Data Center Platform
Engineering & Architecture

EPSI

IBIS

Asian IBIS Summit 2023 in Shanghai

AMI DLL Hook: A novel debug method for IBIS-AMI simulation

Presenter: Chuanyu Li

November 10, 2023

intel

Agenda

- Background
- DLL hook concept
- Previous hook methods in IBIS-AMI
- DLL hook implementation in AMI simulation debug
- Story sharing: a debug experience with AMI DLL hook
- Summary

Background

- When validating our IBIS-AMI models, the following problems appear:
  - **Trend** of batch simulation is **not well-matched** to silicon simulation.
  - Simulation results are **different in different EDA software**.

- As a **model user**, the following troubles are slowing down our debug progress:
  - No access to EDA or IBIS-AMI source code.
  - Limited understanding of mathematical calculation process of EDA. This may lead to improper setting during simulation.
  - Limited debug dump data from EDA or AMI model.
- AMI DLL hook is therefore introduced to speed up our debug.

Background

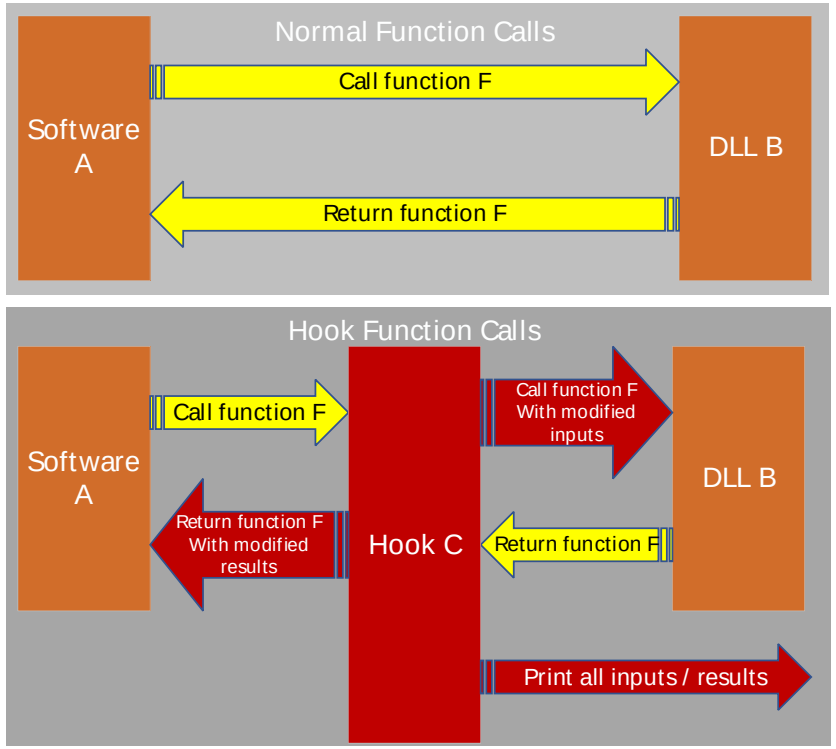
- Target audience:
  - IBIS-AMI model user.
  - IBIS-AMI model validator working with model provider.
  - Have basic C/C++ knowledge.

DLL hook concept

- A bundle of programming technology used to **alter the behavior** of applications / software components, by **intercepting function calls** between software components.[1]

- Neither Software A nor DLL B is aware of the existence of hook C.

- Hooking has been used in software domain for a while. [1]



[1] https://en.wikipedia.org/wiki/Hooking

Previous hook methods in IBIS-AMI

- Xilinx has proposed "debugging hook" in *A New Methodology for Developing IBIS-AMI Models*, on DesignCon 2015
  - It is suggested to add hooks between components during model development phase, to make debugging easier.
  - It is a good proposal but may require model builder's work. As a user, we can't add hook if the models are already there.

DLL hook implemented in AMI simulation debug

- A Hook with standard AMI interface (so it can cheat EDA).
- Call original AMI DLL inside private hook codes.

```
extern "C" __declspec(dllexport) long AMI_Init(double* impulse_matrix, long number_of_rows, long aggressors, double sample_interval, double symbol_time, char* AMI_parameters_in, char** AMI_parameters_out, void** AMI_memory_handle,
extern "C" __declspec(dllexport) long AMI_GetWave(double* wave, long wave_size, double* clock_times, char** AMI_parameters_out, void* AMI_memory);
extern "C" __declspec(dllexport) long AMI_Close(void* AMI_memory);
extern "C" __declspec(dllexport) long AMI_Impulse(double* impulse_matrix, char* BCI_parameters_in, char** BCI_parameters_out, char** AMI_parameters_out, void* AMI_memory);
extern "C" __declspec(dllexport) long AMI_Resolve(double symbol_time, char* AMI_parameters_in, char** AMI_parameters_out);
extern "C" __declspec(dllexport) long AMI_Resolve_Close(char* AMI_parameters_out);
```

Standard AMI interface

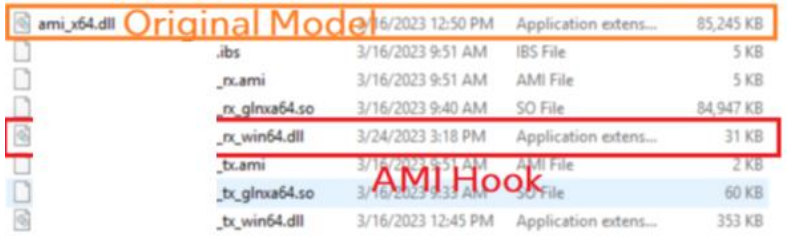
```
typedef long (CALLBACK* pFN_AMI_Init)(double* impulse_matrix, long number_of_rows, long aggressors, double sample_interval, double symbol_time, char* AMI_parameters_in, char** AMI_parameters_out, void** AMI_memory_handle, char** ms
typedef long (CALLBACK* pFN_AMI_GetWave)(double* wave, long wave_size, double* clock_times, char** AMI_parameters_out, void* AMI_memory);
typedef long (CALLBACK* pFN_AMI_Close)(void* AMI_memory);
typedef long (CALLBACK* pFN_AMI_Impulse)(double* impulse_matrix, char* BCI_parameters_in, char** BCI_parameters_out, char** AMI_parameters_out, void* AMI_memory);
typedef long (CALLBACK* pFN_AMI_Resolve)(double symbol_time, char* AMI_parameters_in, char** AMI_parameters_out);
typedef long (CALLBACK* pFN_AMI_Resolve_Close)(char* AMI_parameters_out);
```

Callback definitions

```
class AMI_hook
{
public:
    AMI_hook(                                                                                                    );
    ~AMI_hook();
```

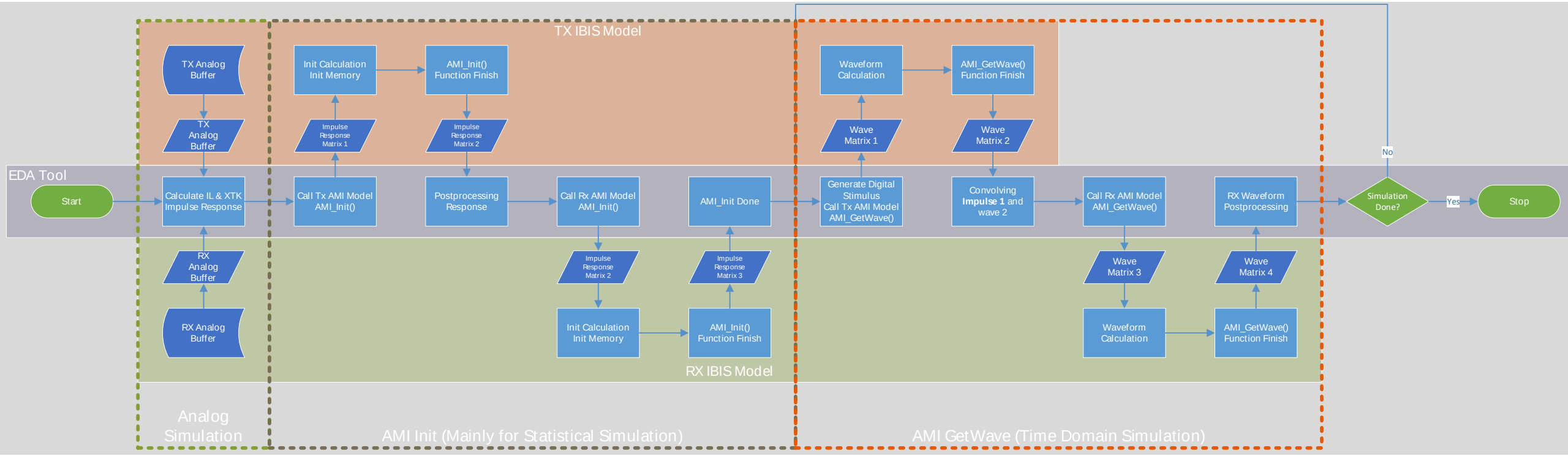
Private hook codes

- Rename the hook dll to ami dll name, and change original dll to internal callback name
- Each dll needs an individual hook.



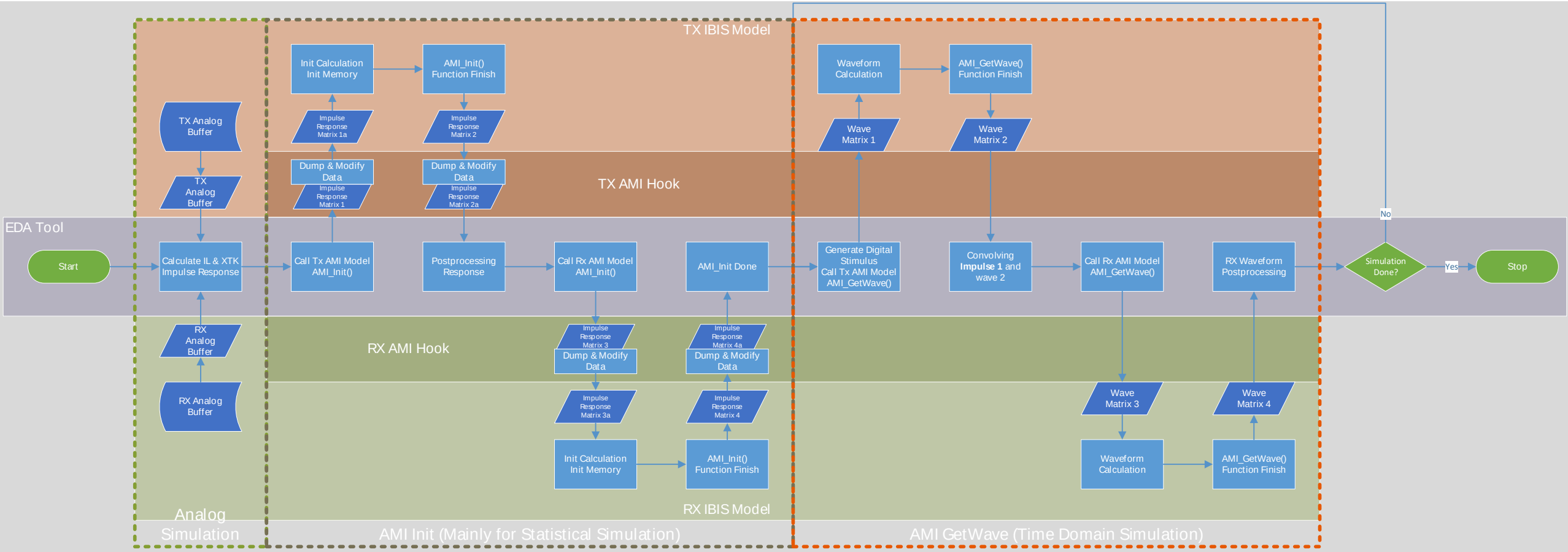
DLL hook implemented in AMI simulation debug

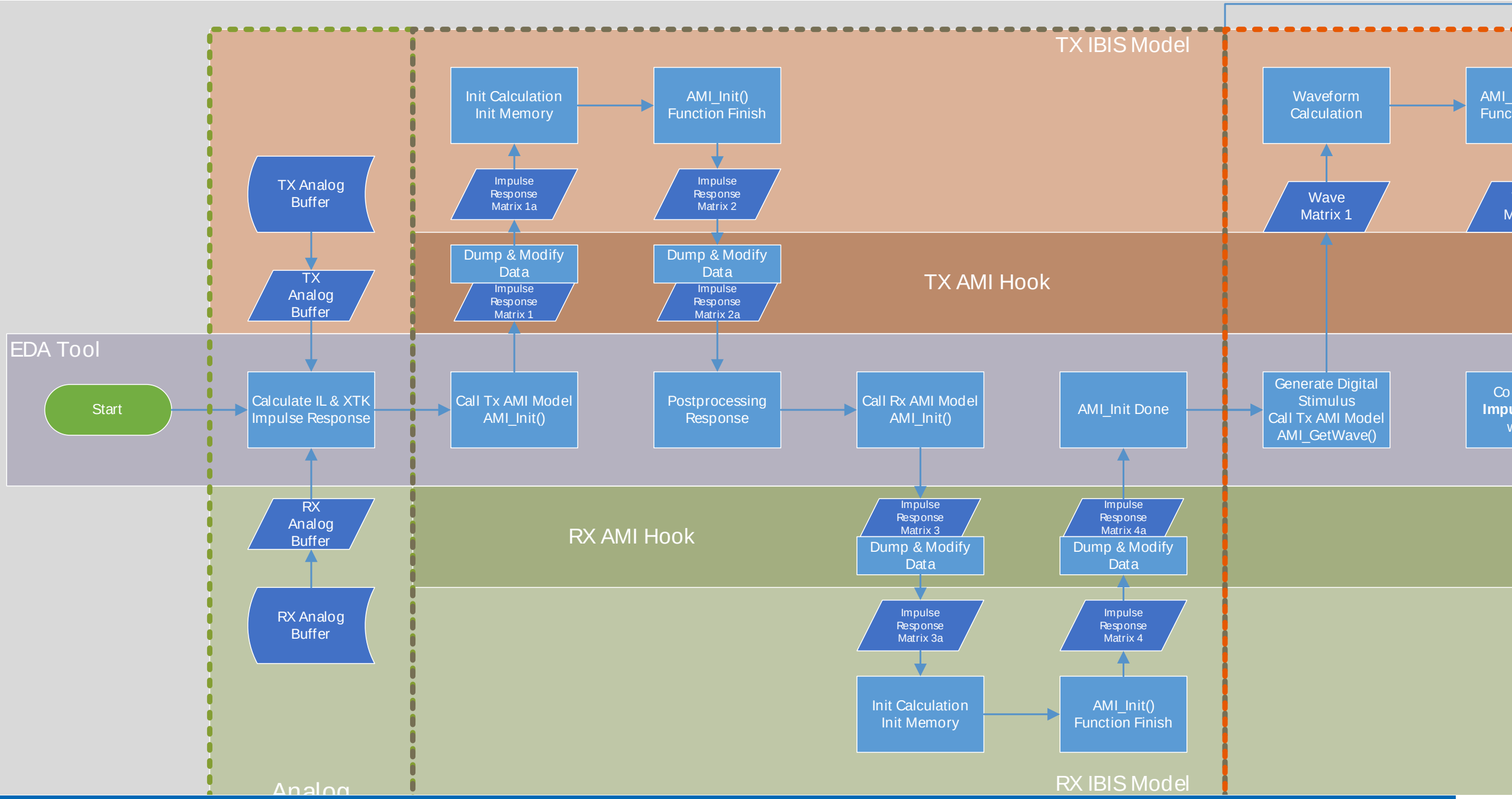
- EDA calculation process **before** implementing AMI DLL hook

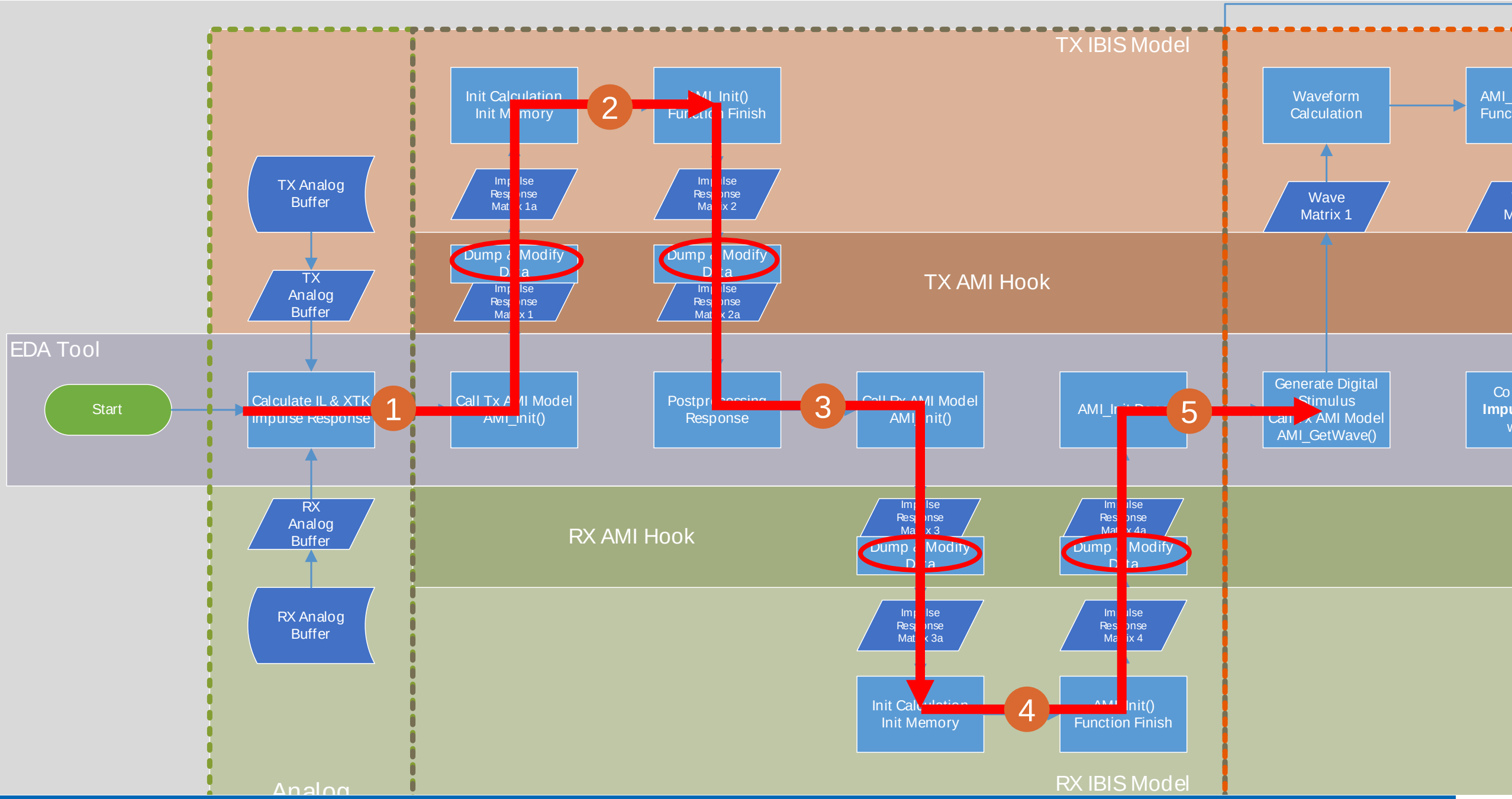


DLL hook implemented in AMI simulation debug

- EDA calculation process **after** implementing AMI DLL hook
- Note: Simulation will be slower with hook injected.





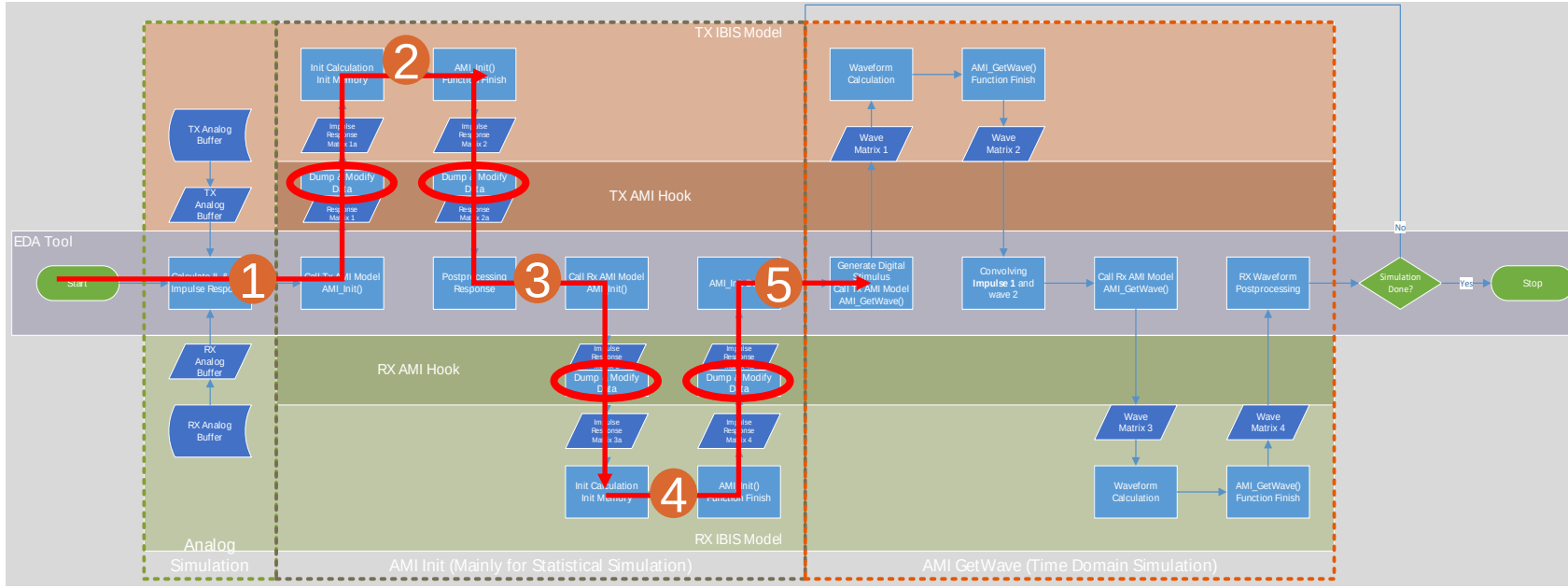


DLL hook implemented in AMI simulation debug

- EDA calculation process **after** implementing AMI DLL hook
- The much bigger problem (results mismatch) will be divided into smaller problems, thus helped debug.
- Example is AMI_Init. The processing of AMI_Init is divided into 5 parts (1-5) as shown.

Note:
In time domain simulation, although AMI_Init steps 2-5 are not considered, it will still process and provide important information for debug.
Details will be shown in upcoming section.

We can also implement hook into AMI_Getwave, but we won't dig into it today.



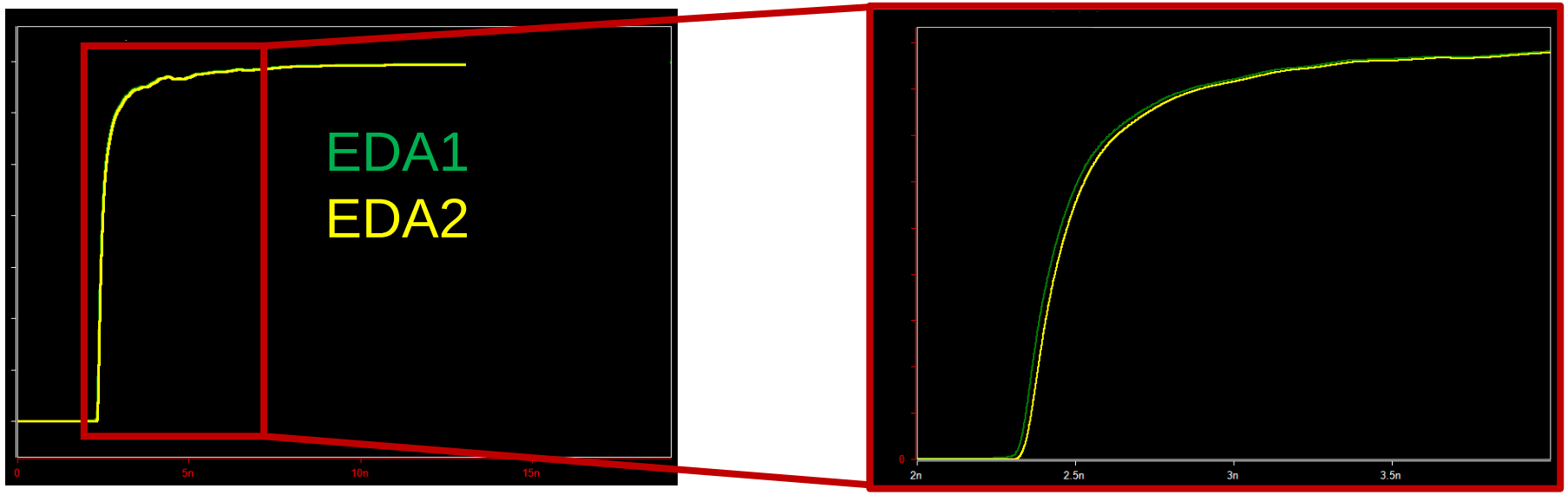
Story sharing:
a debug experience with AMI DLL hook

A debug experience with AMI DLL hook

- The problems:
  - Trend of batch simulation is not well-matched to silicon simulation.
  - Simulation results are different in different EDA software.
- The questions:
  - Is there any not-aligned setting between silicon simulation and our deck?
  - Is there any not-aligned / wrong setting between EDAs?
  - Is there any bug in AMI model?
  - Is there any bug in either EDA?

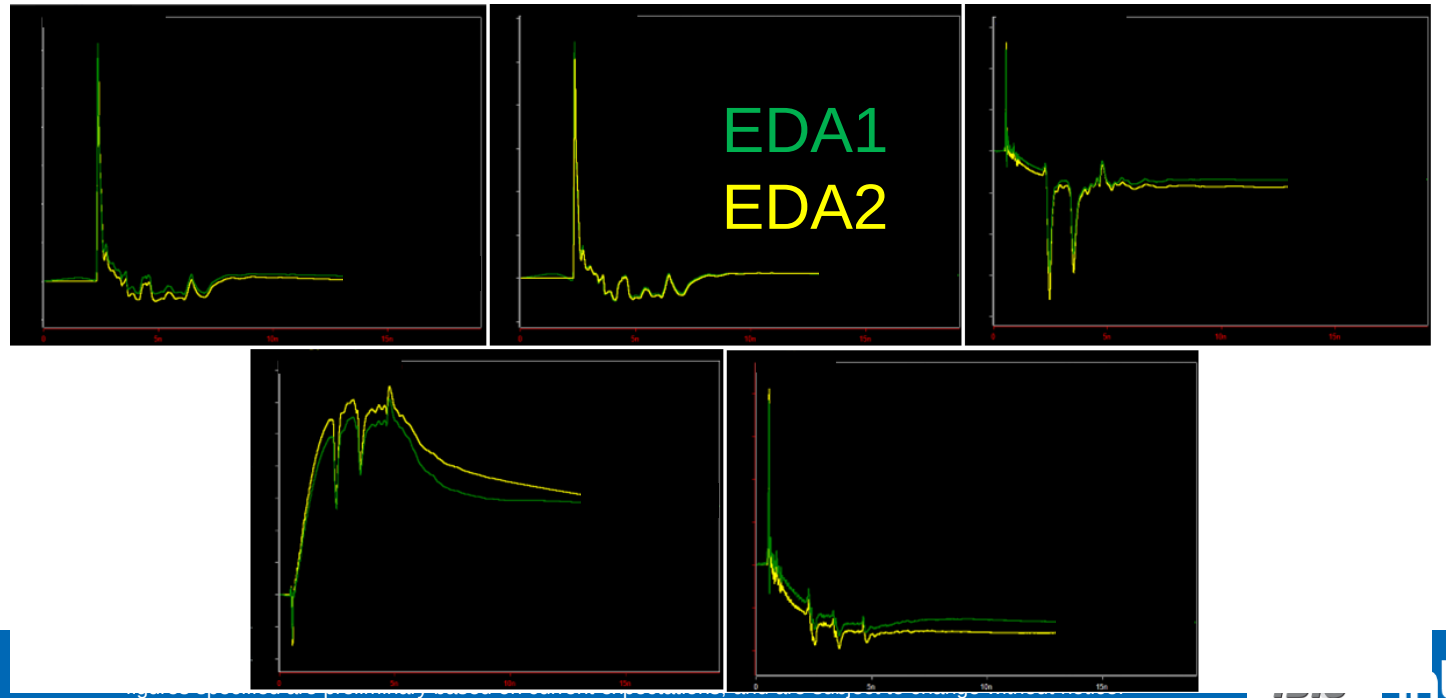
1. Passive channel alignment

- Check step response:
  - Hook is not needed. Both EDAs can provide step response dump.
  - *Step responses are very similar

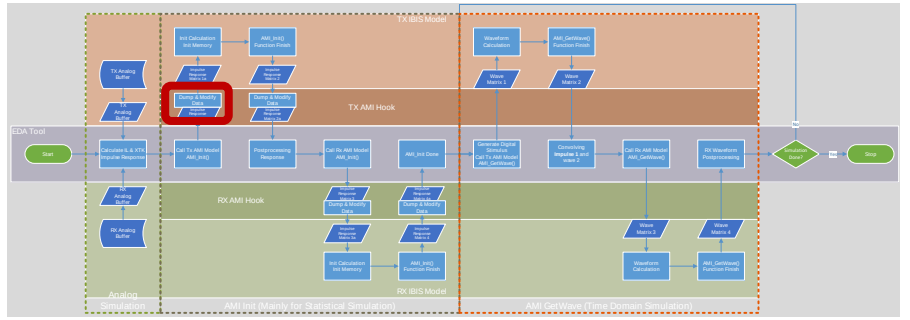


1. Passive channel alignment

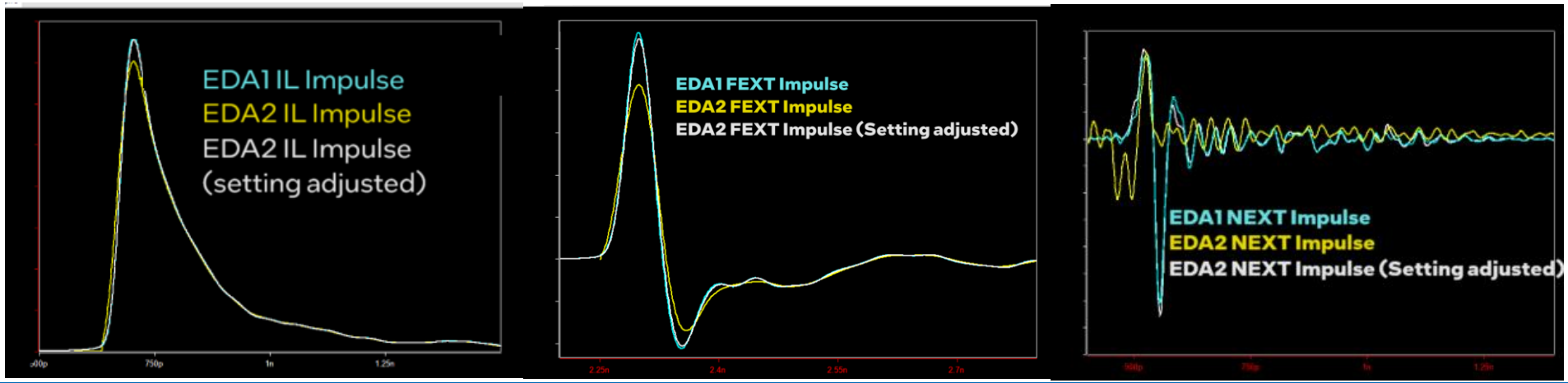
- Check step response:
  - Hook is not needed. Both EDAs can provide step response dump.
  - *Step responses are very similar
  - * Xtalk channels are also similar (*but not exactly same)



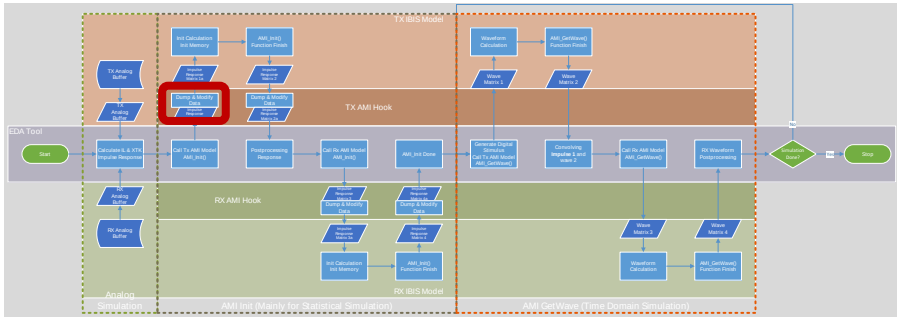
1. Passive channel alignment



- Check impulse response:
  - Hook is needed. Only one of the two EDAs can provide impulse response dump.
  - Difference between EDAs is significant when looking at impulse response. This is later found caused by improper default settings in EDA2.
  - After fixing the settings in EDA2, both EDAs have almost same impulse.

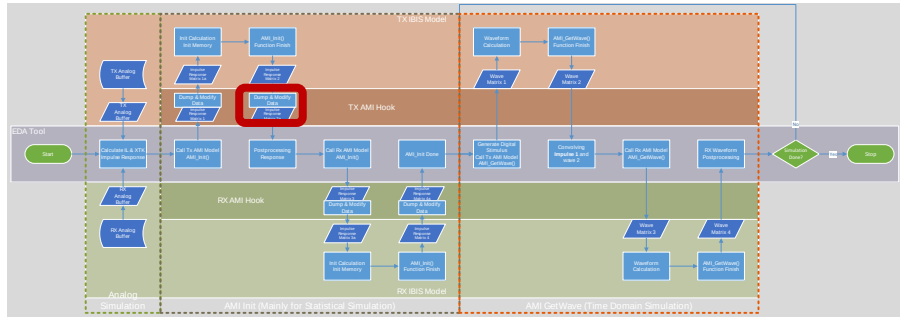


1. Passive channel alignment



- What has hook helped?
  - Dumped step responses, directly from EDA, are not that different.
  - By hooking at TX IBIS AMI input side and dumping impulse response, we divided the big problem (results not matching) into a small problem (input impulse response differed from same passive channel)
  - Solving this small problem is much easier, that it doesn't need long time run and it limited the range of affected settings.

2. AMI input problem

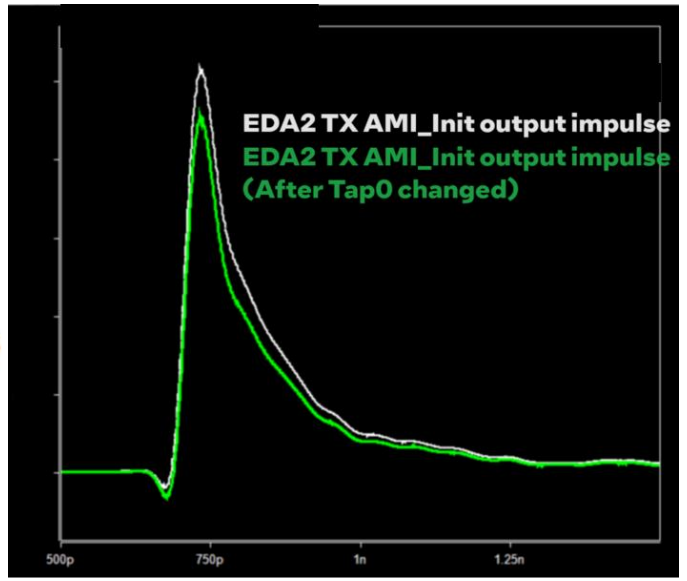


- Check TX output impulse response:
  - Hook is needed. Neither EDA could provide this dump.
  - AMI input in one of the EDAs is suspicious and later confirmed a problem.
  - This is due to our batch run script will provide formula with variables to the AMI input. EDA2 is not dealing with formulas in AMI_parameters_in section.

```
number_of_rows:
aggressors:0
sample_interval:
symbol_time:
AMI_parameters_in:(
    (AMI_Version "6.1")
    (Init_Returns_Impulse True)
    (GetWave_Exists True)
    (Max_Init_Aggressors 6)
    (Modulation "NRZ")
```

EDA1 AMI_Init all inputs

```
    (TapWeights
      (-1 -0.125)
      (0 0.75)
      (1 -0.125)
    )
  )
)
```

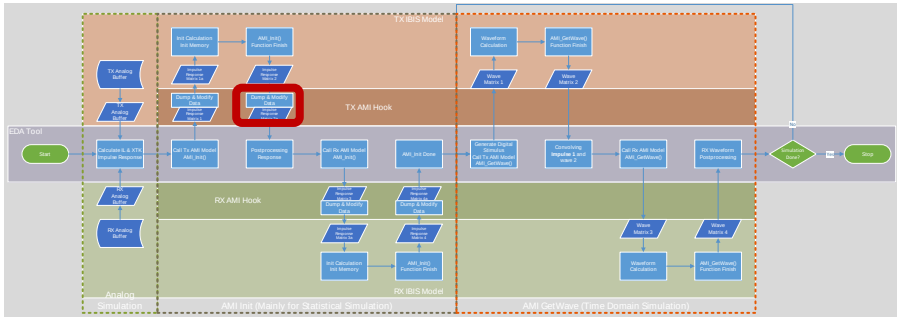



```
number_of_rows:
aggressors:5
sample_interval:
symbol_time:
AMI_parameters_in:(
```

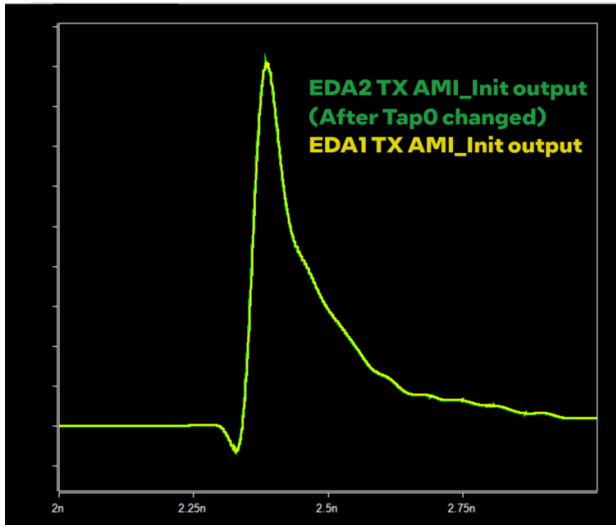
EDA2 AMI_Init all inputs

```
)(TapWeights(-1 -0.125)(0 1+-0.125+-0.125)(1 -0.125)))(Modulation "NRZ") )
```

2. AMI input problem



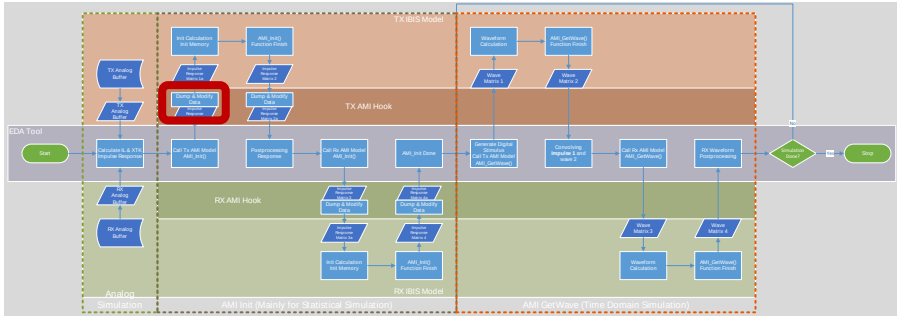
- Check TX output impulse response:
  - After fixing the AMI_parameters_in, both EDAs got same impulse at TX out.



- What has hook helped?
  - It is hard to imagine this issue without hook dumped data
  - It also limited the problem in the range of after TX input but before TX output, so the problem is small and easy to find root cause.

3. Different behaviors in dealing with Xtalk channels

- Found by AMI_Init dump.
  - EDA1: AMI_Init with 0 aggressors. Xtalk channels are treated separately. (Another AMI_init with 1 aggressor)
  - EDA2: AMI_Init with real aggressors.



```
humber_of_rows:
aggressors:0
sample_interval:
symbol_time:
AMI_parameters_in:(
    (AMI_Version "6.1")
    (Init_Returns_Impulse True)
    (GetWave_Exists True)
    (Max_Init_Aggressors 6)
    (Modulation "NRZ")
```

EDA1 AMI_Init dump

```
    (TapWeights
      (-1 -0.125)
      (0 0.75)
      (1 -0.125)
    )
  )
)
```

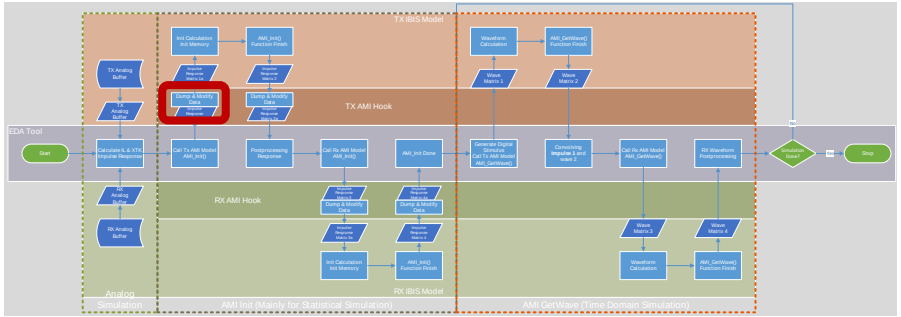
```
humber_of_rows:
aggressors:5
sample_interval:
symbol_time:
AMI_parameters_in:(
```

EDA2 AMI_Init dump

3. Different behaviors in dealing with Xtalk channels

- What has hook helped?
  - We learned the xtalk special processing in EDA1 and may introduce more consideration in future projects.

- An open question:
  - Which way is more common in modern EDAs?
  - What is the recommendation from IBIS open forum?



Summary

- AMI DLL hook can help IBIS-AMI simulation debug by:
  - Split the big problem into multiple checkpoints and thus multiple smaller problems. Each of the small problem is much easier to solve.
  - Provide much more extra information to SI engineer who don't have access to model or software source codes.
- AMI DLL hook has helped and provided 3 significant helps in one of our projects as story sharing suggested.

Open discussion to IBIS Open Forum

- Is it possible to EDA software to provide all IBIS-AMI API dumps to general users? Maybe open it with some internal debug environment variable, etc.

Notices & Disclaimers

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein.

You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at Intel.com, or from the OEM or retailer.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visit www.intel.com/design/literature.htm.

Intel, and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Thank You!

chuanyu.li@intel.com