MODELING THE AGC LOOP FOR DAB RADIO

PUBLIC

Karl Sturm

NOVEMBER 2022

SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2022 NXP B.V.

AGENDA

- System Modeling with SystemC / -AMS
  - Use Case
  - System View
  - Subsystem View
- System Integration with IP-XACT
- System Simulation and Results

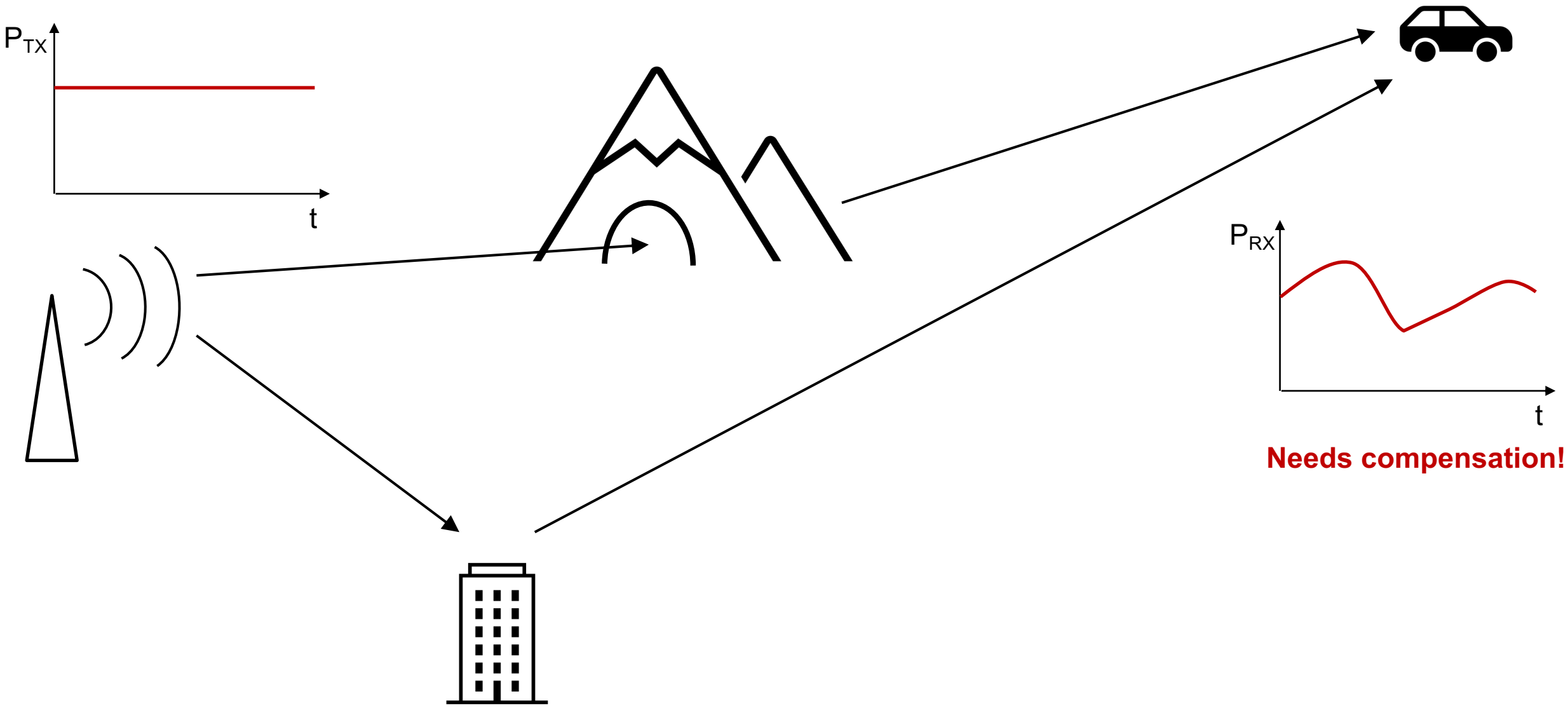
System Modeling with SystemC / -AMS

SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2022 NXP B.V.

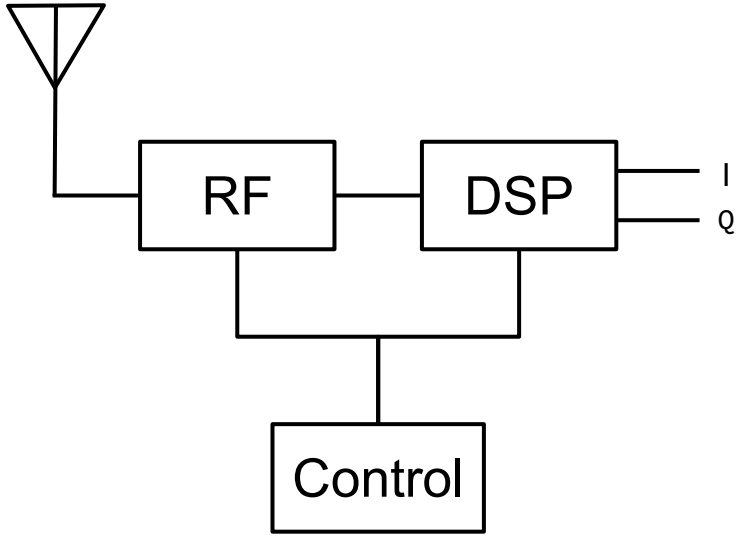
USE CASE



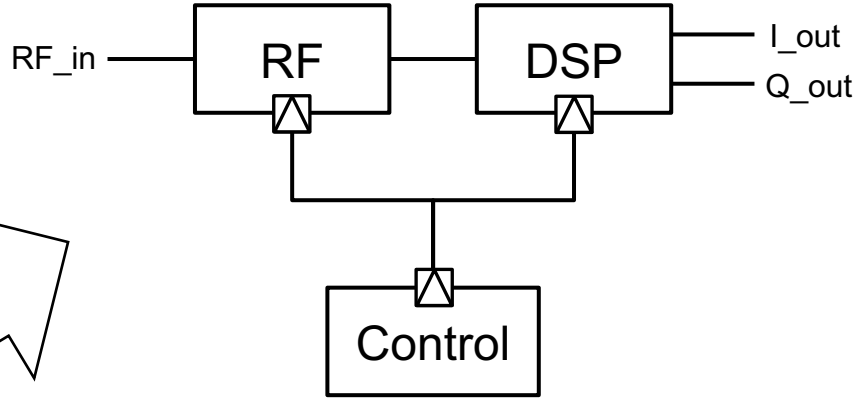
1) Requirements

- Receive RF Signal
- Convert to Baseband
- Send Data Off-Chip

2) Map Functions to Subsystems



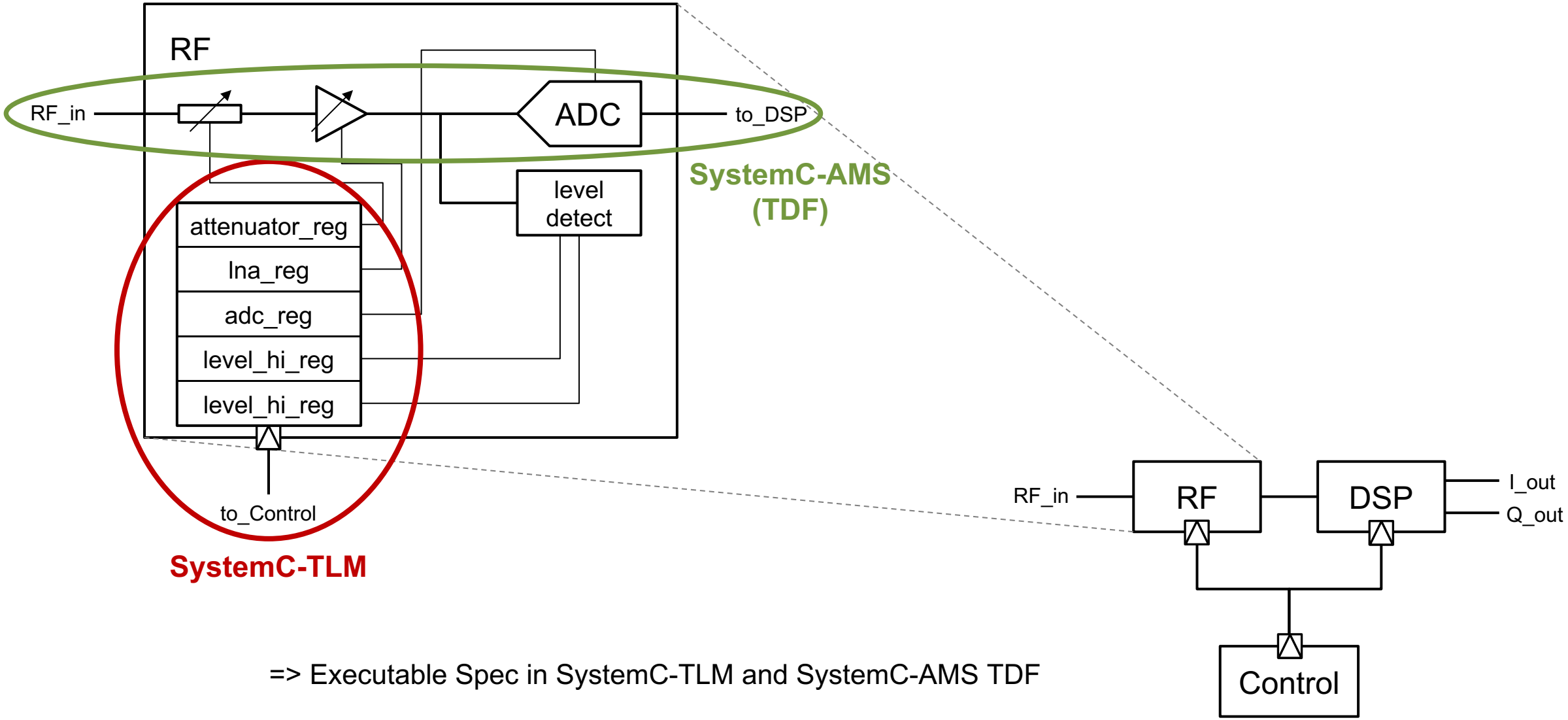
3) System Model



- Realistic Stimuli for Model V&V
- Block Interactions via Bus Messages
- Block Interface Standardization (Registers)

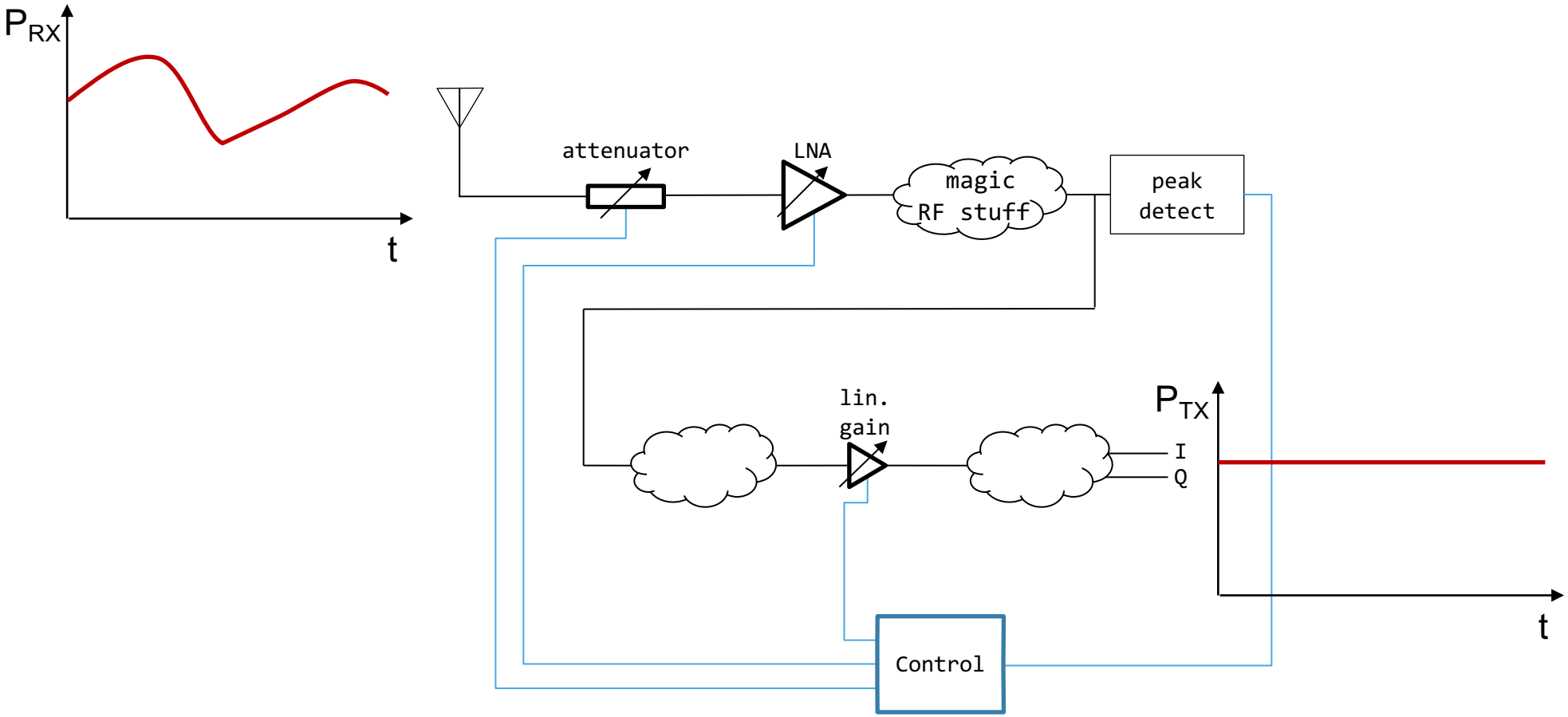
=> Executable Spec in SystemC-TLM

SUBSYSTEM VIEW

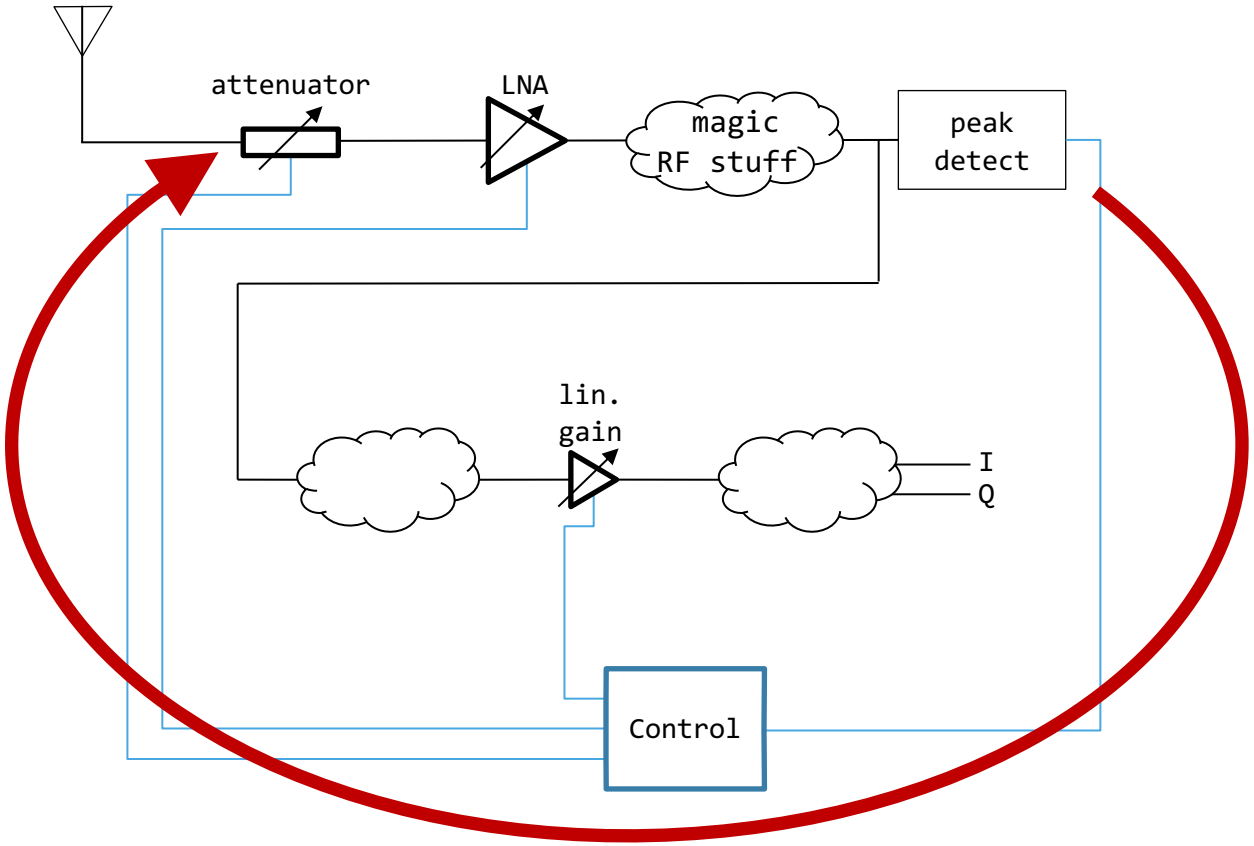


=> Executable Spec in SystemC-TLM and SystemC-AMS TDF

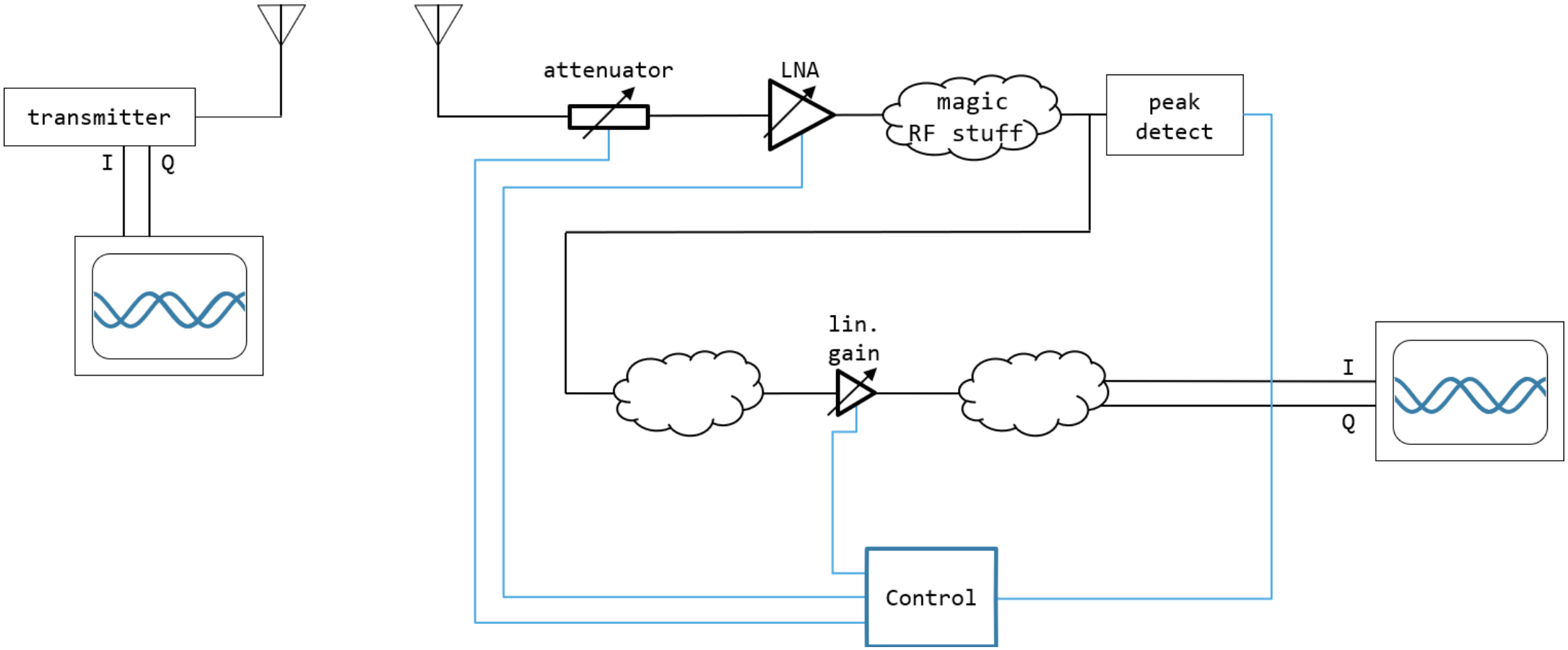
RF AGC LOOP


P_{RX}
t
attenuator
LNA
magic
RF stuff
peak
detect
lin.
gain
P_{TX}
I
Q
t
Control

RF AGC LOOP


attenuator
LNA
magic
RF stuff
peak
detect
lin.
gain
I
Q
Control

END-TO-END MODEL AND TESTBENCH



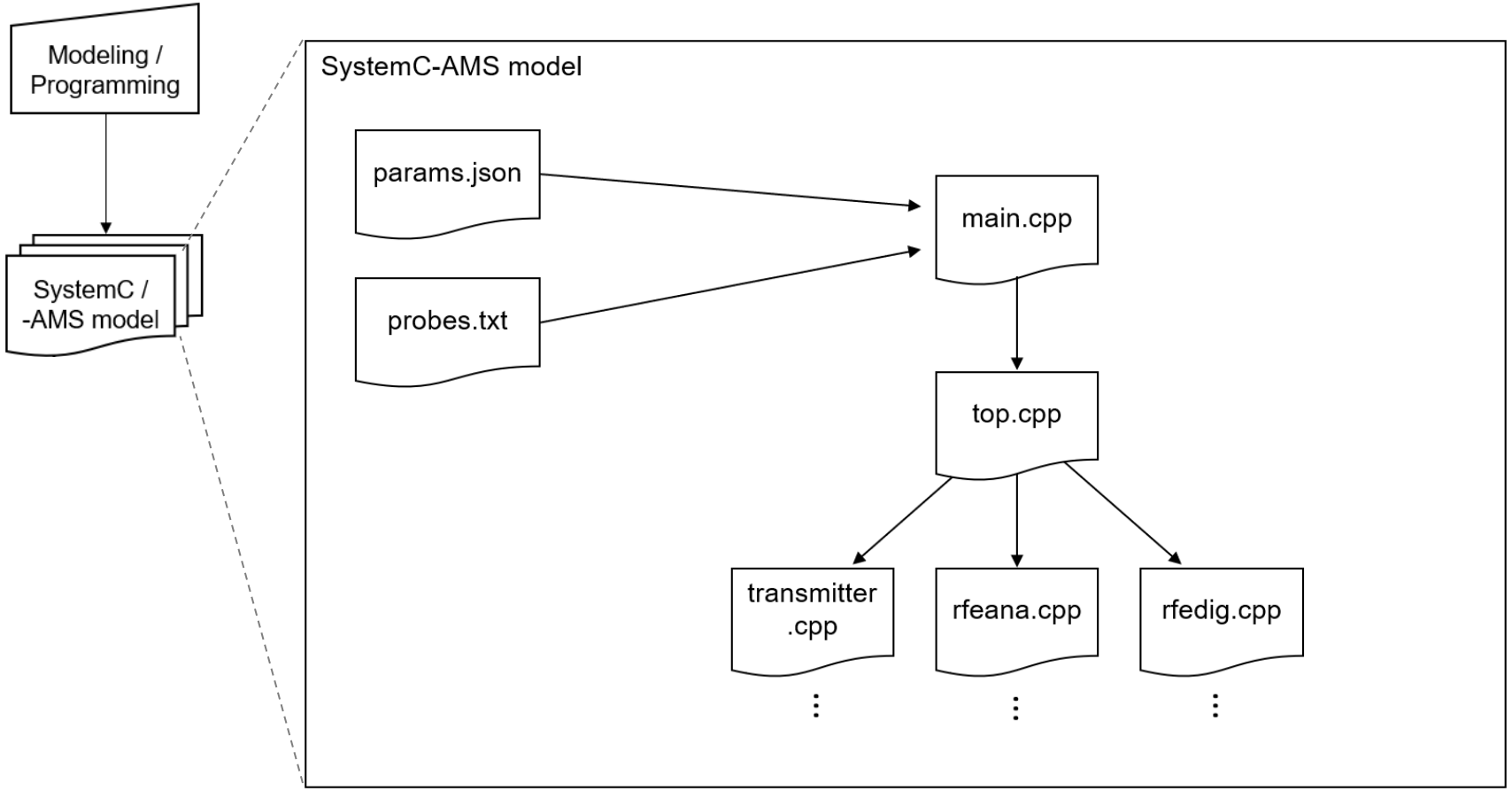
System Integration with IP-XACT

SECURE CONNECTIONS
FOR A SMARTER WORLD

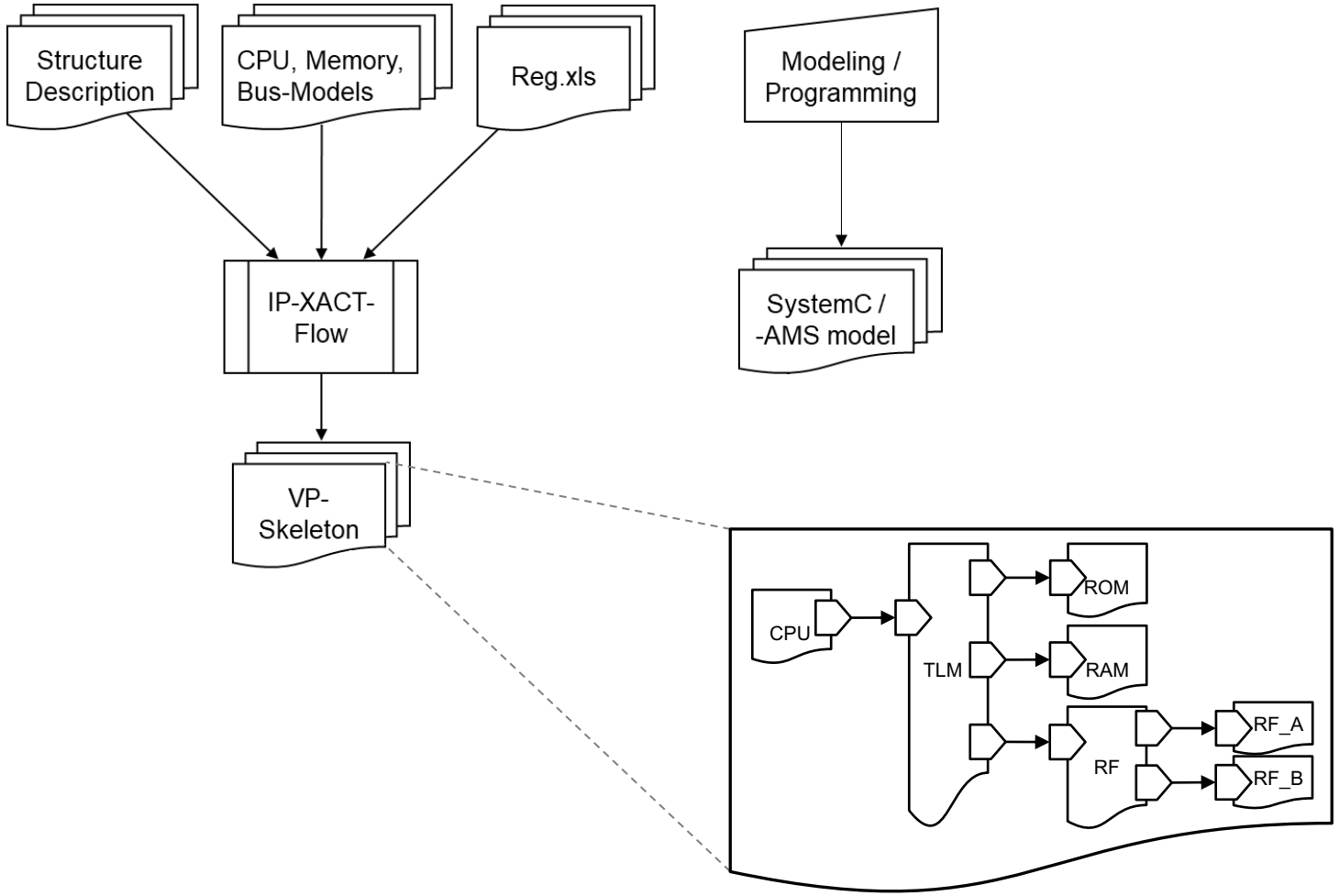
PUBLIC

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2022 NXP B.V.

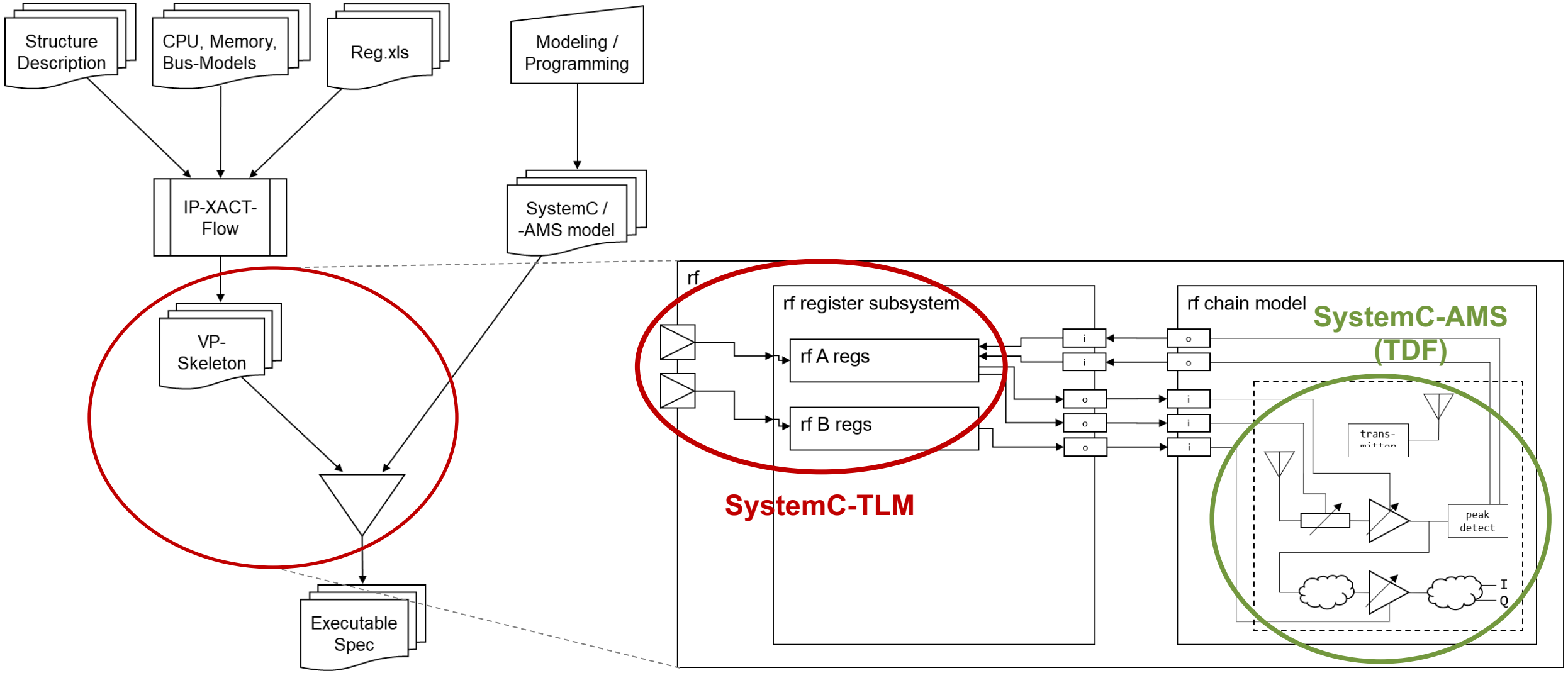
SYSTEM INTEGRATION



SYSTEM INTEGRATION



SYSTEM INTEGRATION



System Simulation and Results

SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2022 NXP B.V.

LIBRARIES AND TECHNOLOGIES USED

- SystemC 2.3.x Implementation from EDA Tool
- SystemC-AMS 2.4 Implementation from EDA Tool
- IP-XACT system integration flow (incl build- and register generation flow)
- EDA tooling for waveform analysis and debug
- In-house libraries developed for configuration, RF models and common utilities

SYSTEMC-AMS MODEL EXAMPLE

```
#include <systemc>
#include <systemc-ams>

template <class T>
class decimator_tdf : public sca_tdf::sca_module {

    public:
        sca_tdf::sca_in<T>  tdf_i;
        sca_tdf::sca_out<T> tdf_o;

        decimator_tdf(sc_core::sc_module_name, unsigned int fir_len,
                                               const T *coeff_,
                                               unsigned int dec_factor_,
                                               unsigned int out_scale_);

        ~decimator_tdf();
        void processing();
        void initialize();
        void set_attributes();

    private:
        T *mem;
        T *coeff;
        unsigned int fir_len;
        unsigned int dec_factor;
        unsigned int out_scale;
};
```

```
#include "decimator.h"

#include <cassert>
#include <cstring>

template <class T>
decimator_tdf<T>::decimator_tdf(sc_core::sc_module_name,
                                unsigned int filter_len_,
                                const T *coeff_,
                                unsigned int dec_factor_,
                                unsigned int out_scale_)
    : tdf_i("tdf_i"),
      tdf_o("tdf_o"),
      fir_len(filter_len_),
      dec_factor(dec_factor_),
      out_scale(out_scale_)
{
    assert(filter_len_ >= dec_factor_);
    mem = new T[fir_len];
    coeff = new T[fir_len];
    memset(mem, 0, sizeof(T)*fir_len);
    memcpy(coeff, coeff_, sizeof(T)*fir_len);
}

template <class T>
decimator_tdf<T>::~decimator_tdf() {
    delete[] mem;
    delete[] coeff;
}

template <class T>
void decimator_tdf<T>::processing() {
    memmove(&(mem[dec_factor]), &(mem[0]), sizeof(T)*(fir_len-1));

    unsigned int i;
    for ( i = 0; i < dec_factor; i++ ) {
        mem[dec_factor-(i+1)] = tdf_i.read(i);
    }

    T output = (T)0;
    for( i = 0; i < fir_len; i++ ) {
        output += coeff[i] * mem[i];
    }

    tdf_o.write(output/out_scale);
}

template <class T>
void decimator_tdf<T>::initialize() { }

template <class T>
void decimator_tdf<T>::set_attributes() {
    tdf_i.set_rate(dec_factor);
    tdf_o.set_rate(1);
}

template class decimator_tdf<double>;
```

Decimator implementation in SystemC-AMS TDF

AGC LOOP CONTROL FIRMWARE EXAMPLE

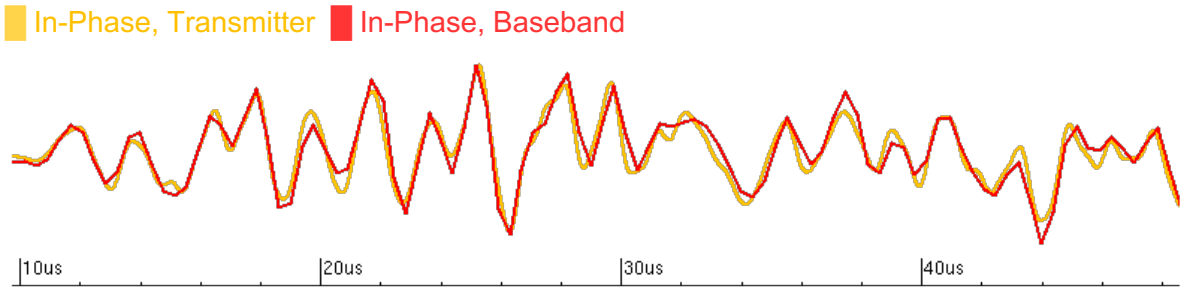
```c
#include "timer.h"
#include "rf_A_regs.h"
#include "rf_B_regs.h"

rf_A_reg_t *rf_A_reg =
    ((rf_A_reg_t *)RF_A_BASEADDRESS);

rf_B_reg_t *rf_B_reg =
    ((rf_B_reg_t *)RF_B_BASEADDRESS);

const double gain_steps[17][2] = { {   0,   0 },
                                   {   0,   1 },
                                   {   0,   3 },
                                   {   0,   7 },
                                   {   0,  15 },
                                   {   0,  31 },
                                   {   0,  63 },
                                   {   0, 127 },
                                   {   0, 255 },
                                   {   1, 255 },
                                   {   3, 255 },
                                   {   7, 255 },
                                   {  15, 255 },
                                   {  31, 255 },
                                   {  63, 255 },
                                   { 127, 255 },
                                   { 255, 255 } };

void set_gain(int i) {
    rf_A_reg->attenuator = gain_steps[i][0];
    rf_A_reg->lna        = gain_steps[i][1];
}

int main(void) {

  unsigned int gain_thres = 42;

  int gain_step = 0;
  unsigned int lo_level;
  unsigned int hi_level;

  for (;;) {
      sleep(49,"us");

      lo_level = rf_B_reg->lo_level;
      hi_level = rf_B_reg->hi_level;

      if (hi_level > lo_level + gain_thres) {
          gain_step++;
          if (gain_step > 16) gain_step = 16;
      }

      if (lo_level > hi_level + gain_thres) {
          gain_step--;
          if (gain_step < 0) gain_step = 0;
      }

      set_gain(gain_step);

      rf_B_reg->reset = 1;
      rf_B_reg->reset = 0;
  }

  return 0;
}
```

AGC LOOP CONTROL FIRMWARE EXAMPLE

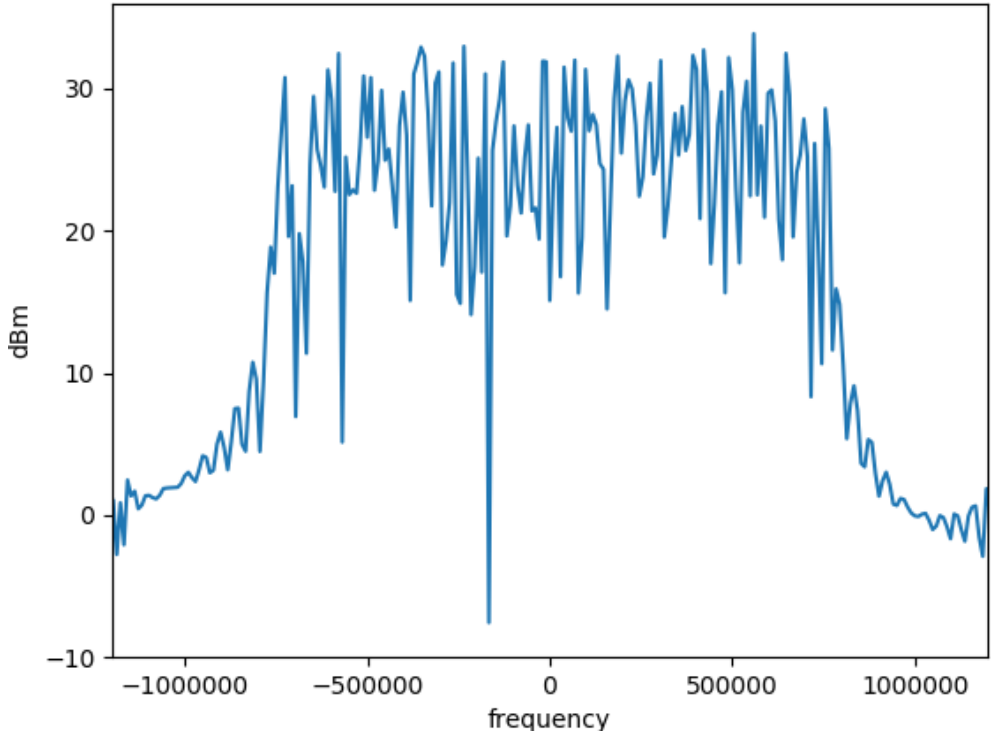
```c
lo_level = rf_B_reg->lo_level;
hi_level = rf_B_reg->hi_level;

if (hi_level > lo_level + gain_thres) {
    gain_step++;
    if (gain_step > 16) gain_step = 16;
}

if (lo_level > hi_level + gain_thres) {
    gain_step--;
    if (gain_step < 0) gain_step = 0;
}
```

```c
1 #include "timer.h"
2 #include "rf_A_regs.h"
3 #include "rf_B_regs.h"
22                                  {   7, 255 },
23                                  {  15, 255 },
24                                  {  31, 255 },
25                                  {  63, 255 },
26                                  { 127, 255 },
27                                  { 255, 255 } };
28
29 void set_gain(int i) {
30     rf_A_reg->attenuator = gain_steps[i][0];
31     rf_A_reg->lna        = gain_steps[i][1];
32 }
33
34 int main(void) {
35
36   unsigned int gain_thres = 42;
37
38   int gain_step = 0;
39   unsigned int lo_level;
40   unsigned int hi_level;
41
42   for (;;) {
43       sleep(49,"us");
44
45       lo_level = rf_B_reg->lo_level;
46       hi_level = rf_B_reg->hi_level;
47
48       if (hi_level > lo_level + gain_thres) {
49           gain_step++;
50           if (gain_step > 16) gain_step = 16;
51       }
52
53       if (lo_level > hi_level + gain_thres) {
54           gain_step--;
55           if (gain_step < 0) gain_step = 0;
56       }
57
58       set_gain(gain_step);
59
60       rf_B_reg->reset = 1;
61       rf_B_reg->reset = 0;
62   }
63
64   return 0;
65 }
66
```

DAB TRANSMISSION SANITY CHECK: “UNITY GAIN” (NO AGC)

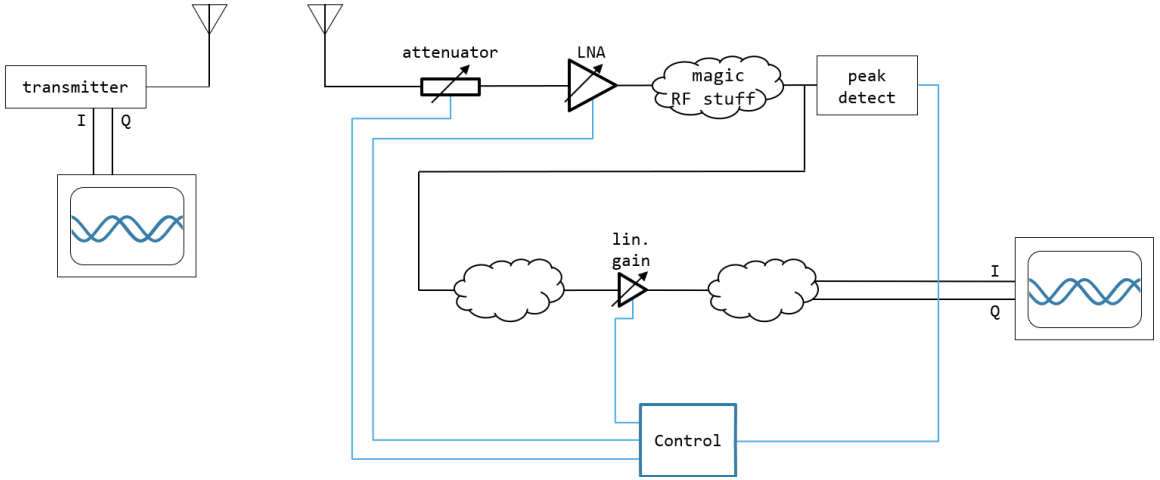


Time domain signal

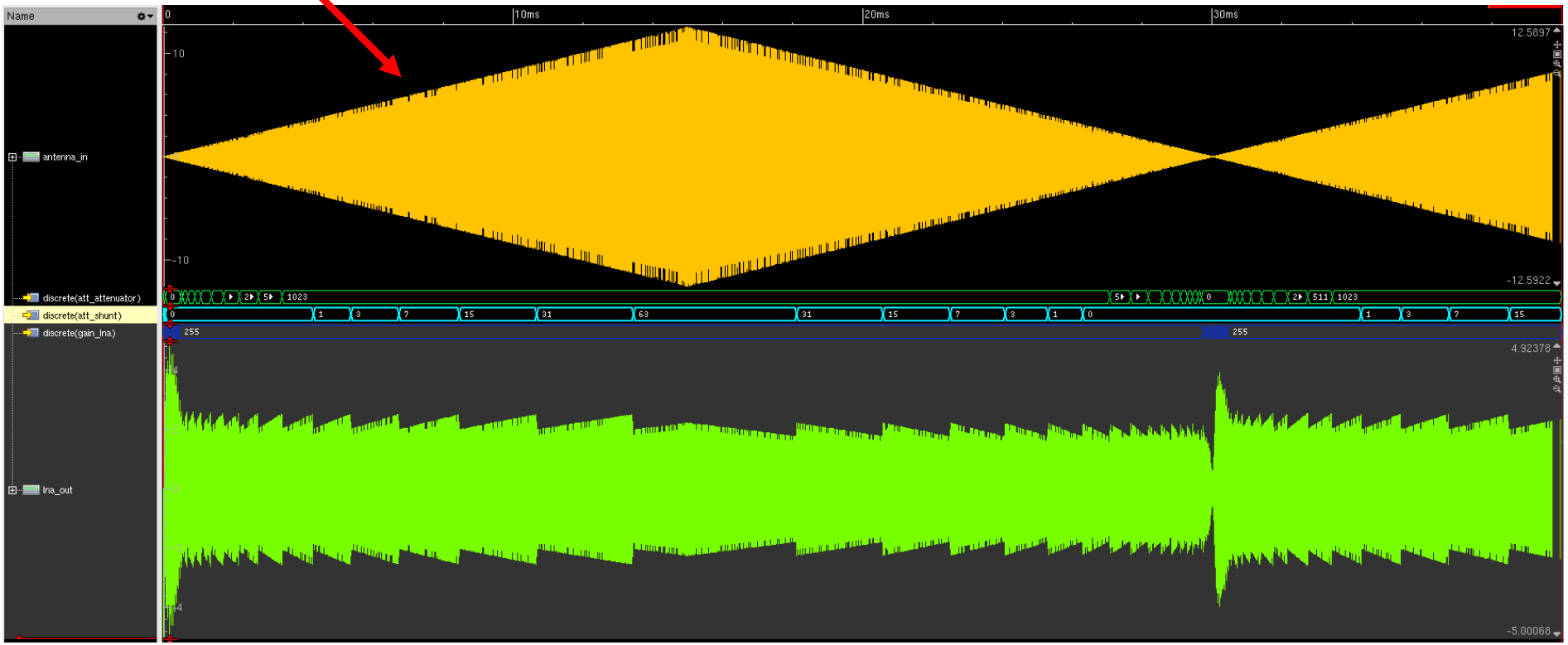


Frequency domain signal

SYSTEM SANITY CHECK: AGC LOOP CONTROL



15 ms ramp
(full dynamic range)



RESULTS

- Modeled AGC loop in an RF-frontend
- System-level interaction: µC ⇔ RF-frontend
- PoC for early driver / FW development
- Simulation Speed:
  - Discrete Event-based: ~2 hrs processing for 1 sec radio stream (passband signal representation)
  - TDF-based: ~10 min processing for 1 sec radio stream (passband signal representation)

SECURE CONNECTIONS
FOR A SMARTER WORLD

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V. ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2022 NXP B.V.