Using Triggered SystemC AMS AC Analysis for Run-Time Parameter Extraction

Peter Alfred Frießnegger
16.11.2022

Agenda

1	Motivation	3
2	ACP - Setup	7
3	ACP - Postprocessing	10
4	ACP - Trigger Generation	14
5	ACP - Results	17
6	ACP - Outlook	22

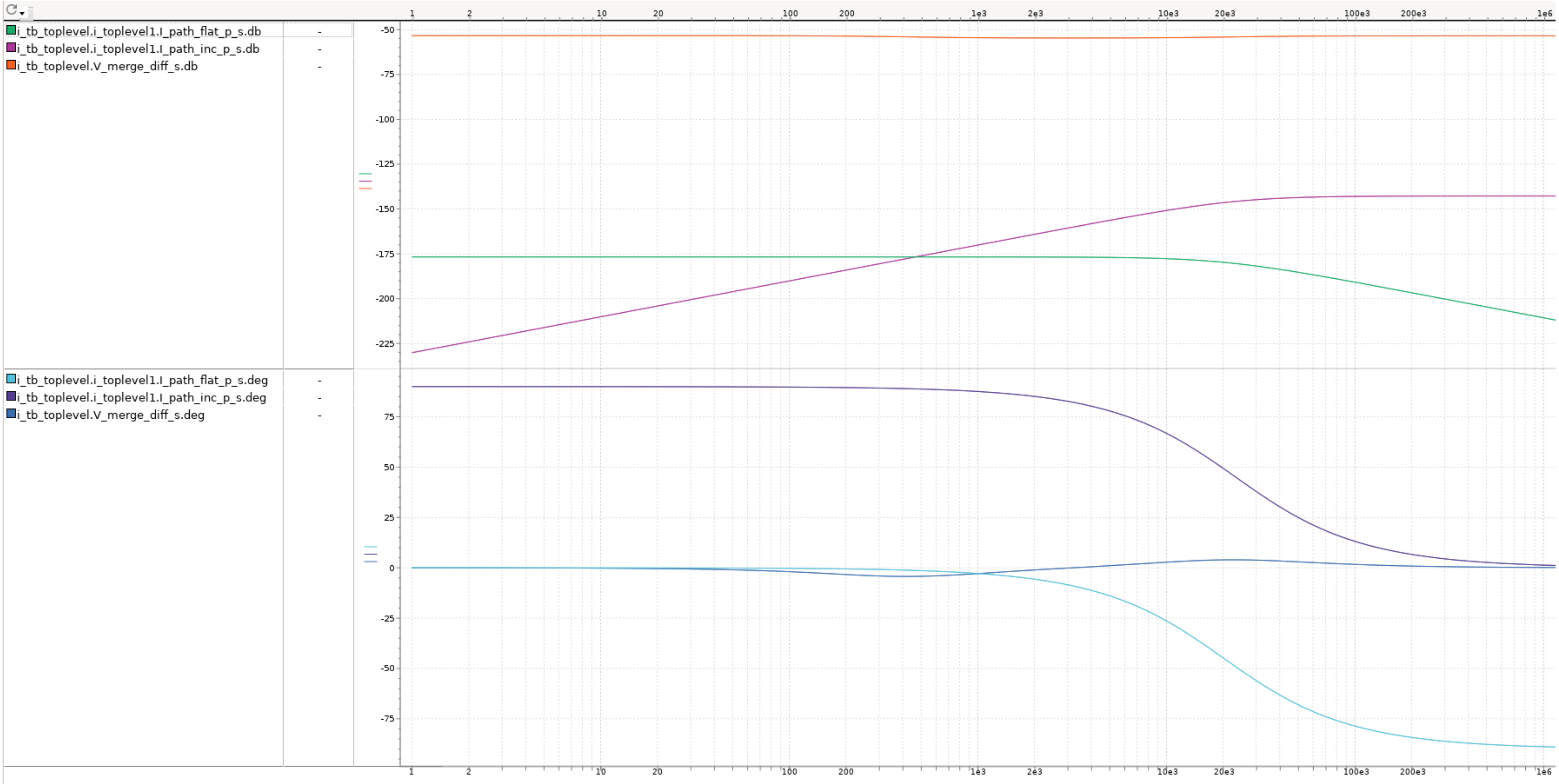
Agenda

1	Motivation	3
2	ACP - Setup	7
3	ACP - Postprocessing	10
4	ACP - Trigger Generation	14
5	ACP - Results	17
6	ACP - Outlook	22

Motivation

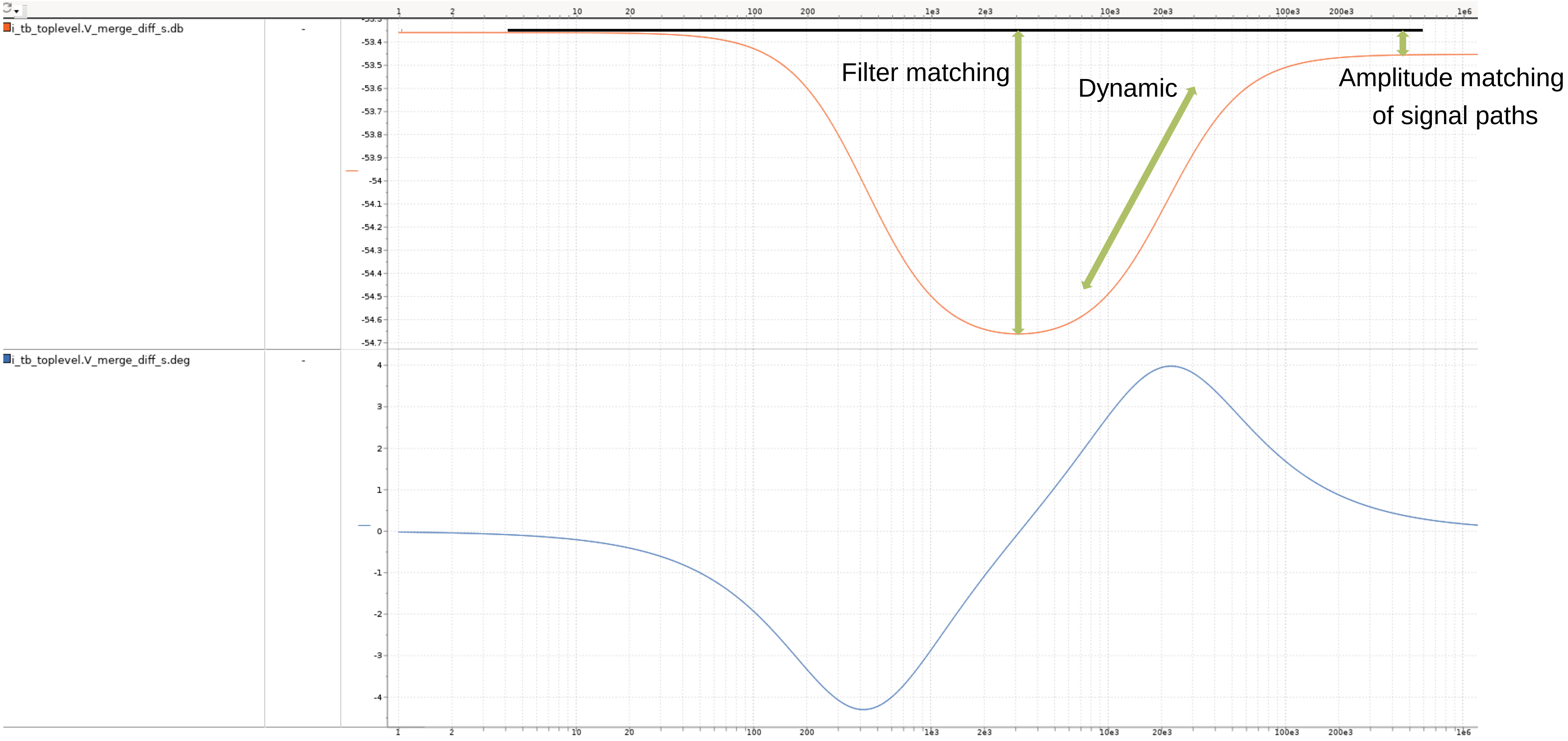
- 2 signal paths with different frequency behavior merged together
- A lot of filter poles/zeros need to match to achieve certain flatness behavior in the frequency domain
- Both – environmental conditions as well as internal states – can lead to significant changes
- Time domain analysis might not show issues present only in a certain (small) bandwidth
- AC analysis without changing environment / triggering changes of internal states does not reflect neither typical nor worst case performance
- Running test cases and delta time based running AC analysis + analyze all of them → not runtime/space-efficient + additional effort to assign results to parameter space + risk of losing coverage

Motivation



i_tb_toplevel.i_toplevel1.I_path_flat_p_s.db -
i_tb_toplevel.i_toplevel1.I_path_inc_p_s.db -
i_tb_toplevel.V_merge_diff_s.db -
-50
-75
-100
-125
-150
-175
-200
-225
i_tb_toplevel.i_toplevel1.I_path_flat_p_s.deg -
i_tb_toplevel.i_toplevel1.I_path_inc_p_s.deg -
i_tb_toplevel.V_merge_diff_s.deg -
75
50
25
0
-25
-50
-75
1
2
10
20
100
200
1e3
2e3
10e3
20e3
100e3
200e3
1e6

Motivation



i_tb_toplevel.V_merge_diff_s.db
i_tb_toplevel.V_merge_diff_s.deg
Filter matching
Dynamic
Amplitude matching
of signal paths
1 2 10 20 100 200 1e3 2e3 10e3 20e3 100e3 200e3 1e6
-53.4 -53.5 -53.6 -53.7 -53.8 -53.9 -54 -54.1 -54.2 -54.3 -54.4 -54.5 -54.6 -54.7
4 3 2 1 0 -1 -2 -3 -4

Agenda

1	Motivation	3
2	ACP - Setup	7
3	ACP - Postprocessing	10
4	ACP - Trigger Generation	14
5	ACP - Results	17
6	ACP - Outlook	22

COS AC Postprocessor approach – Setup ACP

› Construct ACP

```
cos_ac_postprocessor acp;
```

› Add Signals of interest (here: into ‘set_attributes_cpp’ function of tdf module)

```
void ac_access_mod_tdf::set_attributes_cpp()
{
    s.acp.add_signal(tdf_i.name());  //add the inp - port signal to be traced
```

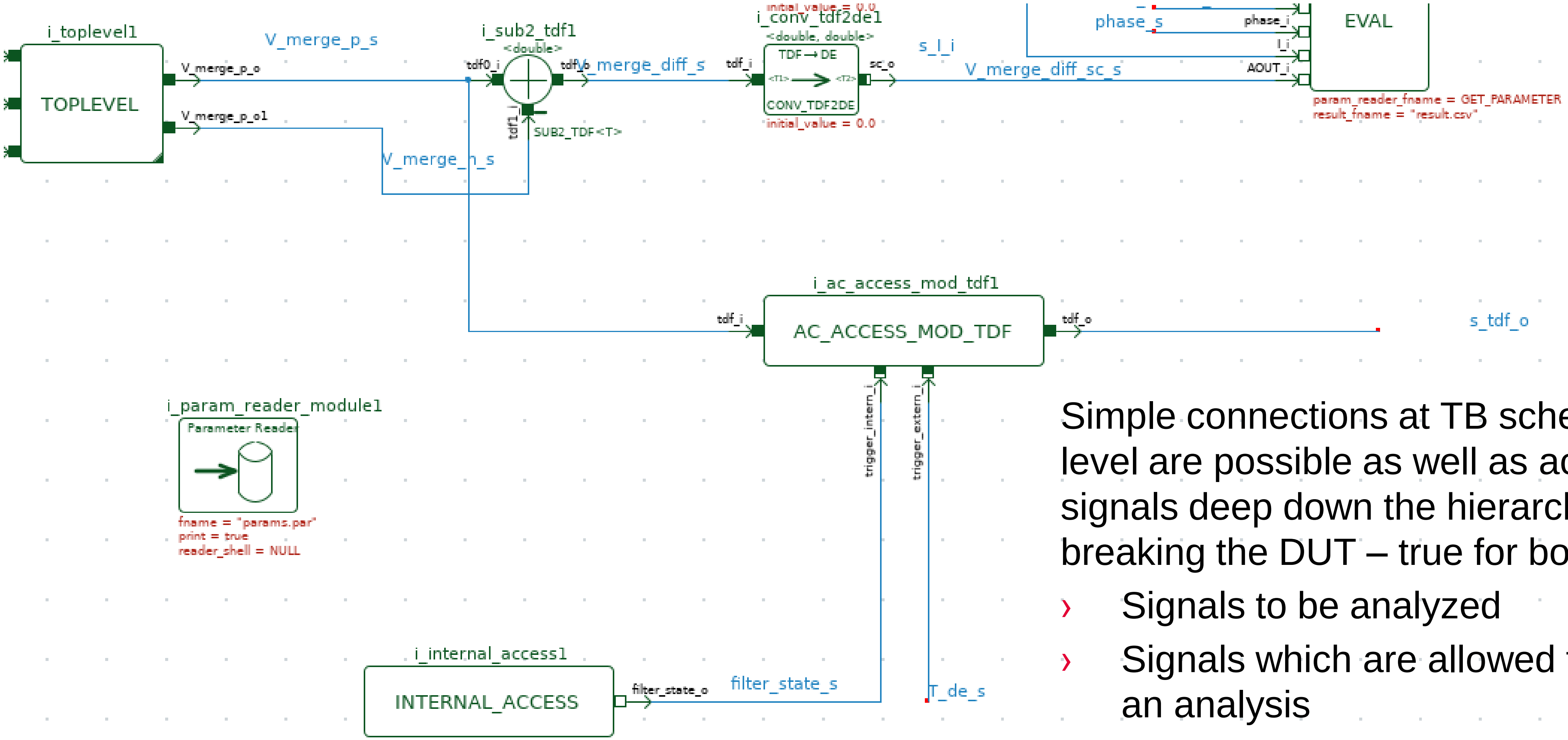
› Results of AC runs can be accessed via callback function (and via ‘&’ assigned to local variables – here: sca_complex vector)

```
s.acp.set_postprocessing_callback( //set the result callback function
        [&](double w, const std::vector<sca_util::sca_complex> &res) {

            for (auto val : res) {
                s.results.push_back(val);
            }
        });
```

COS AC Postprocessor approach – General Setup

› Gaining flexibility by using module / internal access approach



Simple connections at TB schematic level are possible as well as accessing signals deep down the hierarchy without breaking the DUT – true for both

› Signals to be analyzed

› Signals which are allowed to trigger an analysis

Agenda

1	Motivation	3
2	ACP - Setup	7
3	ACP - Postprocessing	10
4	ACP - Trigger Generation	14
5	ACP - Results	17
6	ACP - Outlook	22

COS AC Postprocessor approach – Further Postprocessing

- Complex values → Amplitude db / Phase deg

```
void ac_access_mod_tdf::calcAbsArgFromComplVec(std::vector<sca_util::sca_complex> compl_inp_vec, std::vector<double>* abs_db_vec, std::vector<double>* arg_vec)
{
    abs_db_vec->clear();
    arg_vec->clear();

    std::vector<sca_util::sca_complex>::iterator iter;
    for(iter = compl_inp_vec.begin(); iter < compl_inp_vec.end(); iter++)
    {
        abs_db_vec->push_back( 20 * log10(std::abs(*iter)) );
        arg_vec->push_back( std::arg(*iter) * 180.0 / 3.1415 );
    }
}
```

- Calculate output amplitude behavior vs. some defined target value (might es well be an range / change over f)

```
double ac_access_mod_tdf::calcMaxAbsDBDevTarget(std::vector<double> abs_db_vec, double target_db)
{
    double max_abs_diff_db  = 0;
    std::vector<double>::iterator iter;
    for(iter = abs_db_vec.begin(); iter < abs_db_vec.end(); iter++)
    {
        if( std::abs((*iter) - target_db) > std::abs(max_abs_diff_db) )
            max_abs_diff_db = ((*iter) - target_db);
    }

    return max_abs_diff_db;
}
```

COS AC Postprocessor approach – Further Postprocessing

- Calculate output phase behavior vs. some defined target value (might es well be an range / change over f)

```
double ac_access_mod_tdf::calcMaxAbsArgDevTarget(std::vector<double> arg_vec, double target_arg)
{

    double max_abs_diff_db  = 0;
    std::vector<double>::iterator iter;
    for(iter = arg_vec.begin(); iter < arg_vec.end(); iter++)
    {
        if( std::abs((*iter) - target_arg) > std::abs(max_abs_diff_db) )
            max_abs_diff_db = ((*iter) - target_arg);
    }

    return max_abs_diff_db;
}
```

- Calculate the amplitude spread (in some frequency region)

```
double ac_access_mod_tdf::calcMaxDBSpread(std::vector<double> abs_db_vec)
{
    double max_db = std::numeric_limits<double>::lowest();
    double min_db = std::numeric_limits<double>::max();

    std::vector<double>::iterator iter;
    for(iter = abs_db_vec.begin(); iter < abs_db_vec.end(); iter++)
    {
        if( (*iter) > max_db)
            max_db = *iter;

        if( (*iter) < min_db)
            min_db = *iter;
    }

    return max_db-min_db;
}
```

COS AC Postprocessor approach – Further Postprocessing

› Calculate dynamic behavior of amplitude vs. frequency

```
double ac_access_mod_tdf::calcMaxDBDynamic(std::vector<double> abs_db_vec, double f_st, double f_end, int N_f)
{
    double oct_per_point = (log2(f_end) - log2(f_st)) / double(N_f);

    double max_abs_diff_db  = 0;
    std::vector<double>::iterator iter;
    for(iter = abs_db_vec.begin(); iter < (abs_db_vec.end()-1); iter++)
    {
        if( std::abs( (*iter) - (*(iter+1)) ) > std::abs(max_abs_diff_db) )
            max_abs_diff_db = (*iter) - (*(iter+1));
    }

    return max_abs_diff_db / oct_per_point;

}
```

Agenda

1	Motivation	3
2	ACP - Setup	7
3	ACP - Postprocessing	10
4	ACP - Trigger Generation	14
5	ACP - Results	17
6	ACP - Outlook	22

COS AC Postprocessor approach – Trigger generation

› Cover all occurring values of an internal/external parameter

```
s.new_int = true;
s.new_ext = true;

// check for new internal trigger
for(std::vector<int>::iterator iter = s.trigger_intern_vals.begin(); iter < s.trigger_intern_vals.end(); iter++)
    if((*iter) == trigger_intern_i)
        s.new_int = false;

// check for new external trigger
for(std::vector<double>::iterator iter = s.trigger_extern_vals.begin(); iter < s.trigger_extern_vals.end(); iter++)
    if((*iter) == trigger_extern_i)
        s.new_ext = false;

if(s.new_int)
    s.trigger_intern_vals.push_back(trigger_intern_i);

if(s.new_ext )
    s.trigger_extern_vals.push_back(trigger_extern_i);

if(s.new_int || s.new_ext)
{

    s.acp.enable();
    s.results.clear();

    double f_st = 1.0; double f_end = 1.0e6; int N_f = 200;
    sca_ac_start(f_st, f_end, N_f, sca_ac_analysis::SCA_LOG);

    calcAbsArgFromComplVec(s.results, &s.results_db, &s.results_arg);
    s.max_abs_dev_db  = calcMaxAbsDBDevTarget (s.results_db, -60);
    s.max_abs_dev_arg = calcMaxAbsArgDevTarget(s.results_arg, 0);
    s.max_spread_db   = calcMaxDBSpread(s.results_db);
    s.max_dyn_db_per_oct = calcMaxDBDynamic(s.results_db, f_st, f_end, N_f);
```

COS AC Postprocessor approach – Trigger generation

- Easy if parameters can easily be (indirectly) set by TB – especially true for environmental parameters, might be hard for internal state
  - Check if all internal states of interest occurred can easily be implemented by comparing the vector to a target set

- AC analysis might as well be triggered at each change of a certain signal

- It might be necessary to section the parameter space if a too high number of possible states/values is possible

Agenda

1	Motivation	3
2	ACP - Setup	7
3	ACP - Postprocessing	10
4	ACP - Trigger Generation	14
5	ACP - Results	17
6	ACP - Outlook	22

COS AC Postprocessor approach – Providing results

› The generated results might be provided to the console to e.g. stop a time-consuming simulation during runtime

```
3001 us : T= 0 /  max abs diff db = -0.58754(max spread = 1.12476) / max diff arg = 3.72893 / max dynamic db/oct. = -0.401763
3501 us : T= 10 /  max abs diff db = 0.537785(max spread = 0.637173) / max diff arg = 2.11211 / max dynamic db/oct. = -0.228167
4001 us : T= 20 /  max abs diff db = 0.538231(max spread = 0.155605) / max diff arg = 0.514029 / max dynamic db/oct. = -0.0561039
4501 us : T= 30 /  max abs diff db = 0.858075(max spread = 0.320028) / max diff arg = -1.06513 / max dynamic db/oct. = 0.114122
5001 us : T= 40 /  max abs diff db = 1.32664(max spread = 0.788486) / max diff arg = -2.6223 / max dynamic db/oct. = 0.281895
5501 us : T= 50 /  max abs diff db = 1.78809(max spread = 1.25002) / max diff arg = -4.15687 / max dynamic db/oct. = 0.447495
6001 us : T= 60 /  max abs diff db = 2.24228(max spread = 1.70437) / max diff arg = -5.66351 / max dynamic db/oct. = 0.610108
6501 us : T= 70 /  max abs diff db = 2.6891(max spread = 2.15145) / max diff arg = -7.14502 / max dynamic db/oct. = 0.768791
7001 us : T= 80 /  max abs diff db = 3.12851(max spread = 2.5911
7501 us : T= 90 /  max abs diff db = 3.56052(max spread = 3.0234
```

```
state= 0 / T= 25 /  max abs diff db = -0.68213(max spread = 1.30332) / max diff arg = -4.30036 / max dynamic db/oct. = 0.469942
state= 1 /  max abs diff db = 0.621188(max spread = 0.991919) / max diff arg = -3.27463 / max dynamic db/oct. = 0.357575
state= 2 /  max abs diff db = 0.621189(max spread = 0.684625) / max diff arg = -2.26227 / max dynamic db/oct. = 0.246731
state= 3 /  max abs diff db = 0.621191(max spread = 0.381569) / max diff arg = -1.26375 / max dynamic db/oct. = 0.137334
state= 4 /  max abs diff db = 0.621193(max spread = 0.0938917) / max diff arg = -0.278221 / max dynamic db/oct. = 0.0292491
state= 5 /  max abs diff db = 0.83335(max spread = 0.305927) / max diff arg = -1.01473 / max dynamic db/oct. = 0.110013
state= 6 /  max abs diff db = 1.12334(max spread = 0.595805) / max diff arg = -1.97111 / max dynamic db/oct. = 0.214858
state= 7 /  max abs diff db = 1.40892(max spread = 0.88128) / max diff arg = -2.9113 / max dynamic db/oct. = 0.317775
```

COS AC Postprocessor approach – Providing results

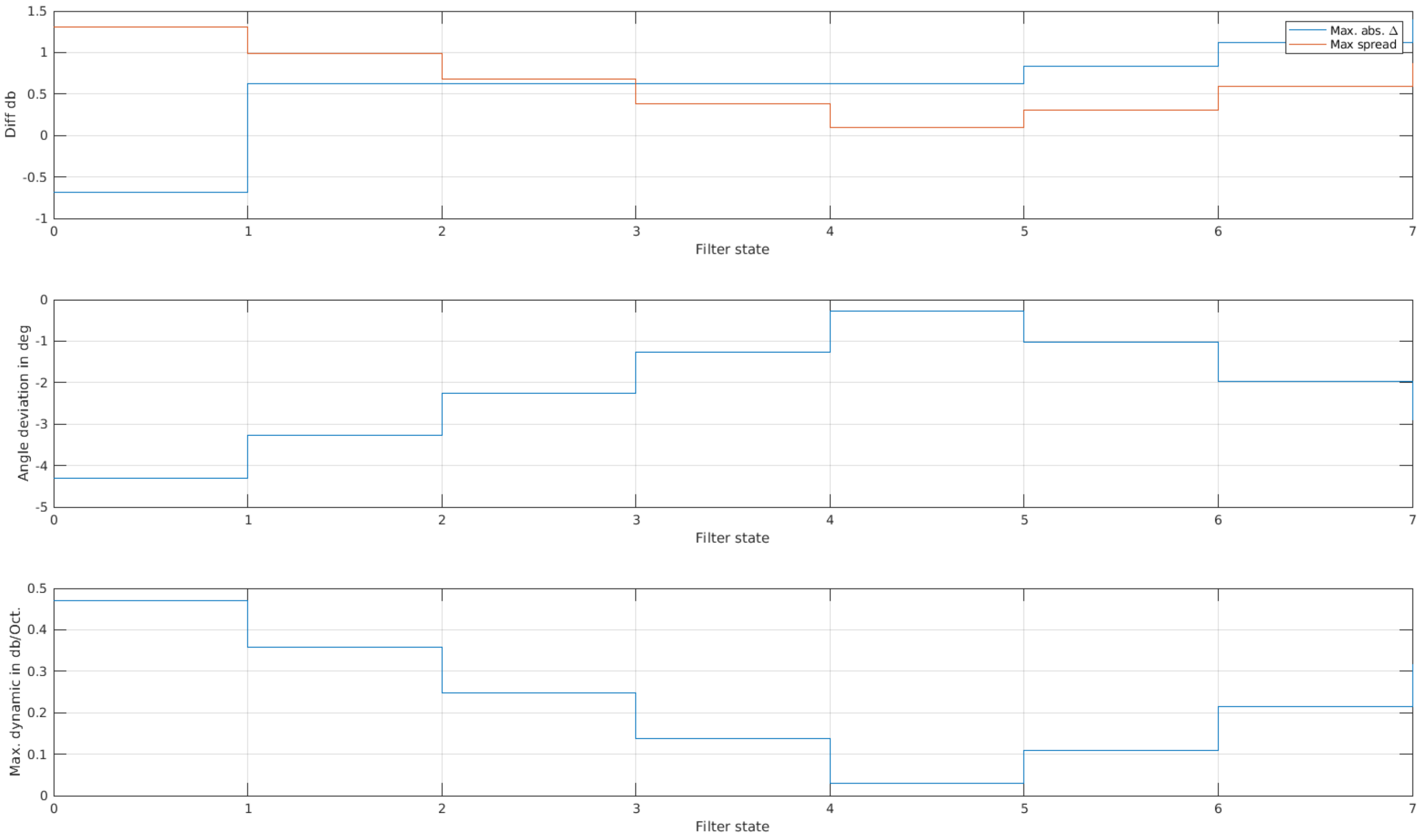
› The results can be exported e.g. by writing to csv files

f_state	T	max_abs_diff_db	max_spread	max_diff_arg	max_dynamic
4	-20	-1.5799	2.11526	6.99557	-0.751068
4	-10	-1.08128	1.61770	5.35706	-0.575607
4	0	-0.58754	1.12476	3.72893	-0.401763
4	10	0.537785	0.637173	2.11211	-0.228167
4	20	0.538231	0.155605	0.514029	-0.0561039
4	30	0.858075	0.320028	-1.06513	0.114122
4	40	1.32664	0.788486	-2.6223	0.281895
4	50	1.78809	1.25002	-4.15687	0.447495
4	60	2.24228	1.70437	-5.66351	0.610108
4	70	2.68910	2.15145	-7.14502	0.768791
4	80	3.12851	2.59117	-8.60055	0.925788
4	90	3.56052	3.02347	-10.0228	1.07844
4	100	3.98516	3.44834	-11.4187	1.22654
4	110	4.40249	3.86580	-12.7884	1.37317
4	120	4.81259	4.27587	-14.1225	1.51486
4	130	5.21557	4.67862	-15.4258	1.65129

f_state	T	max_abs_diff_db	max_spread	max_diff_arg	max_dynamic
0	25	-0.68213	1.30332	-4.30036	0.469942
1	25	0.621188	0.991919	-3.27463	0.357575
2	25	0.621189	0.684625	-2.26227	0.246731
3	25	0.621191	0.381569	-1.26375	0.137334
4	25	0.621193	0.0938917	-0.278221	0.0292491
5	25	0.83335	0.305927	-1.01473	0.110013
6	25	1.12334	0.595805	-1.97111	0.214858
7	25	1.40892	0.88128	-2.9113	0.317775

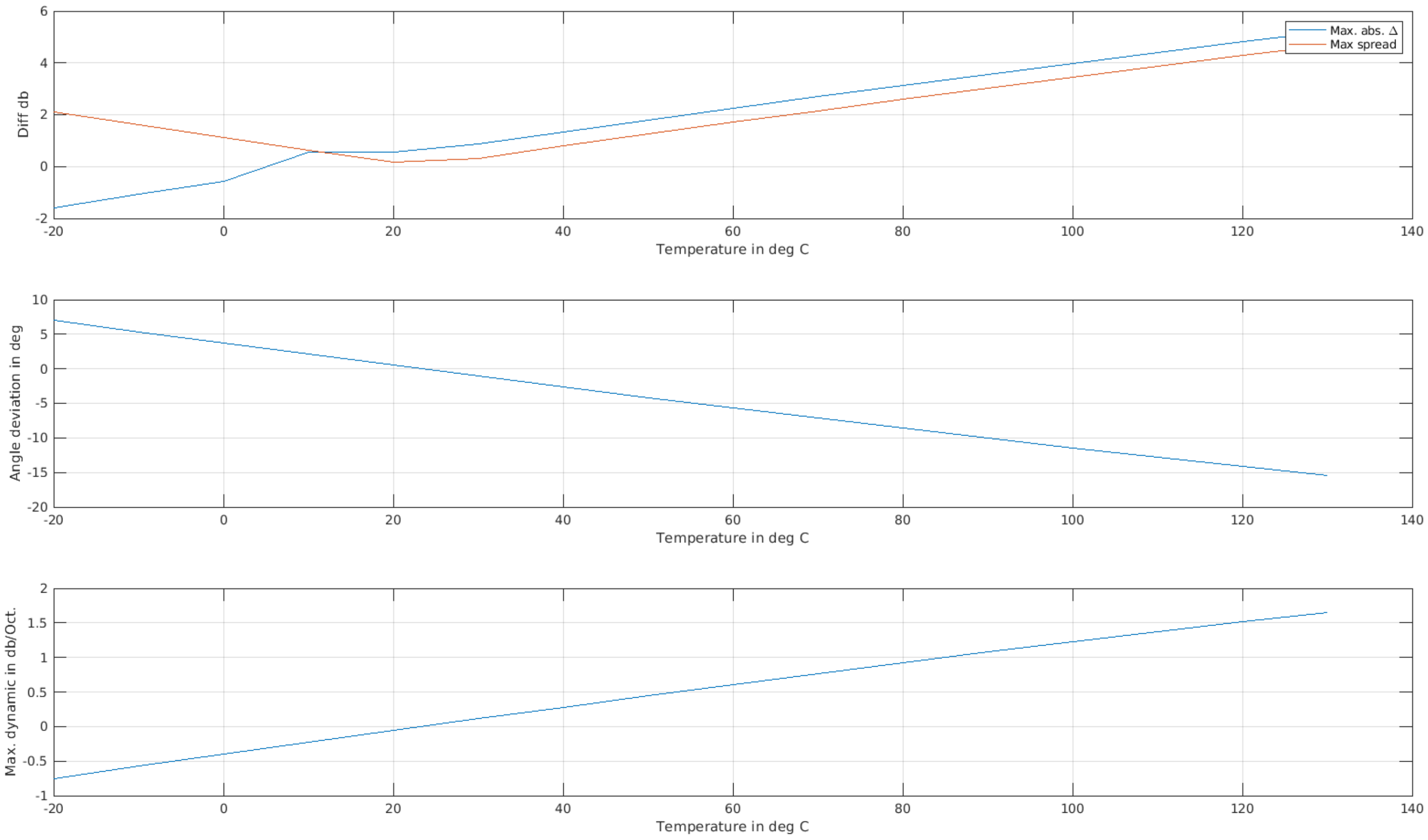
COS AC Postprocessor approach – Providing results

› … and might get further processed by some script language



COS AC Postprocessor approach – Providing results

› … and might get further processed by some script language



Agenda

1	Motivation	3
2	ACP - Setup	7
3	ACP - Postprocessing	10
4	ACP - Trigger Generation	14
5	ACP - Results	17
6	ACP - Outlook	22

COS AC Postprocessor approach – Outlook

- Results could be written to output ports and be further processed
- Runtime calculated values could be used to drive the test bench
- Structure to efficiently deal with huge parameter spaces
- Library of generic processing blocks
- …

Part of your life. Part of tomorrow.