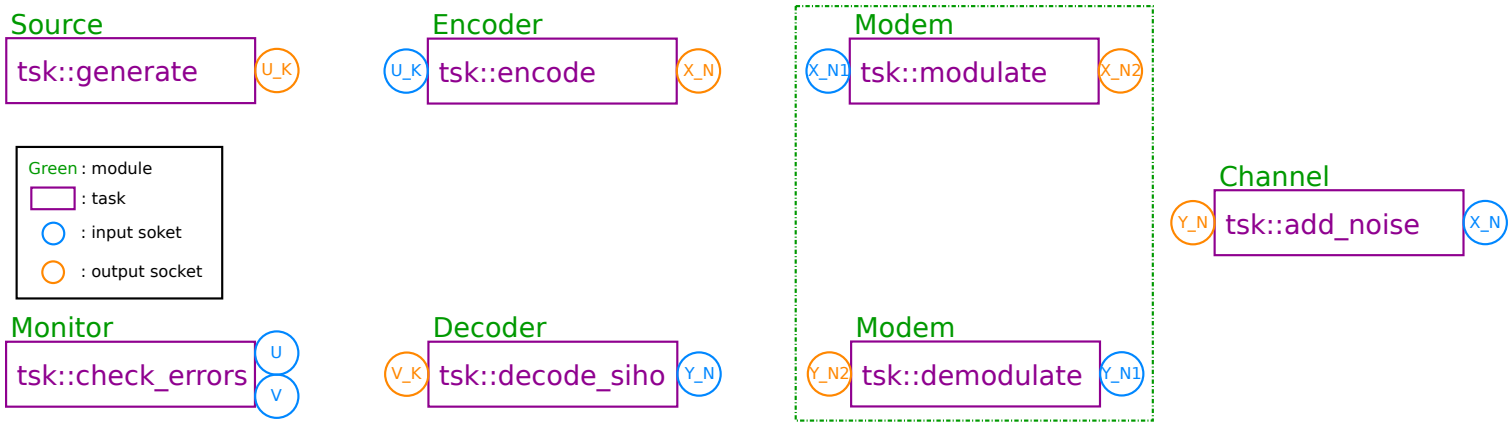

```
1  #include <aff3ct.hpp>
2  using namespace aff3ct;
3
4  // allocate the module objects
5  module::Source_random<>          source (K   );
6  module::Encoder_repetition_sys<> encoder(K, N);
7  module::Modem_BPSK<>             modem  (N   );
8  module::Channel_AWGN_LLR<>       channel(N   );
9  module::Decoder_repetition_std<> decoder(K, N);
10 module::Monitor_BFER<>           monitor(K, E);
```

My Project with AFF3CT: Modules Allocation



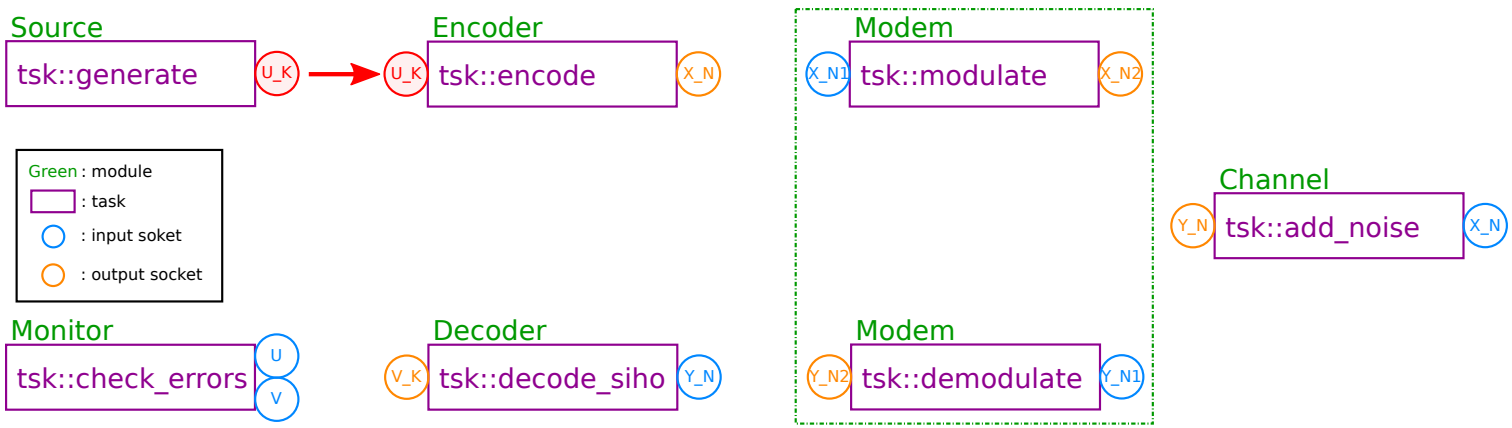

```
1  #include <aff3ct.hpp>
2  using namespace aff3ct;
3
4  // allocate the module objects
5  module::Source_random<>          source (K   );
6  module::Encoder_repetition_sys<> encoder(K, N);
7  module::Modem_BPSK<>             modem  (N   );
8  module::Channel_AWGN_LLR<>       channel(N   );
9  module::Decoder_repetition_std<> decoder(K, N);
10 module::Monitor_BFER<>           monitor(K, E);
```

My Project with AFF3CT: Sockets Binding



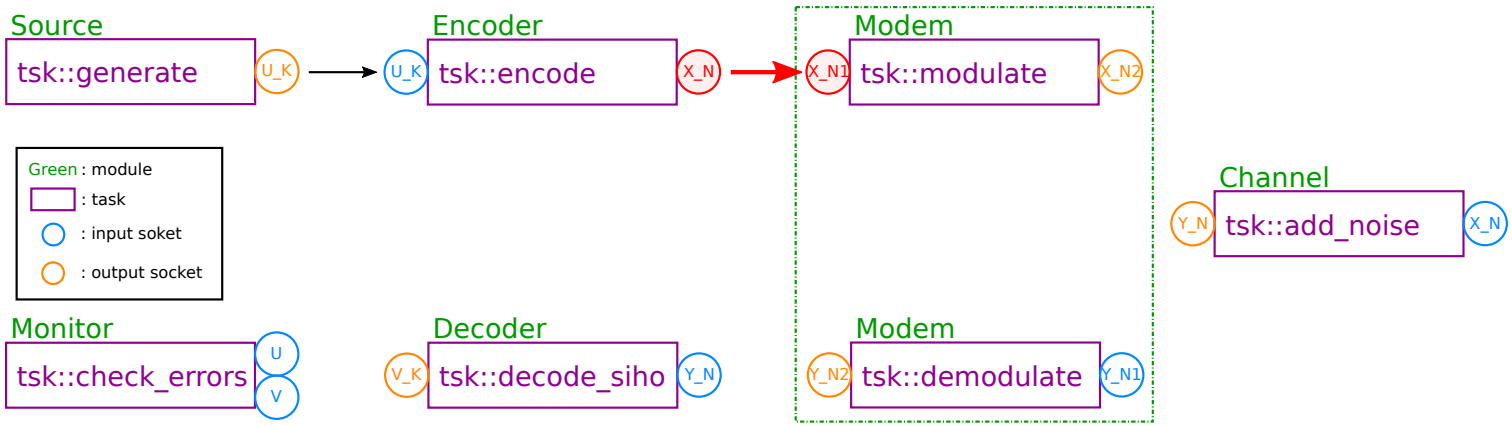

```
1  using namespace aff3ct::module;
2
3  // bind the sockets over the tasks
4
5
6
7
8
9
10
```

My Project with AFF3CT: Sockets Binding



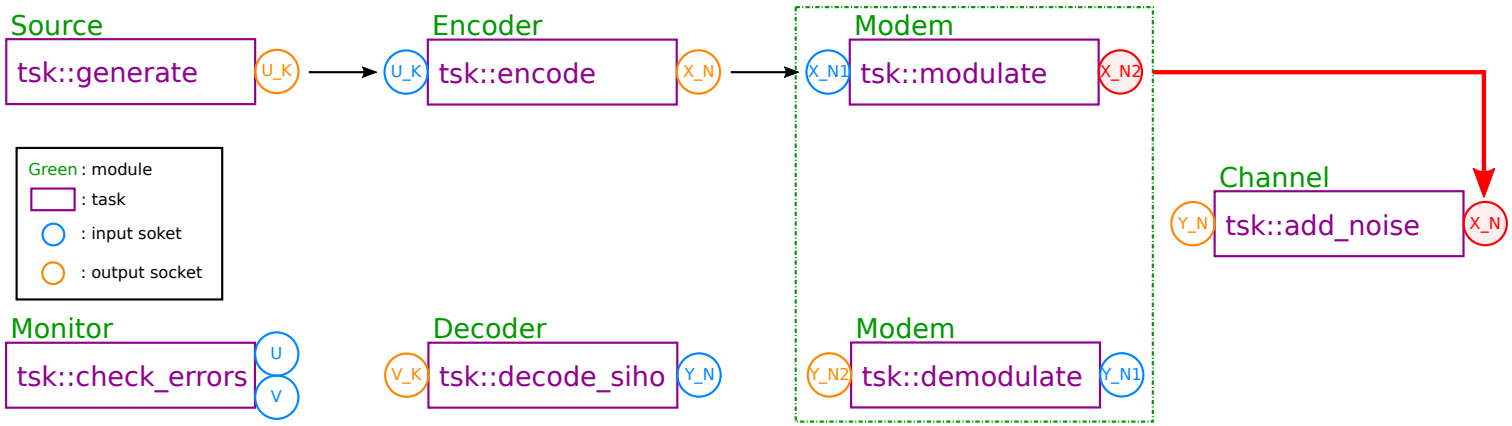

```
1  using namespace aff3ct::module;
2
3  // bind the sockets over the tasks
4  encoder[enc::sck::encode      ::U_K ].bind( source [src::sck::generate   ::U_K ] );
5
6
7
8
9
10
```

My Project with AFF3CT: Sockets Binding



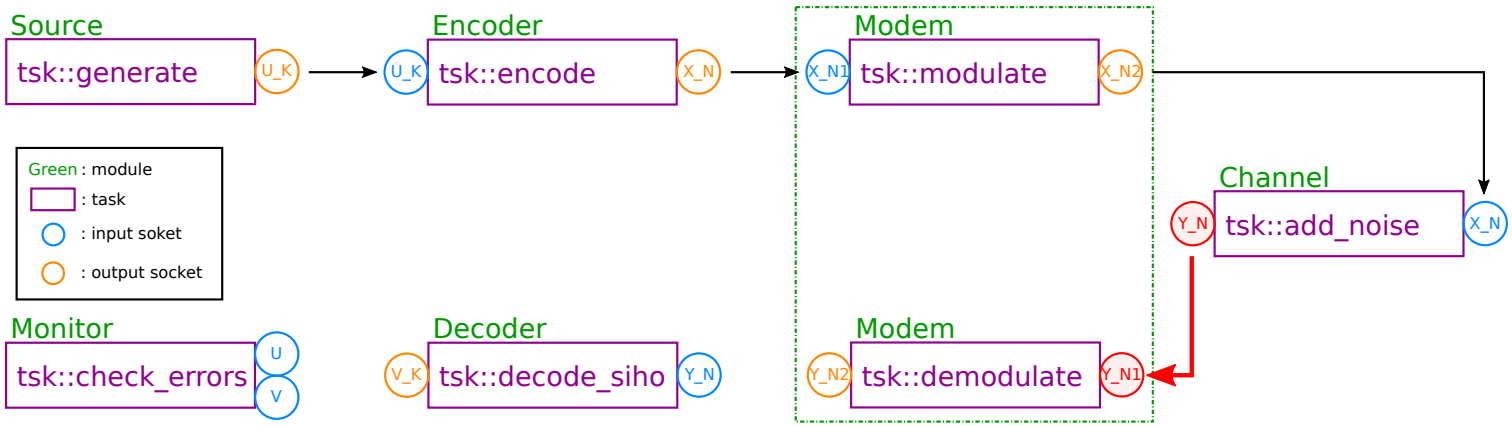

```
1  using namespace aff3ct::module;
2
3  // bind the sockets over the tasks
4  encoder[enc::sck::encode      ::U_K ].bind( source [src::sck::generate   ::U_K ] );
5  modem  [mdm::sck::modulate    ::X_N1].bind( encoder[enc::sck::encode     ::X_N ] );
6
7
8
9
10
```

My Project with AFF3CT: Sockets Binding



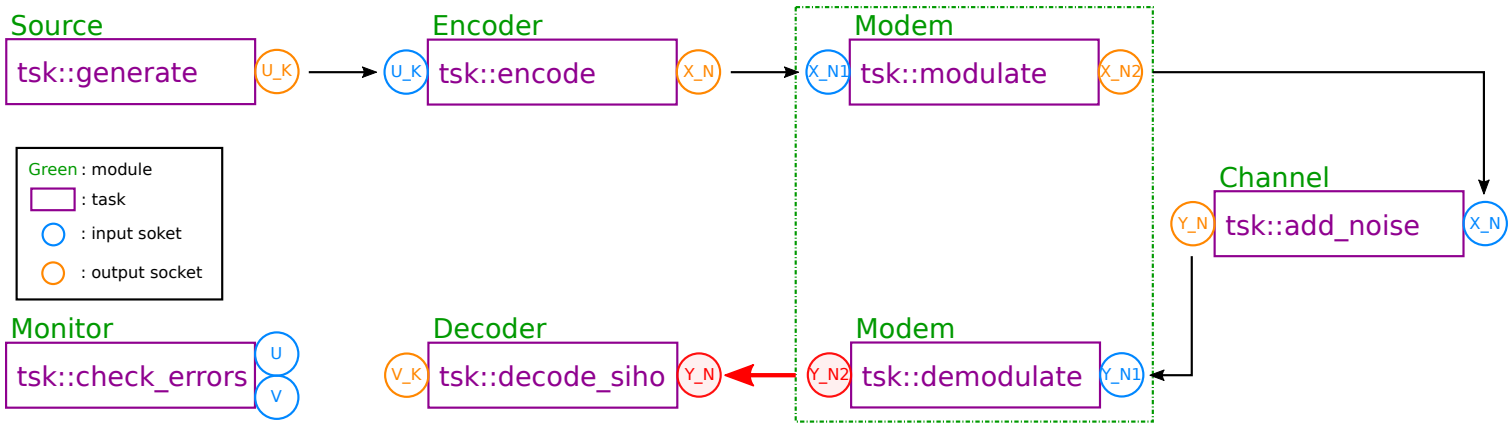

```
1  using namespace aff3ct::module;
2
3  // bind the sockets over the tasks
4  encoder[enc::sck::encode      ::U_K ].bind( source [src::sck::generate   ::U_K ] );
5  modem   [mdm::sck::modulate   ::X_N1].bind( encoder[enc::sck::encode     ::X_N ] );
6  channel[chn::sck::add_noise   ::X_N ].bind( modem  [mdm::sck::modulate   ::X_N2] );
7
8
9
10
```

My Project with AFF3CT: Sockets Binding



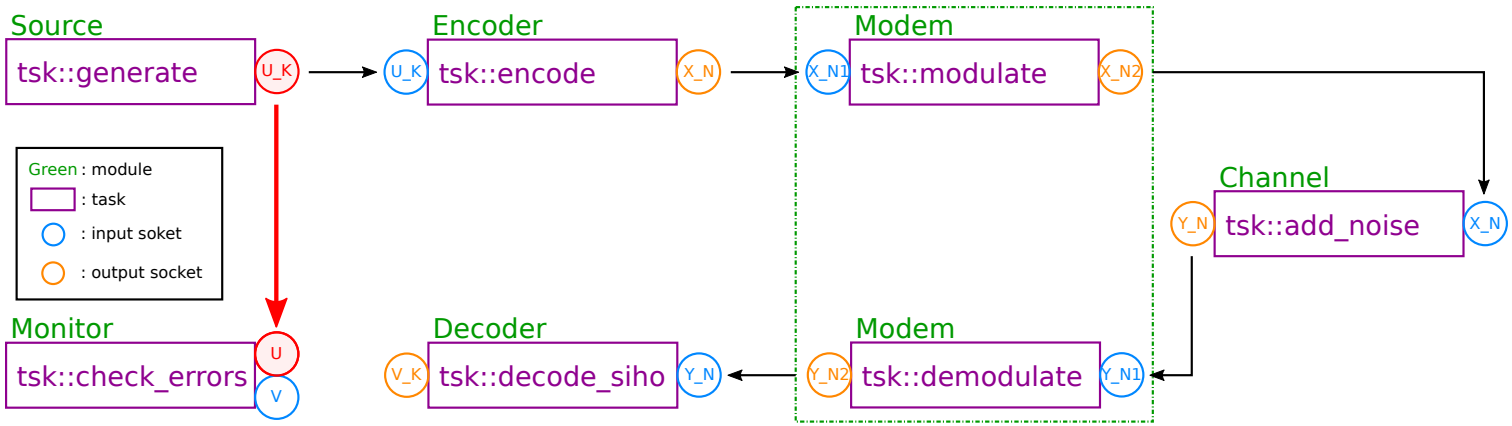

```
1  using namespace aff3ct::module;
2
3  // bind the sockets over the tasks
4  encoder[enc::sck::encode      ::U_K ].bind( source [src::sck::generate   ::U_K ] );
5  modem  [mdm::sck::modulate    ::X_N1].bind( encoder[enc::sck::encode     ::X_N ] );
6  channel[chn::sck::add_noise   ::X_N ].bind( modem  [mdm::sck::modulate   ::X_N2] );
7  modem  [mdm::sck::demodulate  ::Y_N1].bind( channel[chn::sck::add_noise  ::Y_N ] );
8
9
10
```

My Project with AFF3CT: Sockets Binding



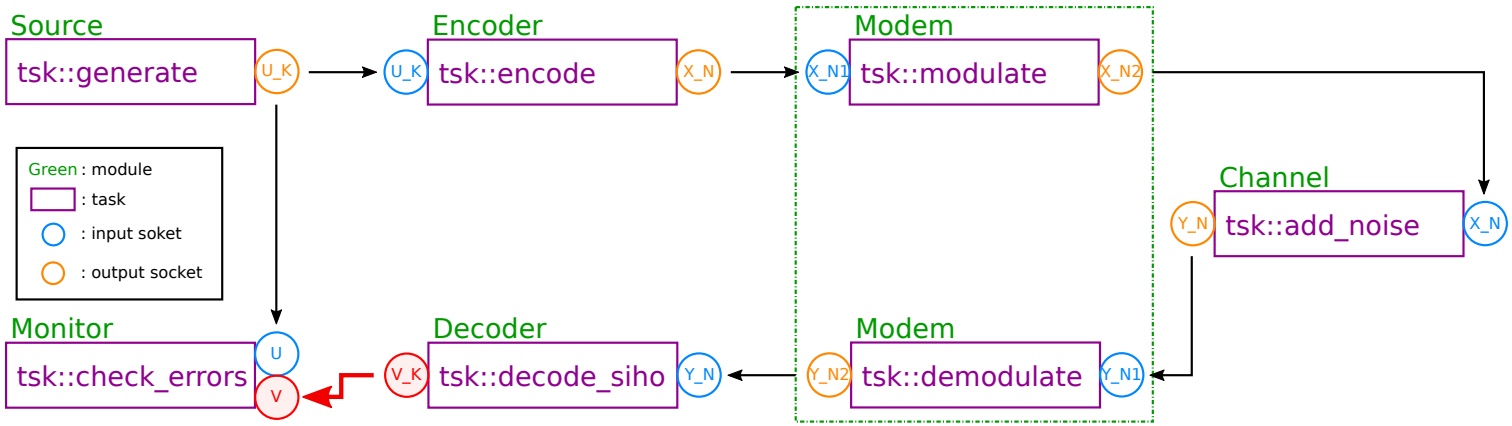

```
1  using namespace aff3ct::module;
2
3  // bind the sockets over the tasks
4  encoder[enc::sck::encode      ::U_K ].bind( source [src::sck::generate   ::U_K ] );
5  modem  [mdm::sck::modulate    ::X_N1].bind( encoder[enc::sck::encode     ::X_N ] );
6  channel[chn::sck::add_noise   ::X_N ].bind( modem  [mdm::sck::modulate   ::X_N2] );
7  modem  [mdm::sck::demodulate  ::Y_N1].bind( channel[chn::sck::add_noise  ::Y_N ] );
8  decoder[dec::sck::decode_siho ::Y_N ].bind( modem  [mdm::sck::demodulate ::Y_N2] );
9
10
```

My Project with AFF3CT: Sockets Binding



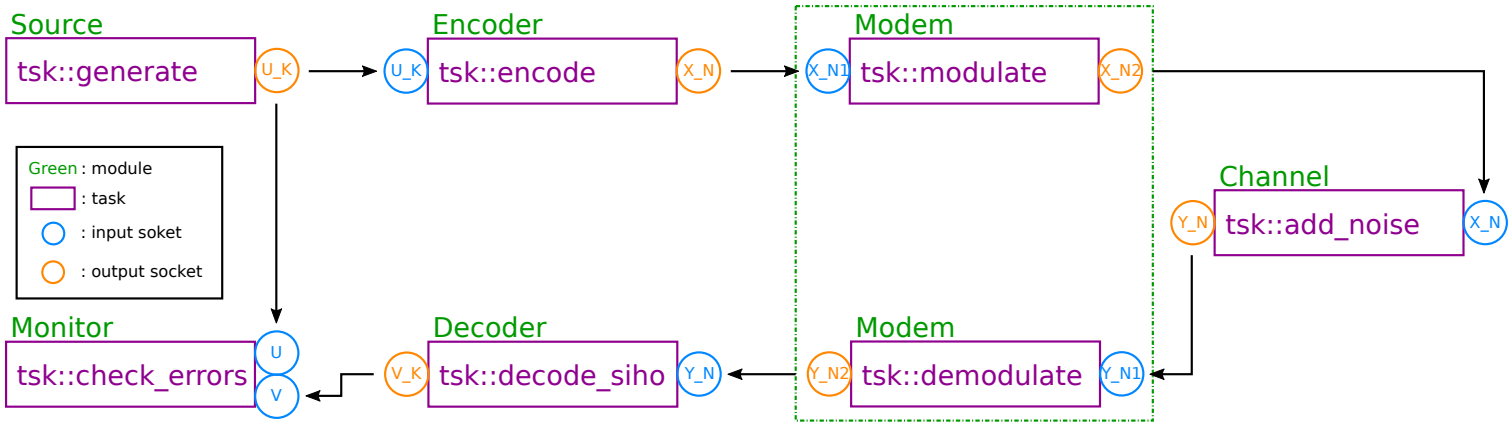

```
1  using namespace aff3ct::module;
2
3  // bind the sockets over the tasks
4  encoder[enc::sck::encode      ::U_K ].bind( source [src::sck::generate   ::U_K ] );
5  modem  [mdm::sck::modulate    ::X_N1].bind( encoder[enc::sck::encode     ::X_N ] );
6  channel[chn::sck::add_noise   ::X_N ].bind( modem  [mdm::sck::modulate   ::X_N2] );
7  modem  [mdm::sck::demodulate  ::Y_N1].bind( channel[chn::sck::add_noise  ::Y_N ] );
8  decoder[dec::sck::decode_siho ::Y_N ].bind( modem  [mdm::sck::demodulate ::Y_N2] );
9  monitor[mnt::sck::check_errors::U   ].bind( source [src::sck::generate   ::U_K ] );
10
```

My Project with AFF3CT: Sockets Binding



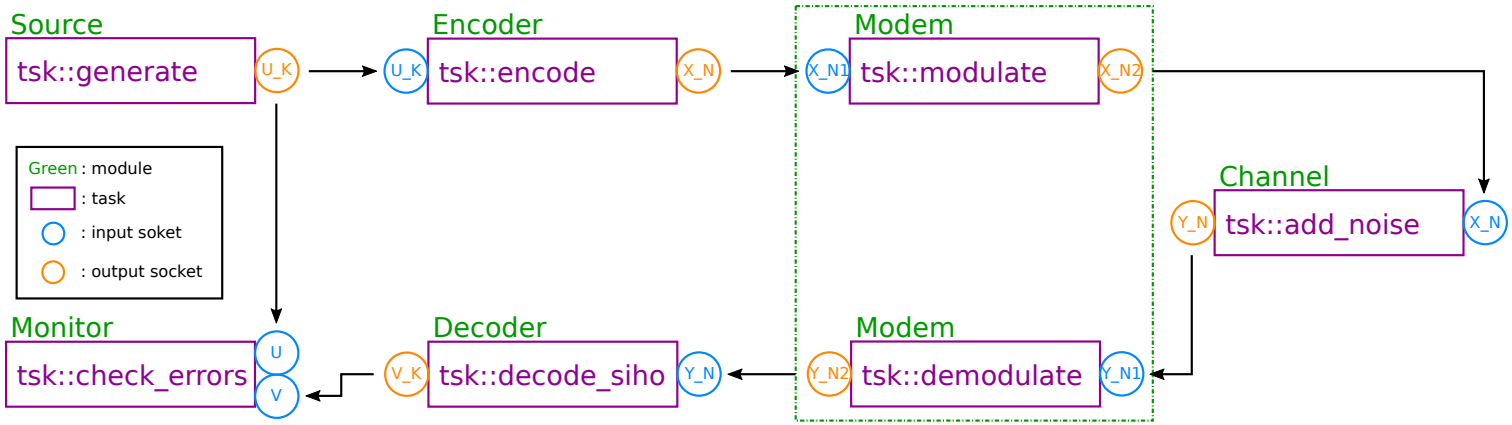

```
1  using namespace aff3ct::module;
2
3  // bind the sockets over the tasks
4  encoder[enc::sck::encode      ::U_K ].bind( source [src::sck::generate   ::U_K ] );
5  modem  [mdm::sck::modulate    ::X_N1].bind( encoder[enc::sck::encode     ::X_N ] );
6  channel[chn::sck::add_noise   ::X_N ].bind( modem  [mdm::sck::modulate   ::X_N2] );
7  modem  [mdm::sck::demodulate  ::Y_N1].bind( channel[chn::sck::add_noise  ::Y_N ] );
8  decoder[dec::sck::decode_siho ::Y_N ].bind( modem  [mdm::sck::demodulate ::Y_N2] );
9  monitor[mnt::sck::check_errors::U   ].bind( source [src::sck::generate   ::U_K ] );
10 monitor[mnt::sck::check_errors::V   ].bind( decoder[dec::sck::decode_siho::V_K ] );
```

My Project with AFF3CT: Sockets Binding



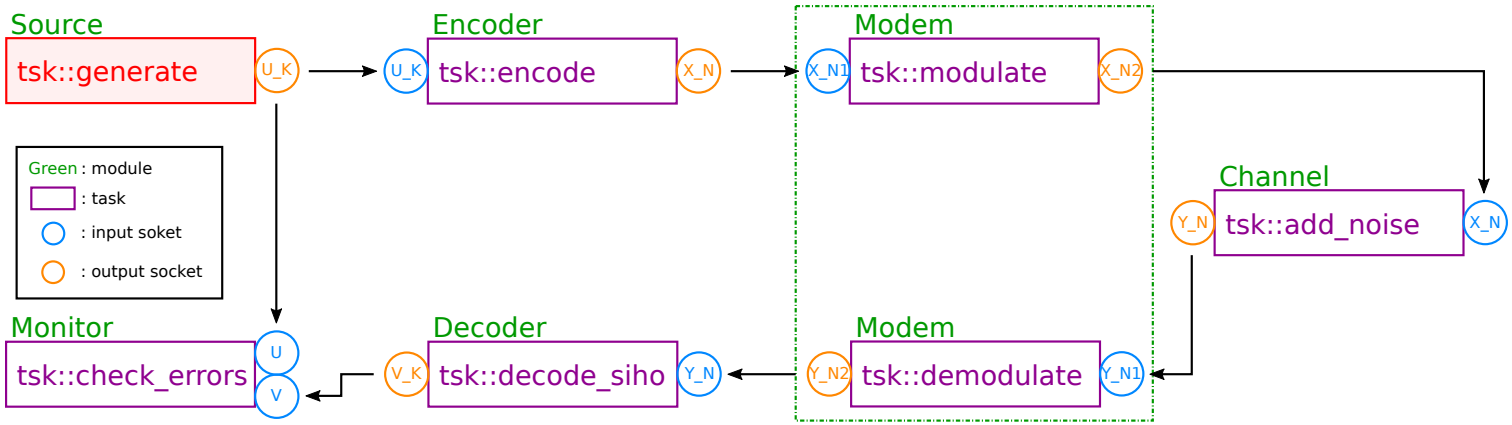

```
1  using namespace aff3ct::module;
2
3  // bind the sockets over the tasks
4  encoder[enc::sck::encode      ::U_K ].bind( source [src::sck::generate   ::U_K ] );
5  modem  [mdm::sck::modulate    ::X_N1].bind( encoder[enc::sck::encode     ::X_N ] );
6  channel[chn::sck::add_noise   ::X_N ].bind( modem  [mdm::sck::modulate   ::X_N2] );
7  modem  [mdm::sck::demodulate  ::Y_N1].bind( channel[chn::sck::add_noise  ::Y_N ] );
8  decoder[dec::sck::decode_siho ::Y_N ].bind( modem  [mdm::sck::demodulate ::Y_N2] );
9  monitor[mnt::sck::check_errors::U   ].bind( source [src::sck::generate   ::U_K ] );
10 monitor[mnt::sck::check_errors::V   ].bind( decoder[dec::sck::decode_siho::V_K ] );
```

My Project with AFF3CT: Tasks Execution



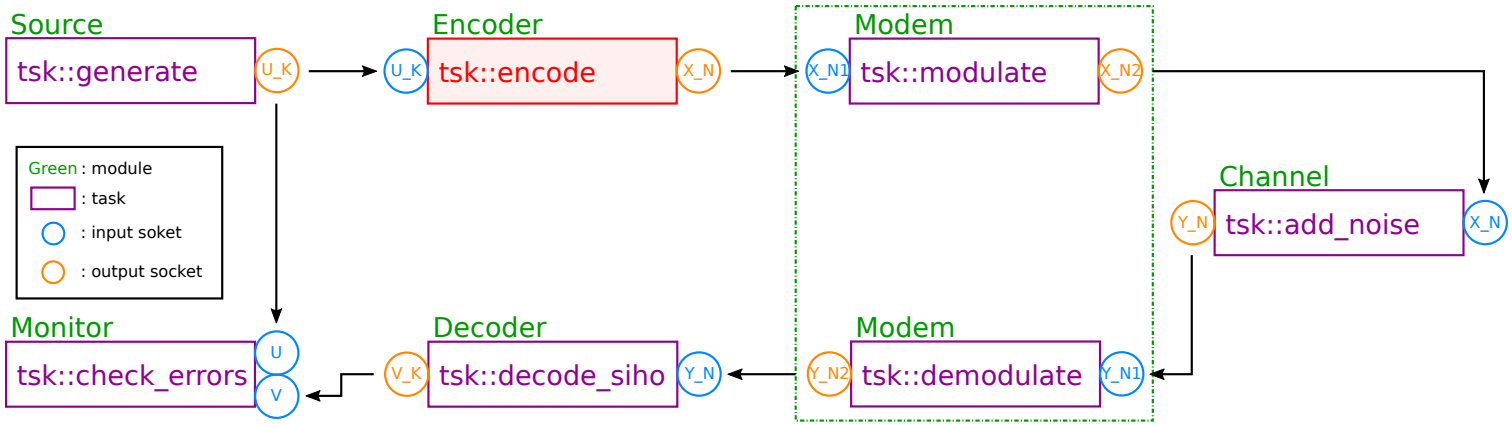

```
1  // the simulation loop
2  while (!monitor.fe_limit_achieved()) {
3
4
5
6
7
8
9
10 }
```

My Project with AFF3CT: Tasks Execution



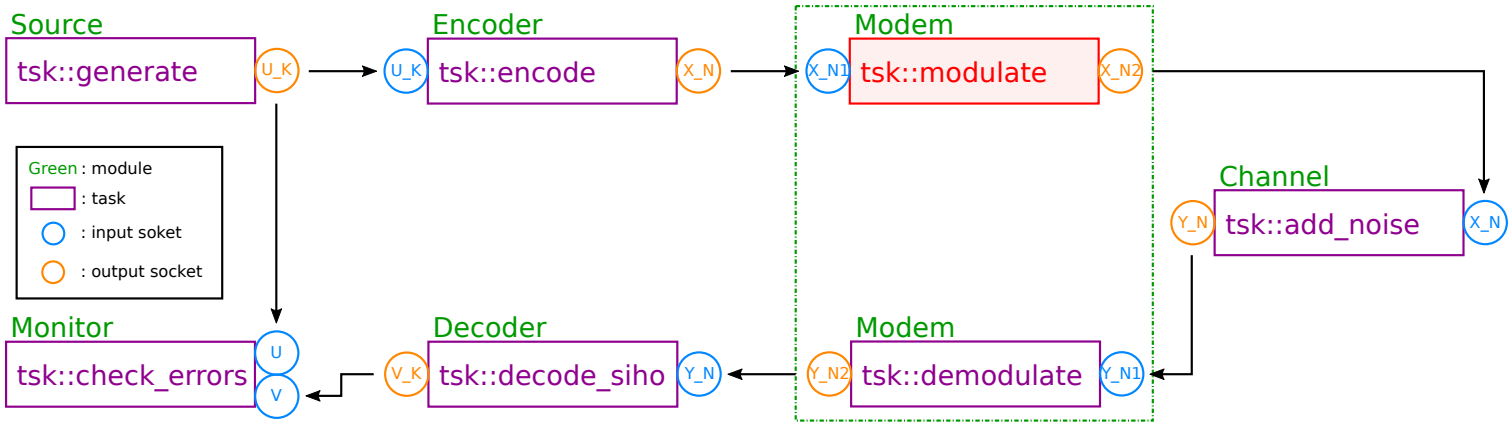

```
1  // the simulation loop
2  while (!monitor.fe_limit_achieved()) {
3    source [src::tsk::generate    ].exec();
4
5
6
7
8
9
10 }
```

My Project with AFF3CT: Tasks Execution



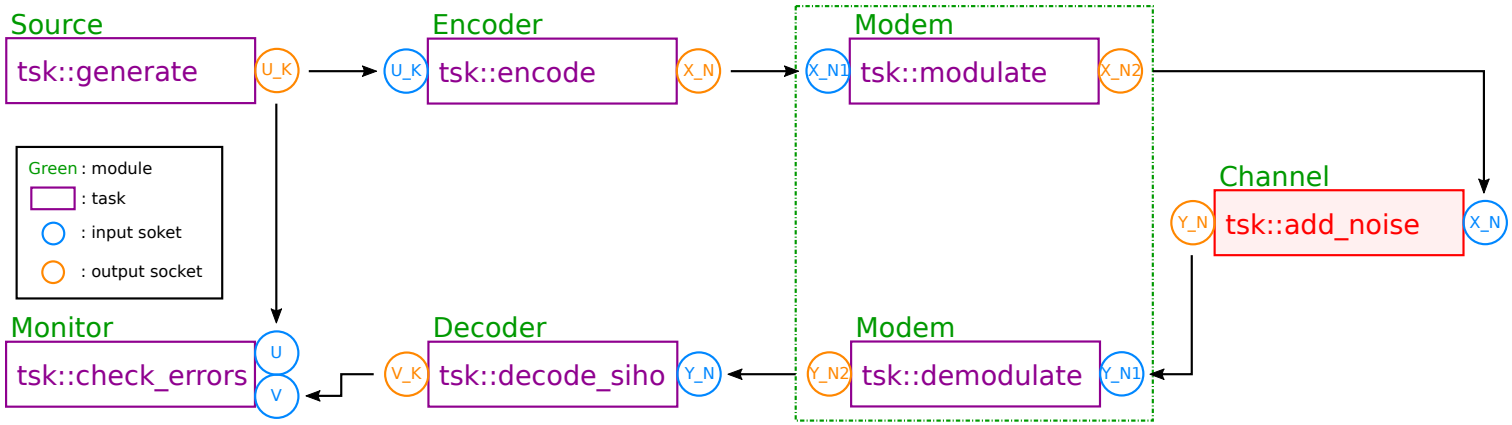

```
1  // the simulation loop
2  while (!monitor.fe_limit_achieved()) {
3    source [src::tsk::generate    ].exec();
4    encoder[enc::tsk::encode      ].exec();
5
6
7
8
9
10 }
```

My Project with AFF3CT: Tasks Execution



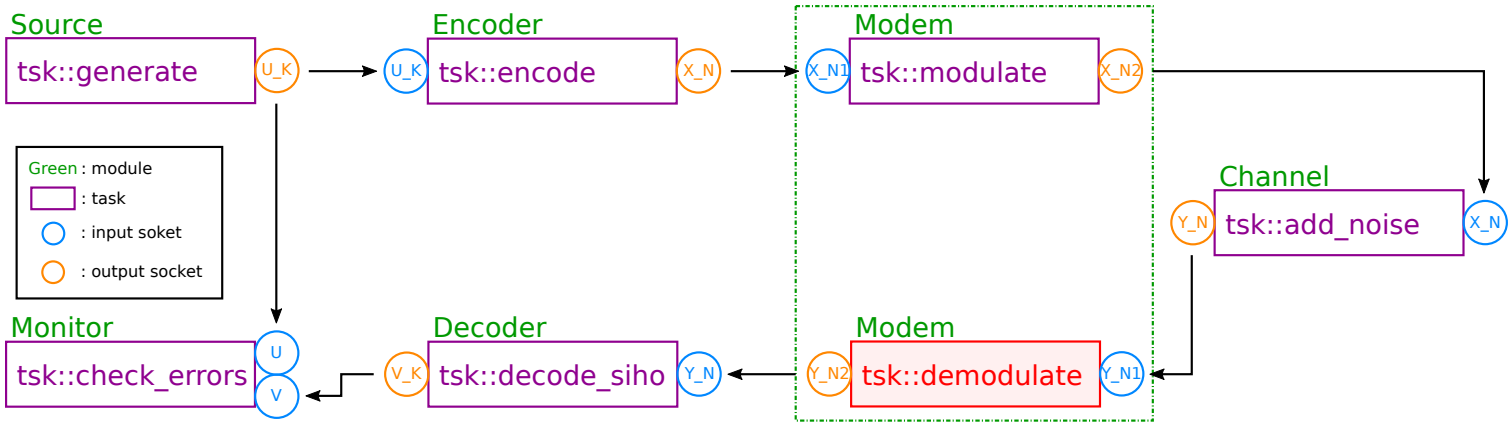

```
1  // the simulation loop
2  while (!monitor.fe_limit_achieved()) {
3    source [src::tsk::generate    ].exec();
4    encoder[enc::tsk::encode      ].exec();
5    modem  [mdm::tsk::modulate    ].exec();
6
7
8
9
10 }
```

My Project with AFF3CT: Tasks Execution



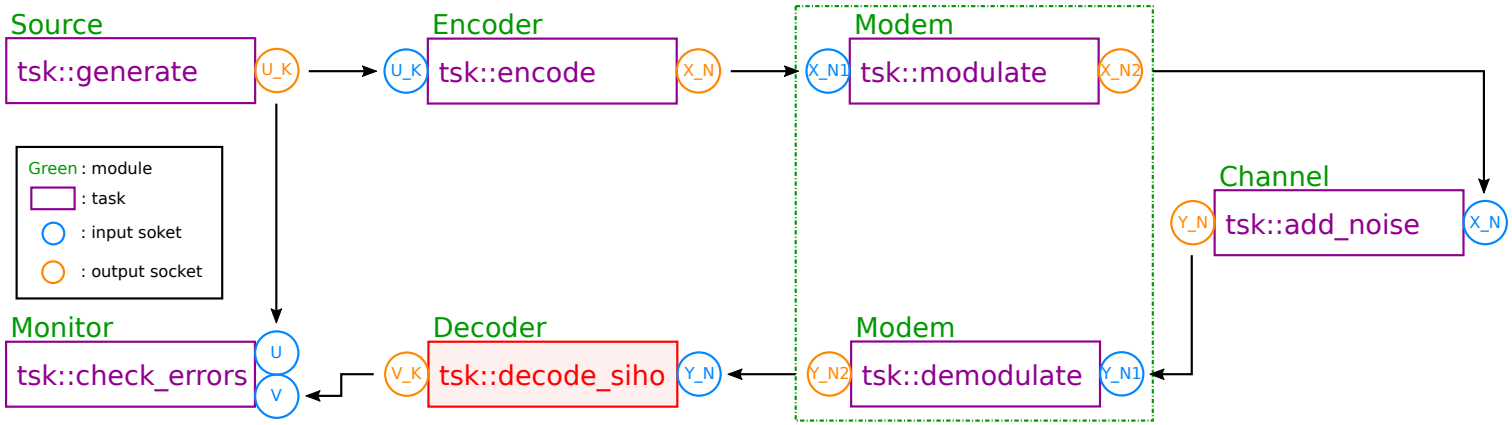

```
1  // the simulation loop
2  while (!monitor.fe_limit_achieved()) {
3    source [src::tsk::generate    ].exec();
4    encoder[enc::tsk::encode      ].exec();
5    modem  [mdm::tsk::modulate    ].exec();
6    channel[chn::tsk::add_noise   ].exec();
7
8
9
10 }
```

My Project with AFF3CT: Tasks Execution



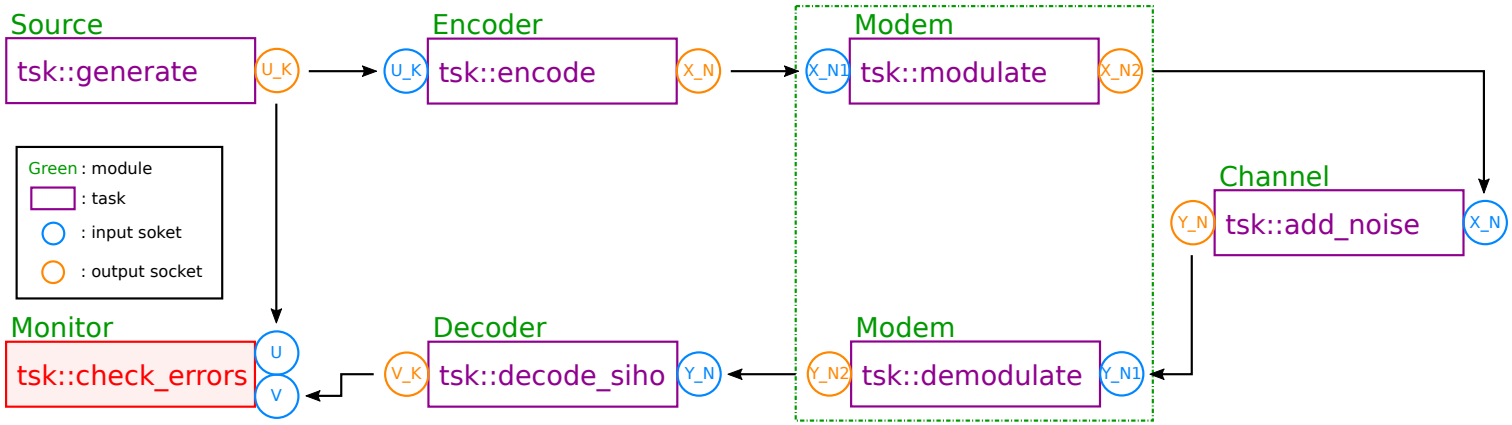

```
1  // the simulation loop
2  while (!monitor.fe_limit_achieved()) {
3    source [src::tsk::generate    ].exec();
4    encoder[enc::tsk::encode      ].exec();
5    modem  [mdm::tsk::modulate    ].exec();
6    channel[chn::tsk::add_noise   ].exec();
7    modem  [mdm::tsk::demodulate  ].exec();
8
9
10 }
```

My Project with AFF3CT: Tasks Execution



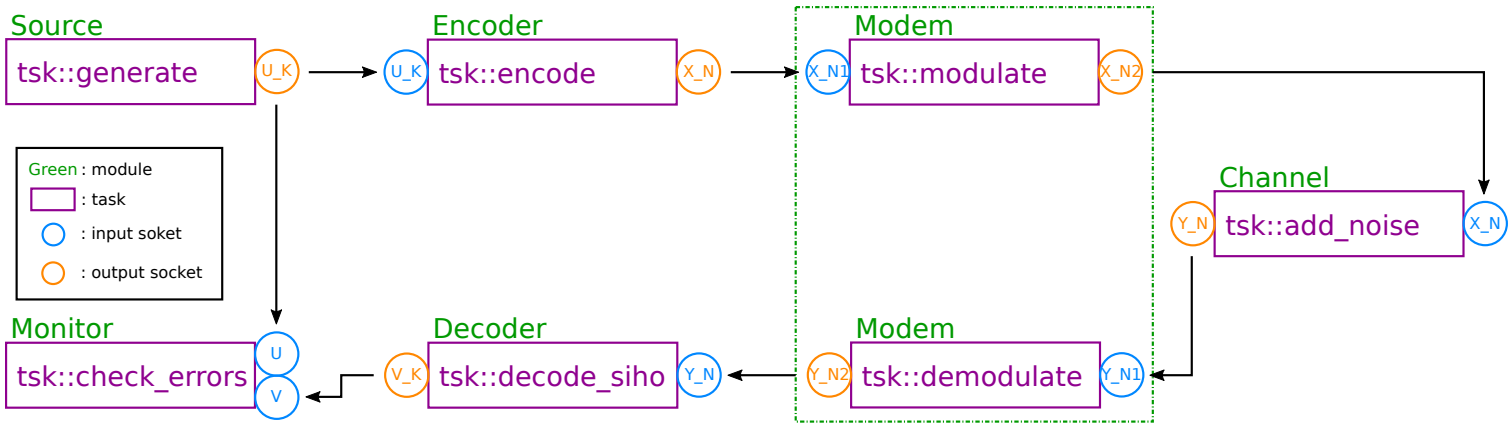

```
1  // the simulation loop
2  while (!monitor.fe_limit_achieved()) {
3    source [src::tsk::generate    ].exec();
4    encoder[enc::tsk::encode      ].exec();
5    modem  [mdm::tsk::modulate    ].exec();
6    channel[chn::tsk::add_noise   ].exec();
7    modem  [mdm::tsk::demodulate  ].exec();
8    decoder[dec::tsk::decode_siho ].exec();
9
10 }
```

My Project with AFF3CT: Tasks Execution




```
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    modem  [mdm::tsk::demodulate  ].exec();
8    decoder[dec::tsk::decode_siho ].exec();
9    monitor[mnt::tsk::check_errors].exec();
10 }
```

My Project with AFF3CT: Tasks Execution



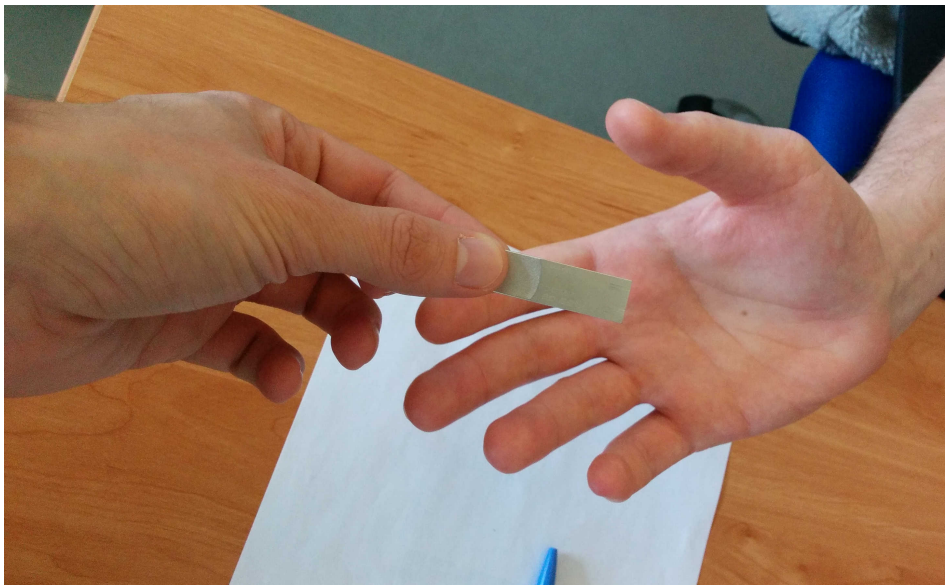

```
1  // the simulation loop
2  while (!monitor.fe_limit_achieved()) {
3    source [src::tsk::generate    ].exec();
4    encoder[enc::tsk::encode      ].exec();
5    modem  [mdm::tsk::modulate    ].exec();
6    channel[chn::tsk::add_noise   ].exec();
7    modem  [mdm::tsk::demodulate  ].exec();
8    decoder[dec::tsk::decode_siho ].exec();
9    monitor[mnt::tsk::check_errors].exec();
10 }
```

Plan

Source Code Management

Motivation

Sharing source code between team members can be very complicated without the appropriate tools.

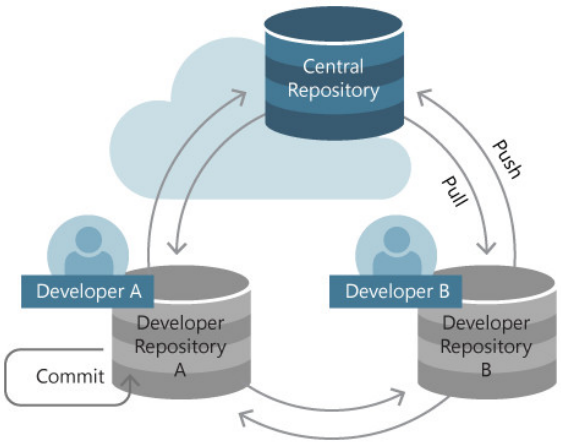


Source Code Management: **Git**

- **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`)



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

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**

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`

Source Code Management: **Add New Feature**

- Create branches from `master` or `development`

```
$ git checkout development
```

Source Code Management: **Add New Feature**

- Create branches from `master` or `development`

```
$ git checkout development
$ git checkout -b 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
```

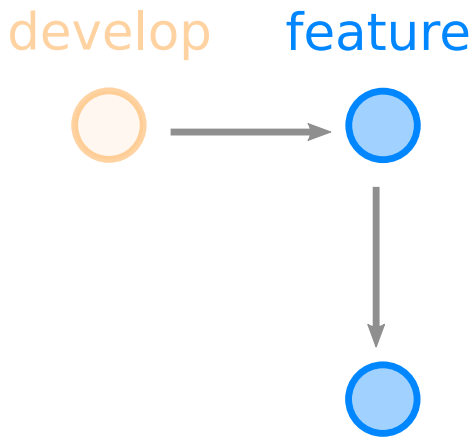
Source Code Management: **Add New Feature**

- 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."
```



Source Code Management: **Add New Feature**

- Create branches from `master` or `development`

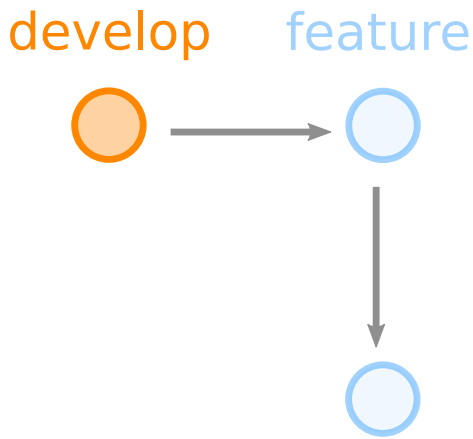
```
$ git checkout development
$ git checkout -b new_feature_branch
```

- 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`

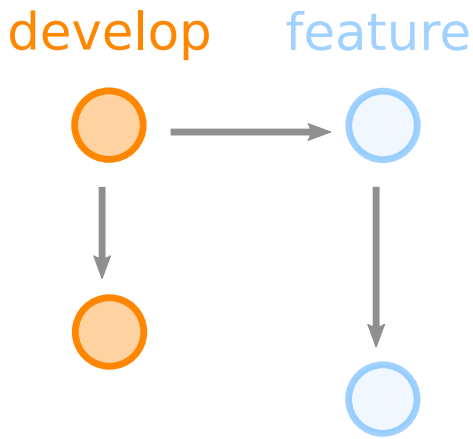
```
$ git checkout development
$ git checkout -b new_feature_branch
```

- 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`

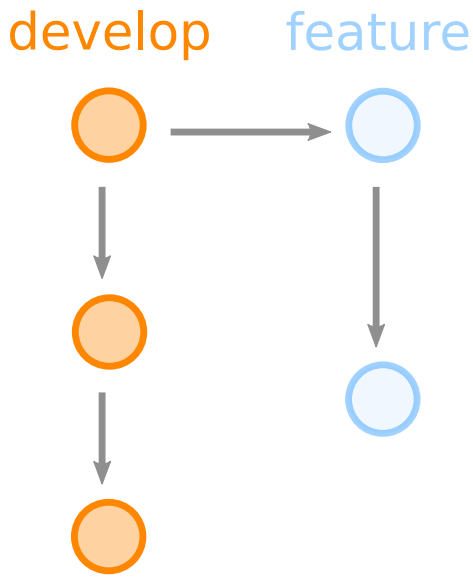
```
$ git checkout development
$ git checkout -b new_feature_branch
```

- 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
```



Source Code Management: **Add New Feature**

- Create branches from `master` or `development`

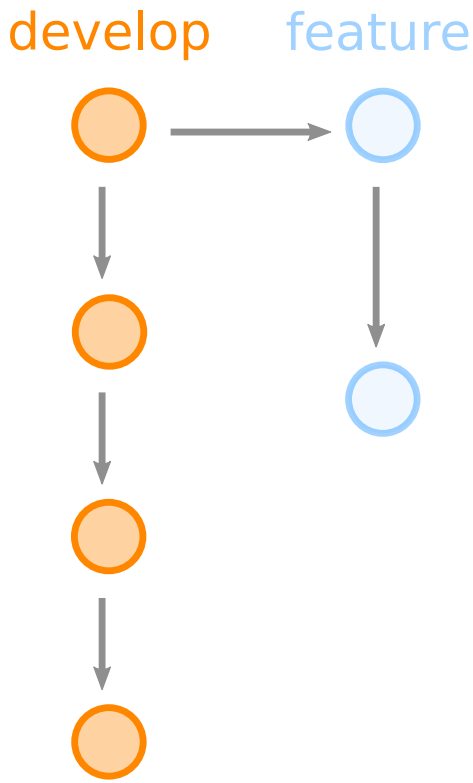
```
$ git checkout development
$ git checkout -b new_feature_branch
```

- 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

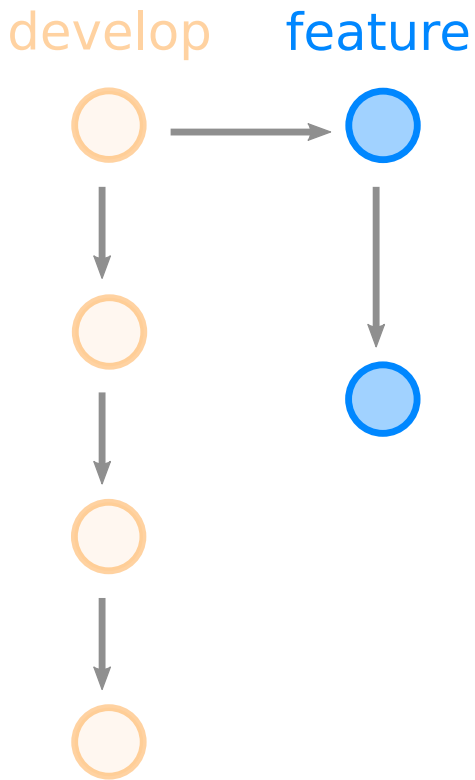
```
$ git add touched_files
$ git commit -m "Nice message."
```

- Update the `master` or `development` branch

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

- Merge `development` in the `feature` branch

```
$ git checkout new_feature_branch
```



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

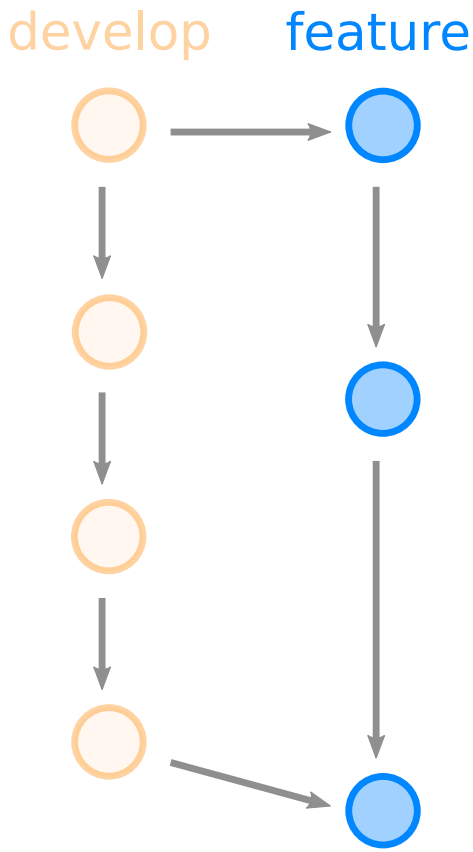
```
$ git add touched_files
$ git commit -m "Nice message."
```

- 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

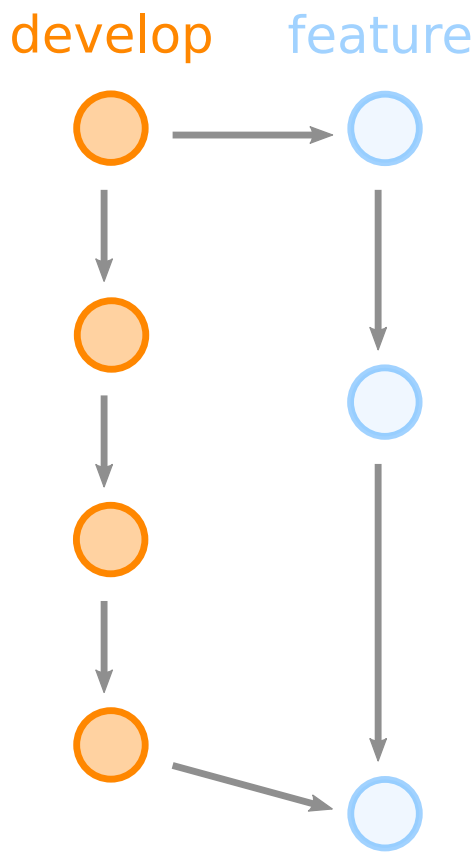
```
$ git checkout development
$ git pull origin development
```

- Merge `development` in the `feature` branch

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

- 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

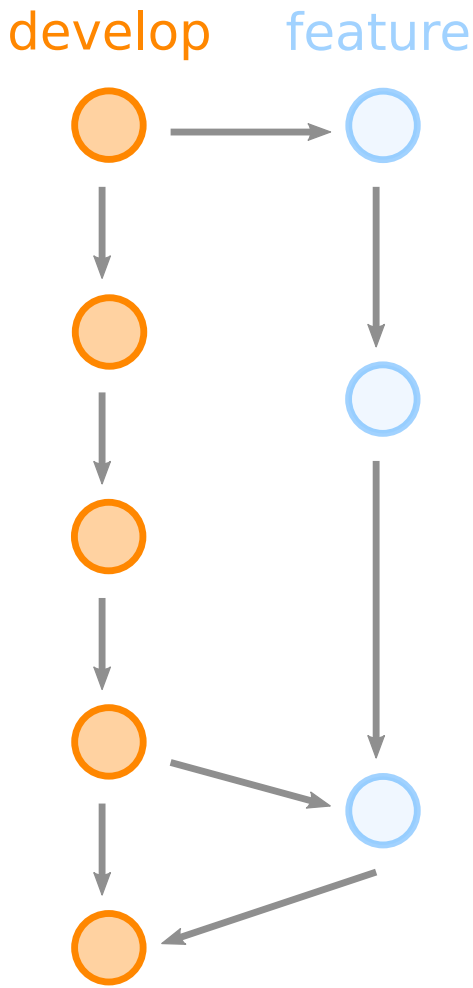
```
$ git checkout development
$ git pull origin development
```

- 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
```



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

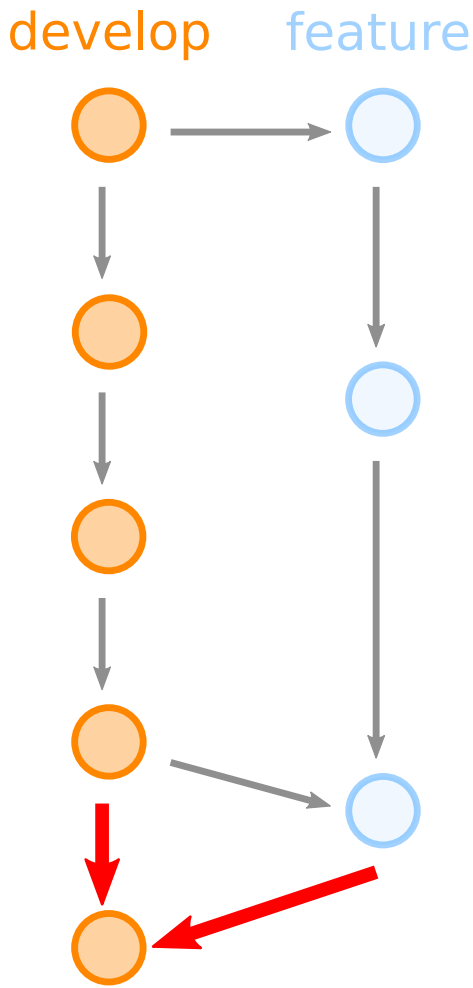
```
$ git checkout development
$ git pull origin development
```

- 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."
```

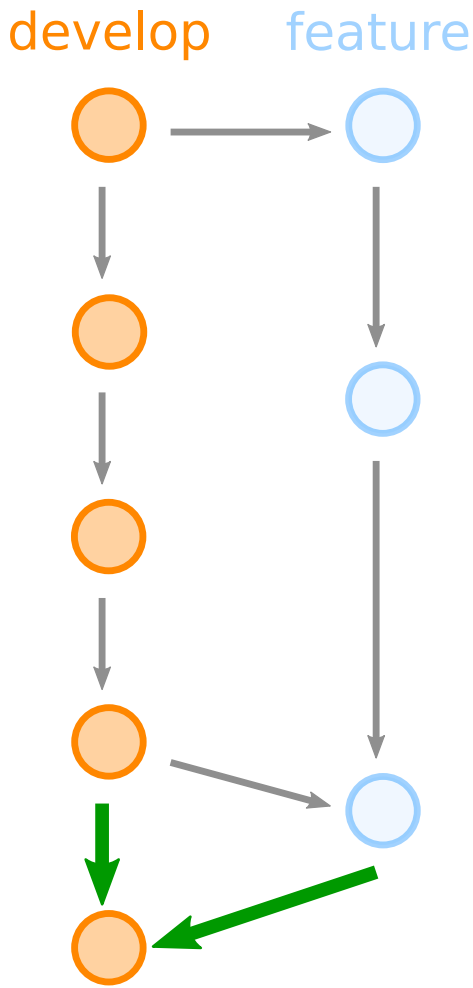
- Update the `master` or `development` branch

```
$ 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
```

- 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

Repositories

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

- `repository`: a specific Git repository

- `aff3ct`: the main **AFF3CT** repository (source code)

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)

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

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

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

Submodules

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

Continuous Integration

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 modifications.

Continuous Integration

Problem

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

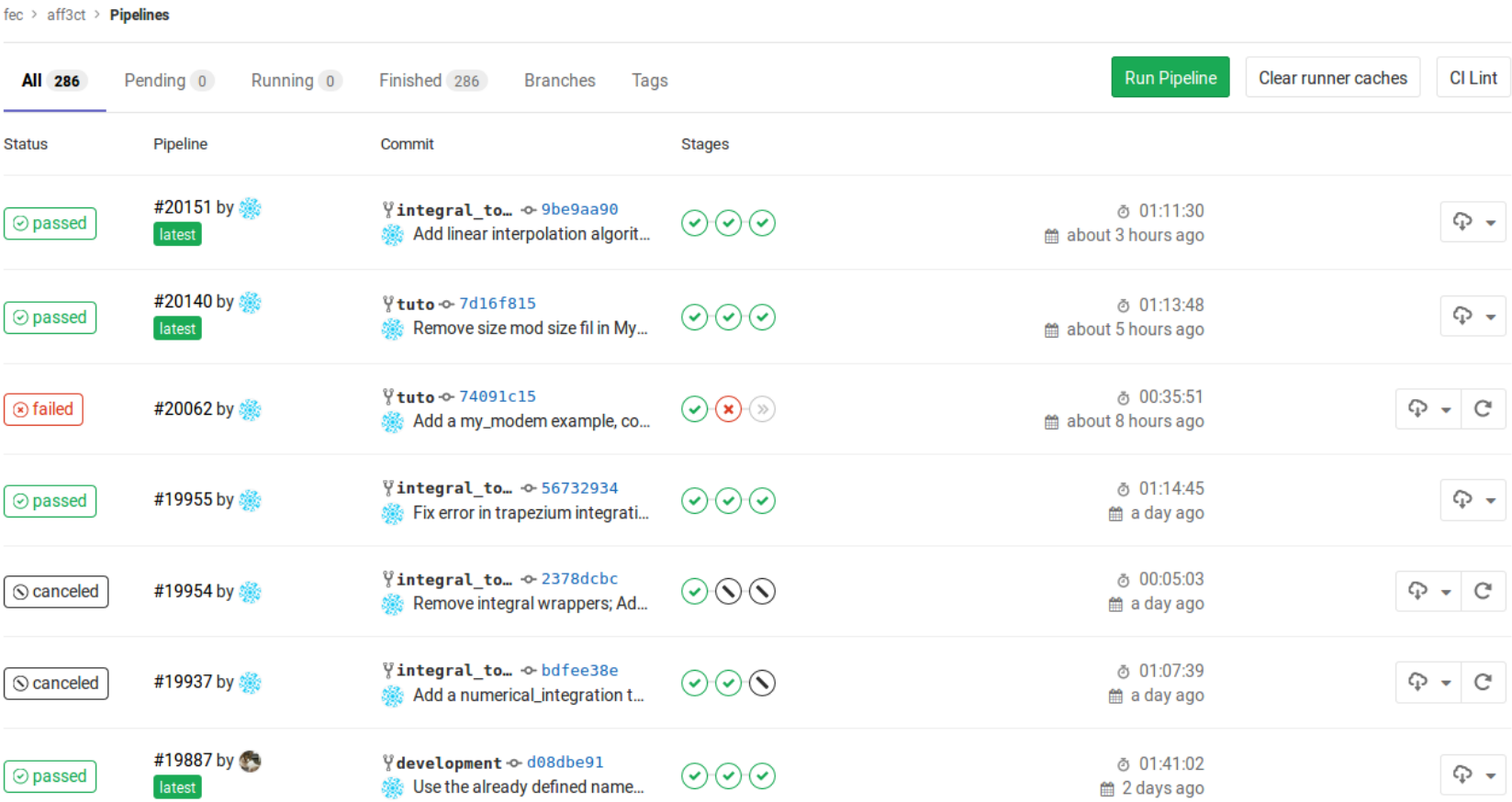
Proposal

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

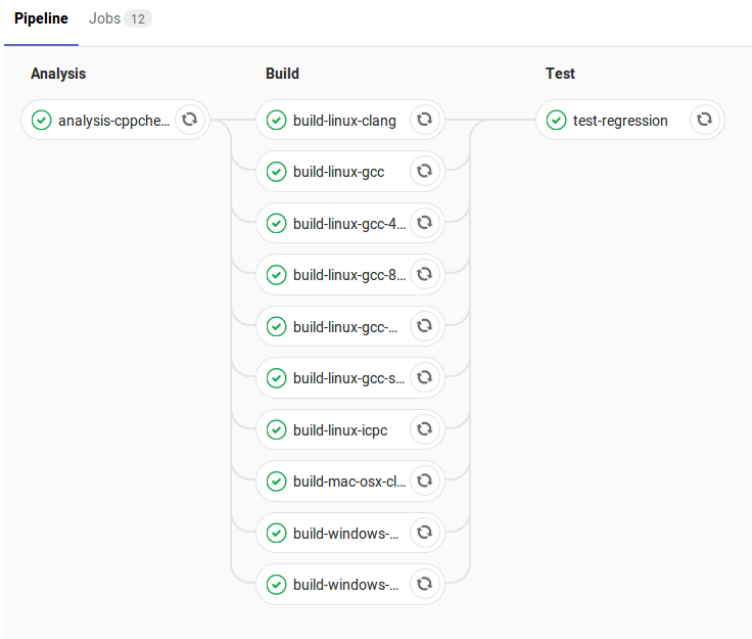
Continuous Integration: Pipelines

The tests are run for each branches (even on the `feature` branches)



Continuous Integration: Focus on a Specific Pipeline

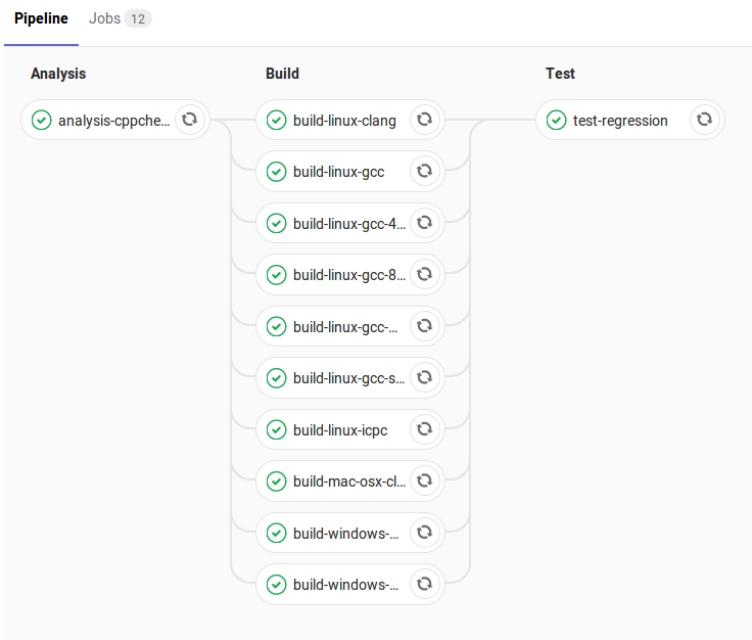
One series of tests is decomposed in **a 3-stage pipeline**



Continuous Integration: Focus on a Specific Pipeline

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

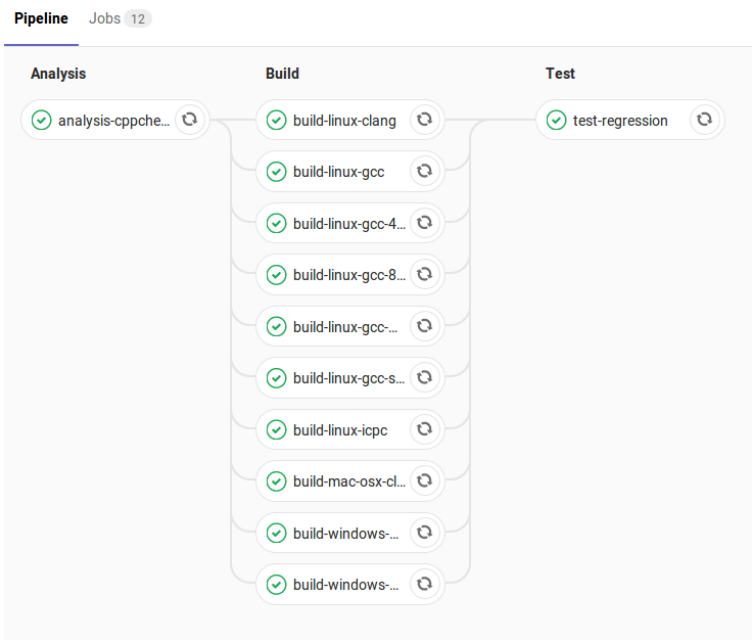
- **Analysis**: checks if the code syntax is valid (with `cppcheck`)



Continuous Integration: Focus on a Specific Pipeline

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

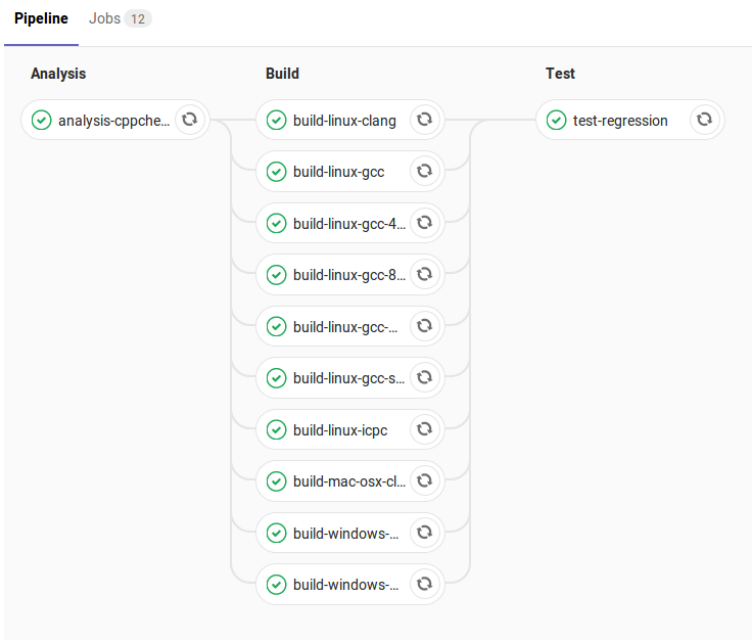
- **Analysis**: checks if the code syntax is valid (with `cppcheck`)
- **Build**: compiles the code on various systems and on various targets



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



Plan

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?**