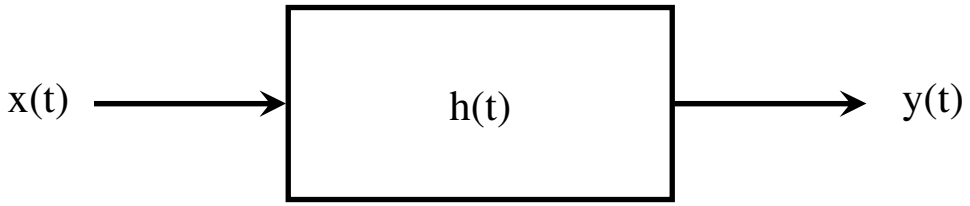
Channel analysis flow with IBIS

IBIS Advanced Technology Modeling Teleconference

October 24, 2006

Arpad Muranyi
Intel Corporation
Signal Integrity Engineering

Channel analysis primer



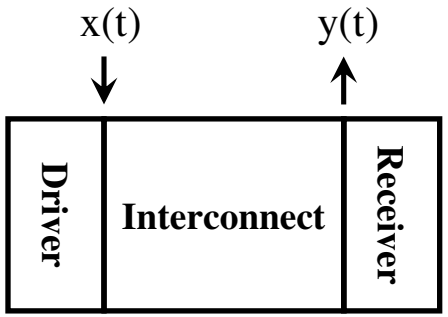
If x(t) is the input to a linear system, and h(t) is the system's impulse response, then the output is y(t) = x(t) ⊗ h(t), where ⊗ is convolution.

Although most of this may be easier to do in the frequency domain, I will only talk about time domain in this presentation for simplicity.

The impulse response h(t) can be derived from s-parameters.

If x(t) is a digital pulse, y(t) will be the pulse response.

In channel analysis, the driver and receiver model must be included to get the correct waveforms.



Two ways to get a pulse response

1

- "Characterize" the system including the driver and receiver models (which are in steady state) to obtain the impulse response h(t) of the channel.
- Convolve a pulse waveform x(t) with h(t) to get the pulse response. This waveform may have non-ideal edges to represent the driver's switching characteristics.

2

- Apply a digital pulse (stimulus) to the driver's input and let the time domain simulator generate a pulse response using the driver, receiver and interconnect models with a normal time domain simulation.

In either case, the driver and receiver characteristics (Z_{11}) have to be known (set) so that their interaction with the channel would generate correct waveforms, i.e.:

- **the driver's drive strength (tap coefficients) must be set,**
- **the receiver's termination impedance must be set**

Once we have a pulse response

… we can do any of the following tasks with statistical numerical analysis techniques:

- calculate the worst case eye opening,
- calculate bath tub curves,
- calculate bit error rate (BER),
- calculate worst case bit patterns,
- etc…

Using these results we can optimize the channel by adjusting the

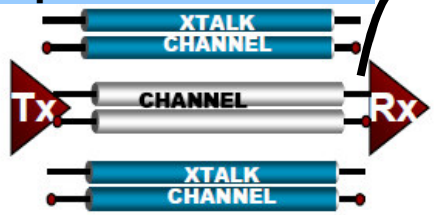
- driver's drive strength, edge rate, pad capacitance, etc…
  - this will usually involve optimizing the "tap coefficients"
- receiver's terminator impedance, pad capacitance, etc…
- package characteristics,
- T-line impedance (board stack up), line length, line spacing (cross talk),
- etc…

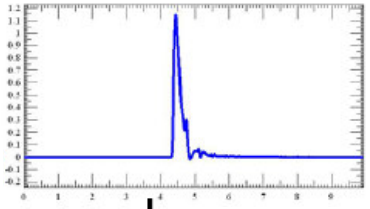
Kumar's optimization loop (annotated)

START AMI_Init

after choosing an initial set of parameters (Init), the IBIS simulator generates a pulse response

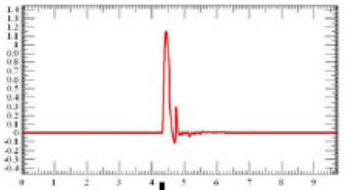
pulse response waveform sent to statistical engine (GetWfm)





statistical engine calculates eye, BER, etc…

IBIS simulator's optimizer may select new model parameters to generate new pulse response (Init)



statistical engine returns results to IBIS simulator to be displayed (PutWfm?)

Internal storage

Channel model requirements

The optimization loop on the previous page can use normal IBIS or IBIS *-AMS models for Tx and Rx

However, the optimization loop requires that the model's parameters are accessible to the optimizer tool (the IBIS simulator)

- IBIS-A(MS) models can be fully parameterized
- but range and step (increment) definitions need to be added to the IBIS specification for [External Model] and [External Circuit] calls
- anything else missing? (goal, error function, etc…)?

A simple IBIS BIRD could be written to add range and step definitions for the *-AMS calls

- none of these additions would define a ***methodology***

How about optimizing interconnect models (ICM)?

- can the IBIS tool take care of this on its own based on geometric T-line and stack up descriptions?
- should this be done by parameter passing into ICM models?
- do we need range and step for ICM, or would an "ICM selector" be a better choice?

Once the channel is optimized

… we can further optimize the system by making adjustments to the receiver's equalization circuit (if we have one in the receiver, that is) by setting its EQ tap coefficients

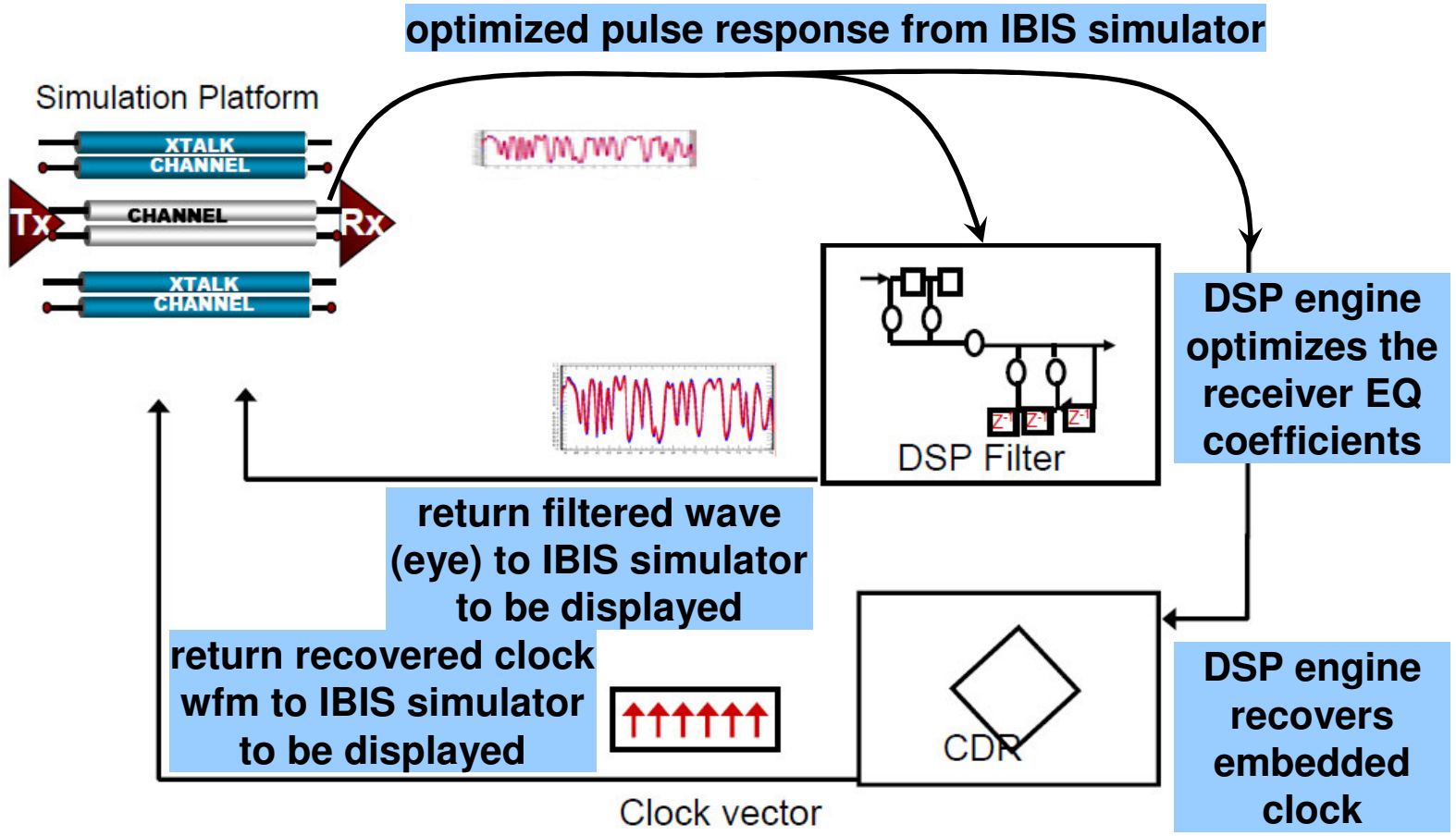
This step is entirely a "DSP" process and *may be* done independently from the IBIS simulator

The receiver EQ model *may be* a completely independent model written for a DSP engine, *but it could also be part of the *-AMS receiver model*

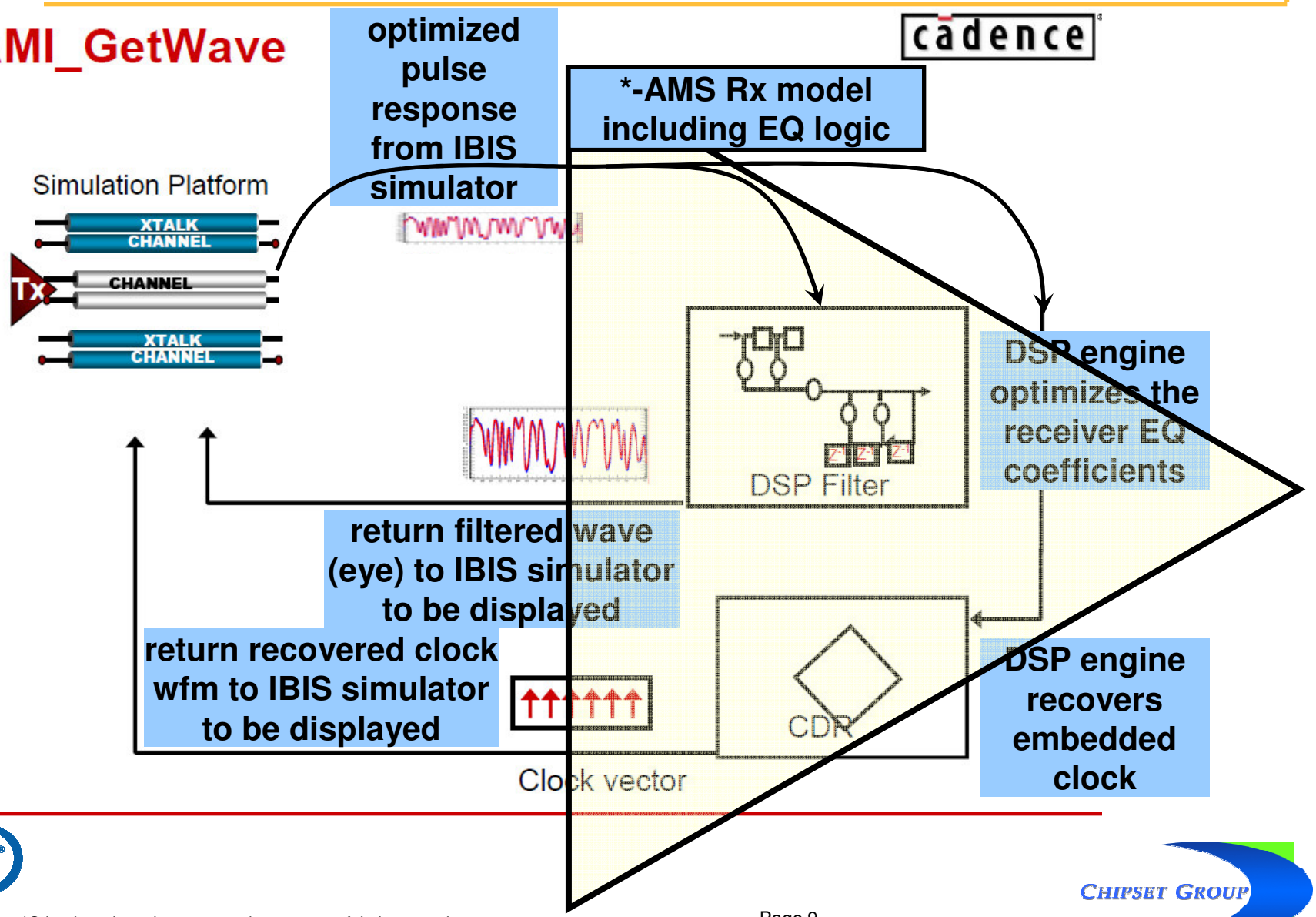
- the DSP engine could use the same receiver model as the IBIS simulator for its own purpose
- in this case it would be useful to have a "disable" switch for the EQ portion of the receiver model to save CPU time while generating/optimizing the pulse response waveforms

Kumar's DSP loop (annotation 1)

AMI_GetWave



Kumar's DSP loop (annotation 2)



API requirements

The optimization loops shown on the previous pages exchange nothing more than waveform information between the IBIS simulator and the statistical engine

This waveform exchange could very well be done within the capabilities of the *-AMS languages

The *-AMS languages seem to have their own API definitions already which could do all this

It doesn't seem to make sense to develop "yet another API" for this purpose (reinvent the wheel)