Serdes Channel Simulation Algorithms and Workflows Using IBIS AMI Models

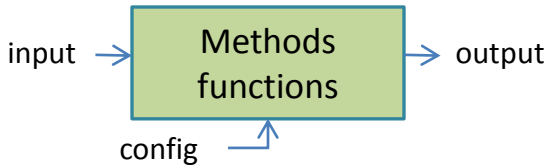
James Zhou, QLogic Corp.
James.zhou@qlogic.com

IBIS Summit at DesignCon
Santa Clara, CA
January 31, 2013

AMI Reference Flow – Brief History

- BIRD 104.1, (10/2007)
  - One of the earlier public proposals of IBIS-AMI
- BIRD 107.1, and IBIS Specification 5.0, (05/2008, 08/2008)
  - Introduced Use_Init_Output to solve the double counting issue when filtering exist in both AMI_Init and AMI_Getwave functions
  - Added dedicated section to describe reference flow
- BIRD 120.1 (04/2011)
  - Deprecated Use_Init_Output
  - Revised reference flow section to separate statistical and time-domain flows
  - Corrected inconsistencies in IBIS 5.0 flow for NLTV systems

What Are AMI Reference Flows ?

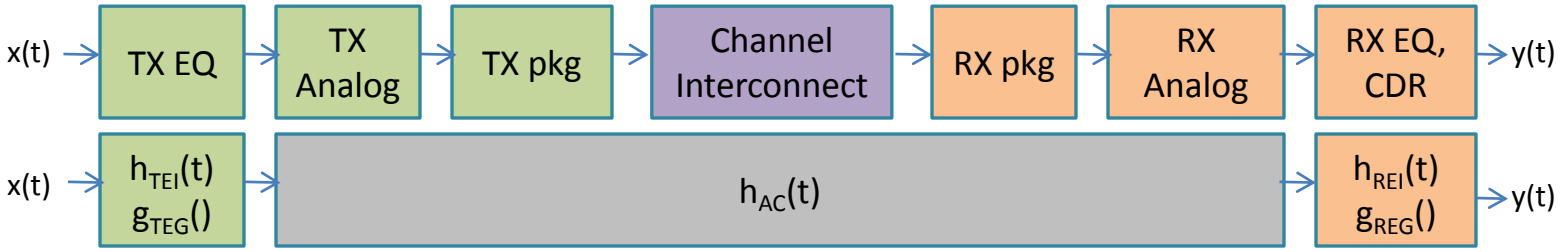


- A standardized workflow is a process to
  - analyze input data
  - use prescribed methods/functions
  - achieve a unique result true to the system
- AMI References flows are processes to analyze
  - channel performance using statistical and time-domain methods
  - Tx/Rx behaviors contained in *.dll
  - System configurations in *.ami
  - Interface between algorithmic and analog blocks

IBIS AMI FAQ

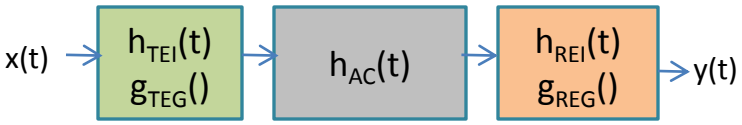
- What are the definitions of AMI_Init, AMI_Getwave input/output quantities?
- What is the content and format of AMI analog model?
- How many cases are there and, how to resolve AMI reference flows for various combinations of Tx/Rx Init_Returns_Impulse, GetWave_Exists?
- What are the assumptions made in the IBIS-AMI models and reference flows ?
- What are the mandatory vs. optional features of AMI flow ?

IBIS AMI – Basic Concepts



- x(t) is input, y(t) is output
- All channel interconnects are cascaded into a single block, which impulse response is $h_{AC}(t)$
- Tx/Rx black box (algorithmic) characteristics are provided by two functions of _Init() and _GetWave() ($h_{TEI}(t)$, $g_{TEG}(t)$, $h_{REI}(t)$, $g_{TEG}(t)$)
- _Init() functions are for LTI systems and _GetWave() functions are for NLTV systems
- The _GetWave() functions are the only two black boxes in the system

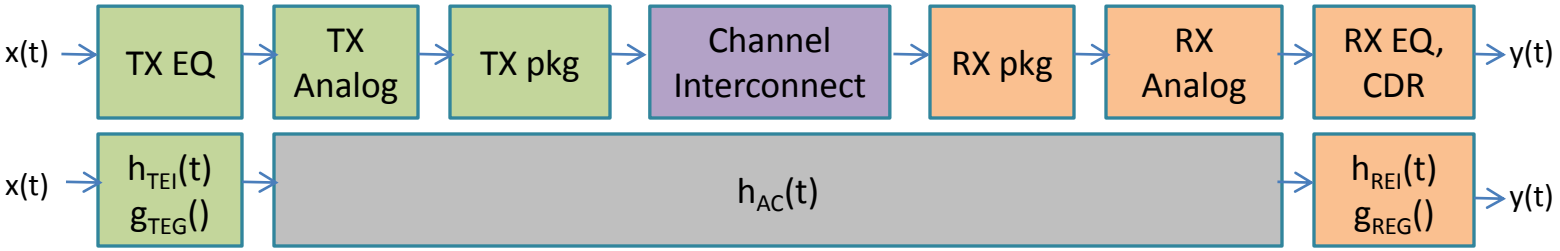
System Equations



Case 1: $y(t) = h_{REI}(t)*h_{AC}(t)*h_{TEI}(t)*x(t)$
Case 2: $y(t) = g_{REG}[h_{AC}(t)*h_{TEI}(t)*x(t)\ ]$
Case 3: $y(t) = h_{REI}(t)*h_{AC}(t)*g_{TEG}[x(t)]$
Case 4: $y(t) = g_{REG}[h_{AC}(t)*g_{TEG}[x(t)]]$

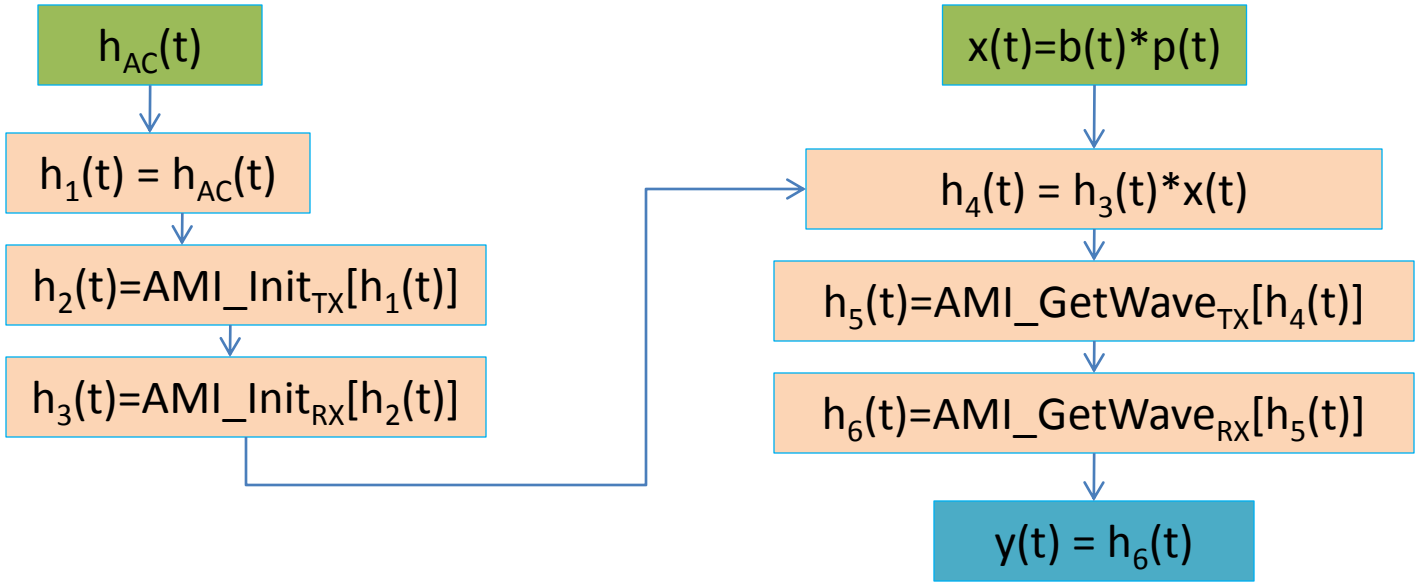
- Based on signals and systems theory 101, the system equations are derived readily and shown above
- Now that we have the system equations, what's next?

Reference Flow – IBIS 5.0



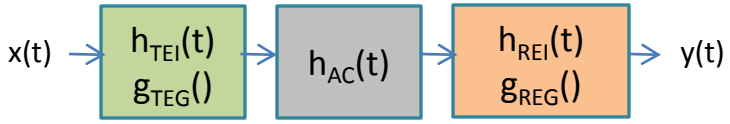
Step 1: $h_1(t) = h_{AC}(t)$	
Step 2a: $h_{2a}(t) = AMI\_Init_{TX}[h_1(t)] = h_{TEI}(t)*h_{AC}(t)$ Step 2b: $h_{2b}(t) = h_1(t) = h_{AC}(t)$	(Tx Use_Init_Output = TRUE) (Tx Use_Init_Output = FALSE)
Step 3a: $h_{3a}(t) = AMI\_Init_{RX}[h_2(t)] = h_{REI}(t)*h_2(t)$ Step 3b: $h_{3b}(t) = h_2(t)$	(Rx Use_Init_Output = TRUE) (Rx Use_Init_Output = FALSE)
Step 4: $h_4(t) = h_3(t) * b(t) * p(t)$	
Step 5a: $h_{5a}(t) = AMI\_GetWaveTX[h_4(t)]$ Step 5b: $h_{5b}(t) = h_4(t)$	(Tx GetWave_Exists = TRUE) (Tx GetWave_Exists = FALSE)
Step 6a: $h_{6a}(t) = AMI\_GetWaveRX[h_5(t)]$ Step 6b: $h_{6b}(t) = h_5(t)$	(Rx GetWave_Exists = TRUE) (Rx GetWave_Exists = FALSE)
Step 7: $y(t) = h_6(t)$	

Reference Flow Diagram – IBIS 5.0

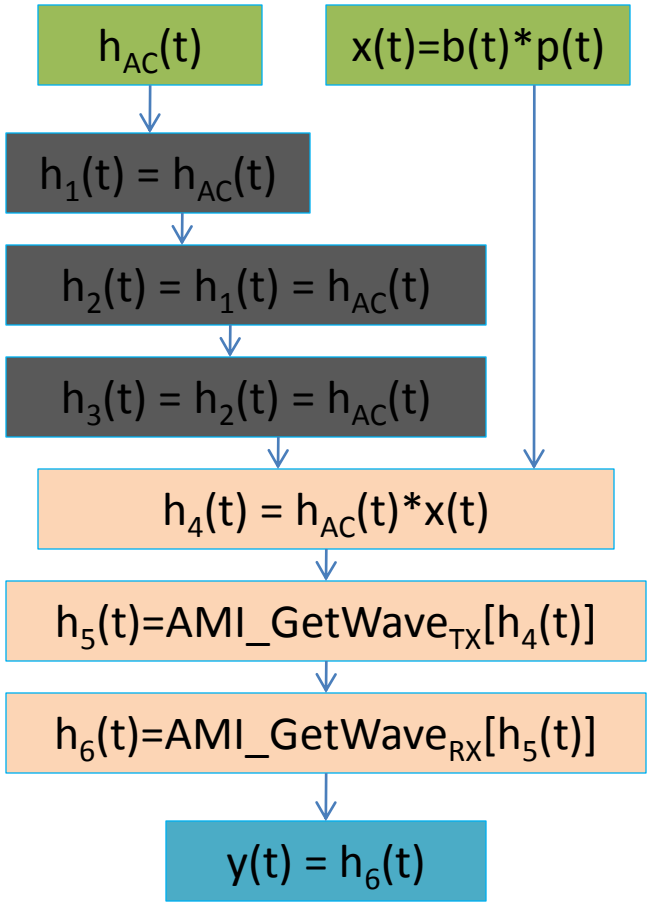


- Two-phased process. Output of Init phase convolves with stimulus to become the input of GetWave phase.
- Use_Init_Output, Init_Returns_Impulse and (Tx/Rx) GetWave_Exists parameters are defined to guide EDA tools on how to process data

Reference Flow Diagram – IBIS 5.0



- If Use_Init_Output = FALSE, Init phase is bypassed
- Convolving x(t) directly with $h_{AC}(t)$ without including the Tx AMI block (Step 4) makes this flow invalid for NLTV Tx AMI block



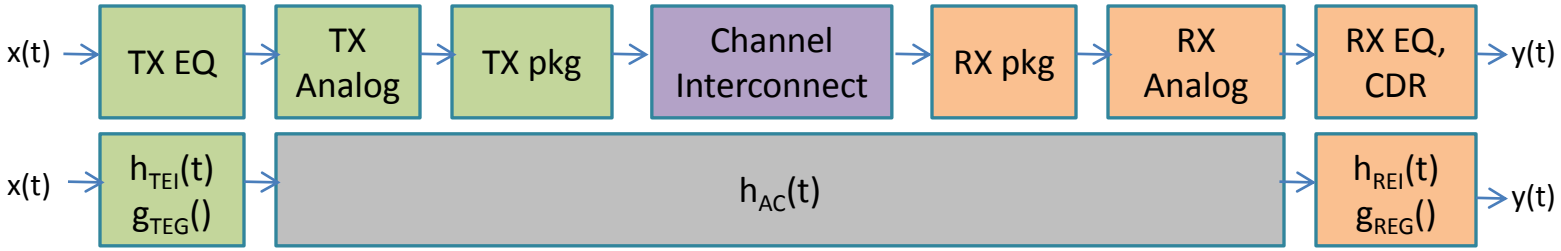
Observations

- $h_{AC}(t)$ is the cascaded LTI analog channel impulse response
- b(t)*p(t) is the input waveform to Tx AMI block
- It is not possible to map this flow to system equations relating output to input in a manner that is consistent with signal processing principles
- Use_Init_Output directs the EDA tool how to process AMI_Init output
- Init_Returns_Impulse indicates whether output AMI_Init is modified

What is Double-Counting

- Tx and Rx models both attempt to “automatically optimize” channel performance based on channel impulse response hAC(t). Without knowing what each other is doing, one can easily out-smart the other, causing the so-called double-counting phenomenon. A more accurate term is mis-counting because the mis-calculation can go both ways.
- The double-counting issue is also related to the fact that the physical meaning of input and output variables of AMI_GetWave calls are not clearly defined.
- Use_Init_Output was introduced to allow bypassing of AMI_Init function calls by directly convolving the analog channel with stimulus before calling AMI_GetWave functions.
- The reference flows become complicated when all combinations of Use_Init_Output, Init_Returns_Impulse and GetWave_Exists must be dealt with in a consistent manner.

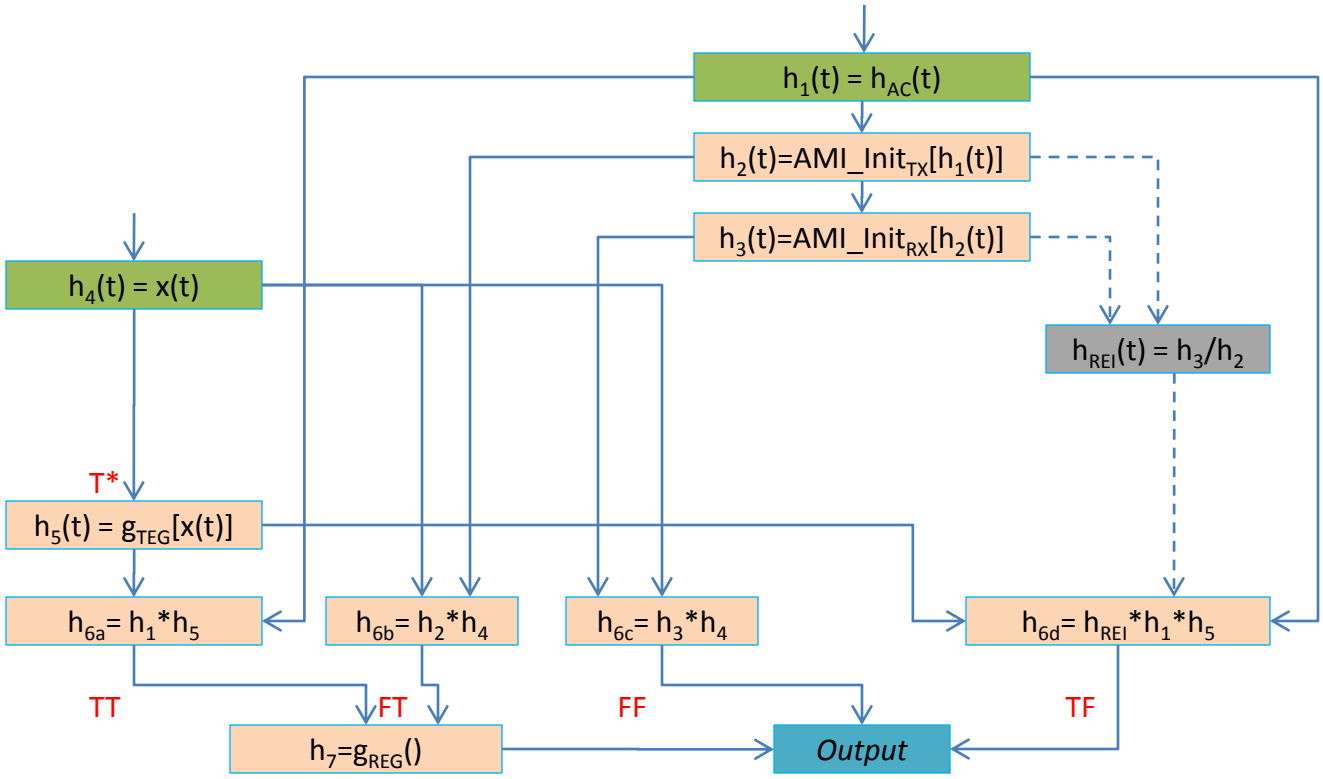
Reference Flow – IBIS 5.1



Step 1: $h_1(t) = h_{AC}(t)$
Step 2: $h_2(t) = Tx\_AMI\_Init[h_1(t)] = h_{TEI}(t)*h_{AC}(t)$
Step 3: $h_3(t) = Rx\_AMI\_Init[h_2(t)] = h_{REI}(t)*h_{TEI}(t)*h_{AC}(t)$
Step 4: $h_4(t) = x(t) = b(t)*p(t)$
Step 5: $h_5(t) = g_{TEG}[h_4(t)];$ (TxGE = TRUE)
Step 6a: $h_{6a}(t) = g_{REG}[h_1(t)*h_5(t)];$ (TxGE=TRUE;RxGE=TRUE)
Step 6b: $h_{6b}(t) = g_{REG}[h_2(t)*h_5(t)];$ (TxGE=FALSE;RxGE=TRUE)
Step 6c: $h_{6c}(t) = h_3(t)*h_4(t);$ (TxGE=FALSE;RxGE=FALSE)
Step 6d: $h_{6d}(t) = h_{REI}(t)*h_1(t)*h_5(t);$ (TxGE=TRUE;RxGE=FALSE)
Step 7: $h_{7a,b}(t) = g_{REG}[h_{6a,b}(t)];$
Step 8: $h_8(t) = \{h_{7a}(t), h_{7b}(t), h_{6c}(t), h_{6d}(t)\}$

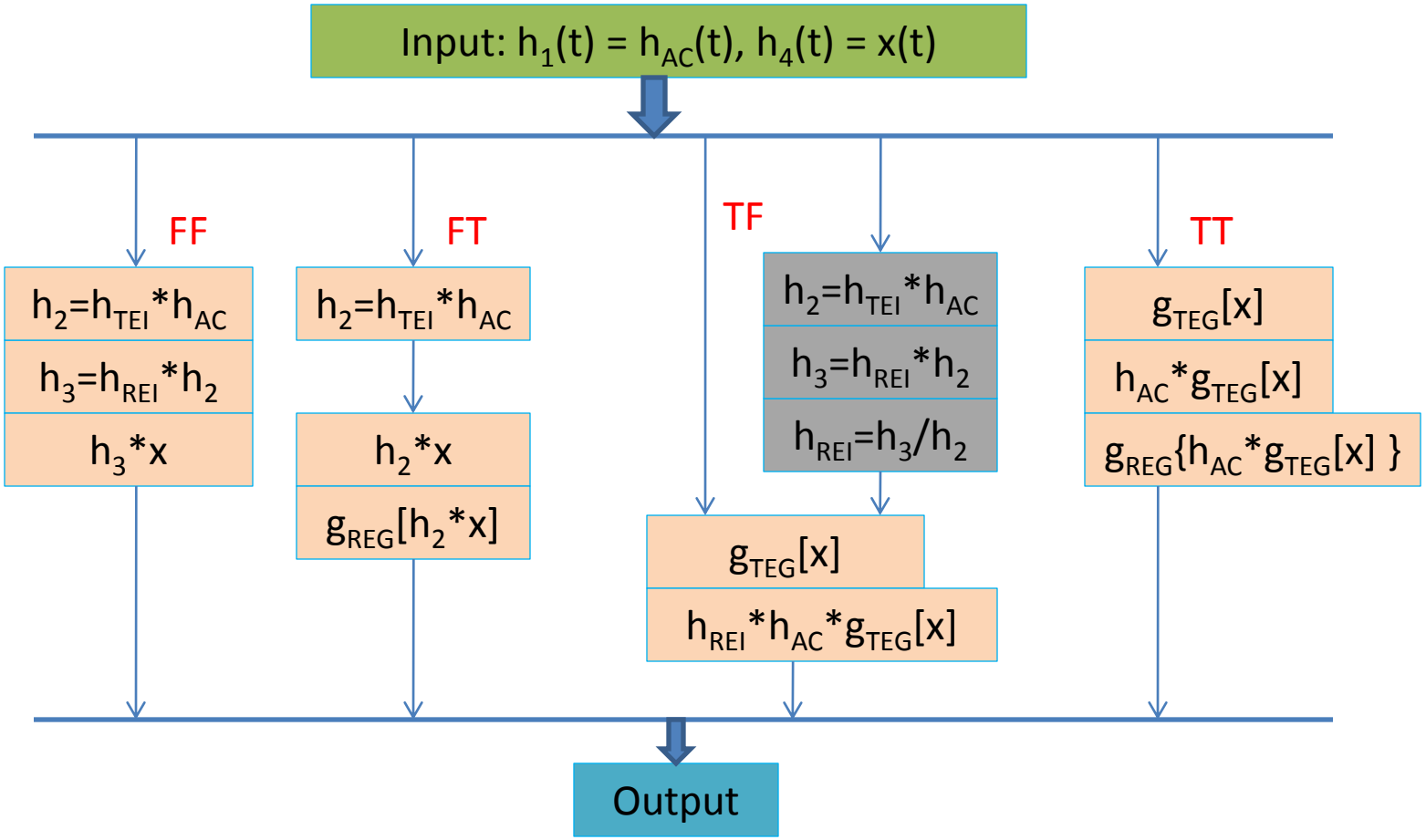
- [Note]: TxGE is TX GetWave_Exists; RxGE is RX GetWave_Exists

Reference Flow Diagram - Original



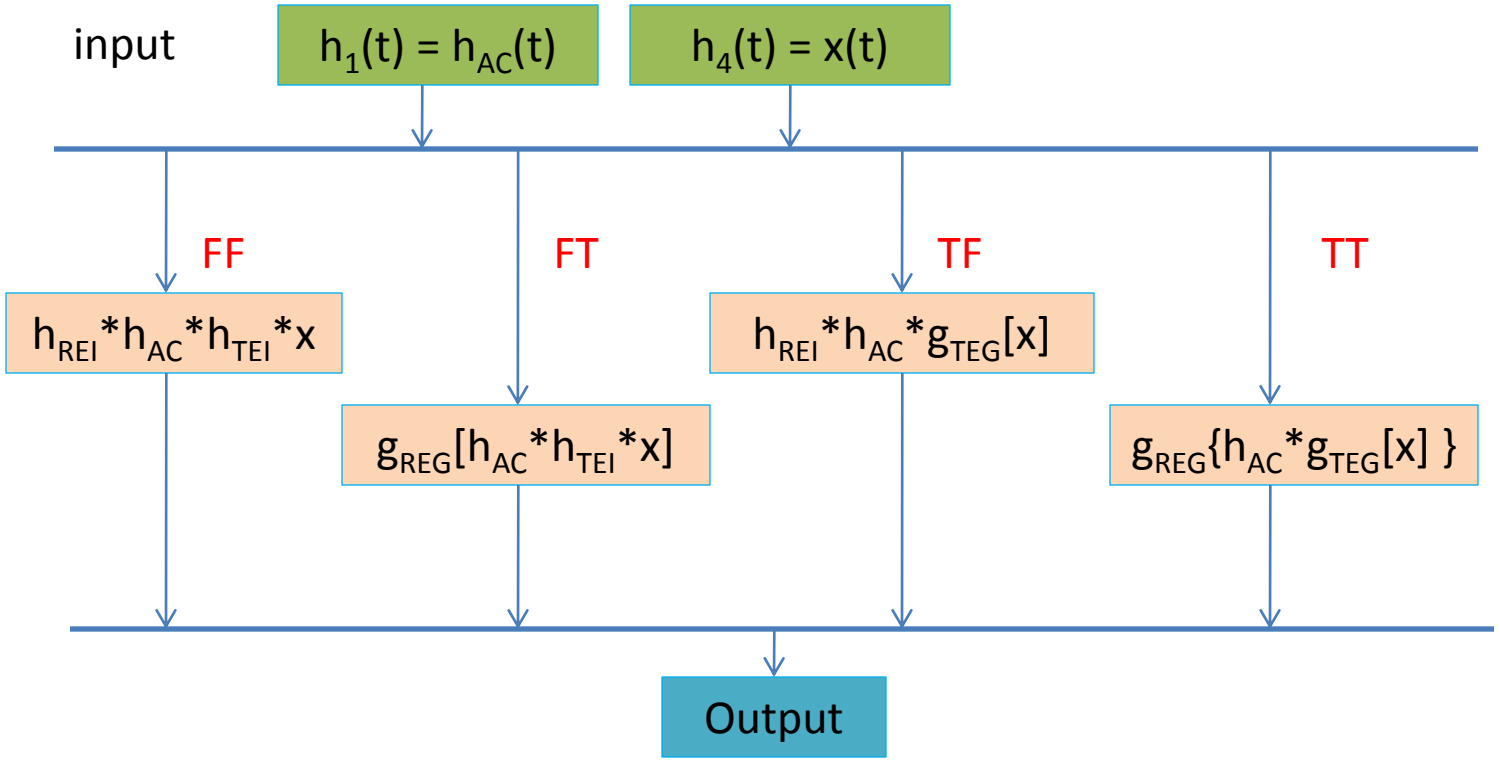
- Four possible combinations of Tx GetWave_Exists and Rx GetWave_Exists are: FF,FT,TF and TT

Reference Flow Diagram – Expanded



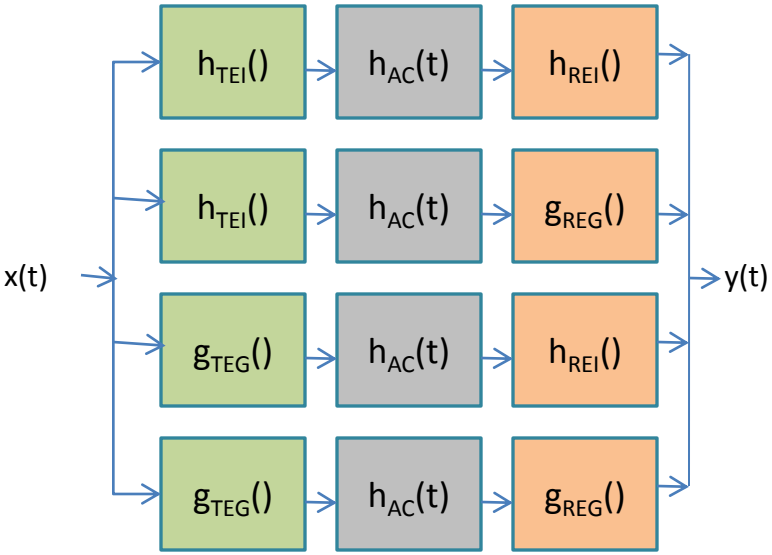
- This is equivalent to reference flow on previous page but easier to identify the four branches from start to finish

Reference Flow Diagram – Consolidated



- Each branch contains four components of: (1) stimulus; (2) Tx AMI; (3) analog channel; (4) Rx AMI

Block Diagram and Equations



FF: $y(t) = h_{REI}(t)*h_{AC}(t)*h_{TEI}(t)*x(t)$

FT: $y(t) = g_{REG}[h_{AC}(t)*h_{TEI}(t)*x(t)\ ]$

TF: $y(t) = h_{REI}(t)*h_{AC}(t)*g_{TEG}[x(t)]$

TT: $y(t) = g_{REG}[h_{AC}(t)*g_{TEG}[x(t)]]$

- Four possible cases of Tx and Rx AMI system with analog channel in between
  - [Tx GetWave_Exists, Rx GetWave_Exists] = {FF,FT,TF,TT}
- "Init_Returns_Impulse = False" is a trivial special case for "Tx AMI_Init = dirac delta function". In the generalized system equations, this special case is degenerate.

System Equation Expansion

TX Getwave _Exists	RX Getwave _Exsits	Case #	Equation	Step #
False	False	1	$y(t) = h_{REI}(t)*h_{AC}(t)*h_{TEI}(t)*x(t)$	1,2,3,4, 6c
False	True	2	$y(t) = g_{REG}[h_{AC}(t)*h_{TEI}(t)*x(t)]$	1,2, 4, 6b,7
True	False	3	$y(t) = h_{REI}(t)*h_{AC}(t)*g_{TEG}[x(t)]$	1, 4,5,6d[*]
True	True	4	$y(t) = g_{REG}[h_{AC}(t)*g_{TEG}[x(t)]]$	1, 4,5,6a,7

- Steps 1 and 4 obtain external input variables. They are the input nodes in the flow and are the only common denominators of all branches.
- Steps 2,3,5,6[abcd] and 7 can be consolidated into one step with four branches.
- [*] computation of $h_{REI}(t)$ requires $h_2(t)$ and $h_3(t)$

Init_Returns_Impulse

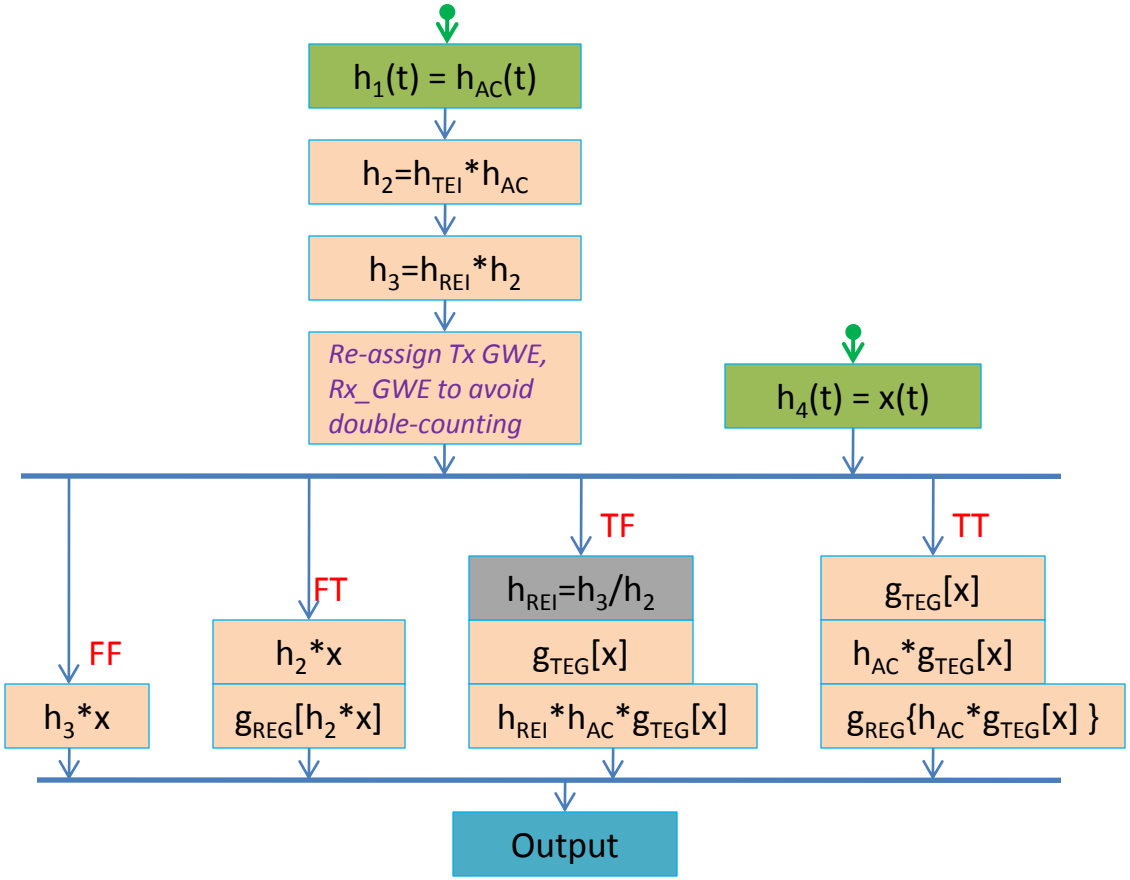
TX Init_ Returns_ Impulse	TX Getwave_ Exists	RX Init_ Returns_ Impulse	RX Getwave_ Exists	TX Selected Function	RX Selected Function	Case #	Comment
True	True	True	True	Getwave	Getwave	4	
True	True	True	False	Getwave	Init	3	
True	True	False	True	Getwave	Getwave	4	
True	False	True	True	Init	Getwave	2	
True	False	True	False	Init	Init	1	
True	False	False	True	Init	Getwave	2	
False	True	True	True	Getwave	Getwave	4	
False	True	True	False	Getwave	Init	3	
False	True	False	True	Getwave	Getwave	4	

- Out of 16 possible combinations, 9 valid combinations of TX and RX Init_Returns_Impulse and Getwave_Exists are mapped to four unique cases of TX and RX function call combinations [1-6]

Double Counting

- IBIS 5.1: "when the Tx AMI model contains an AMI_GetWave function that performs a similar or better equalization than the Tx AMI_Init function, there is a possibility for "double-counting" the equalization effects in the Tx executable model file. To allow for such models to work correctly, the EDA tool can operate in one of several ways, two of which are documented here:
  - not utilize the Tx AMI_GetWave functionality, by treating the Tx AMI model as if the Tx GetWave_Exists was False."
- The root cause of "double counting" is to allow the models to operate independently and freely beyond the true behaviors of the silicon chips they are supposed to represent.
- serdes transceiver ASICs never double count. (there could be improper settings of serdes transceiver ASICs)

Contention of Intentions



- The criteria for reassigning TX GetWave_Exists is neither defined in the spec nor logically conceivable. This practice is highly questionable both in theory and practice.

Tx Auto Optimization

- Tx and Rx AMI functions are allowed to “optimize” its tap coefficients and other parameters based on channel interconnect impulse response $h_{AC}(t)$
- The assumption that Tx and Rx could both independently optimize to desired performance without loopback and still maintaining correlation with silicon is highly questionable for following reasons

Tx Auto Optimization (cont.)

1. Tx silicon is not capable of doing this auto optimization
2. It deprived users the opportunity to manually set Tx tap coefficients and other Tx/Rx parameters, which is a very important feature of channel simulation
3. It is logically inconsistent and is fundamentally different from how Rx silicon optimize its parameters

Proposed Solution for Double Counting

- Models should be true to the silicon they represent. In case models attempt to out-smart the silicon by running algorithms that is different from the true behavior of the silicon, it should only be done under explicit instructions
- Rx models could send instructions/info to Tx through the backchannel, however Rx models (or EDA tools/users) should not be allowed to dictate the Tx output *a priori*; Rx should only process the Tx output *a posteriori*.
- Since most silicon accepts fixed EQ settings, models should be able to do the same. Automatic optimization should be optional rather than mandatory.

Observations

- The time-domain reference flow has only four unique cases, controlled by Tx and Rx GetWave_Exists
- Init_Returns_Impulse is informational only and does not impact the workflow. In the nine valid cases involving Init_Returns_Impulse , five of them are duplicates
- Flow can be mapped to system equations from input to output for each block
- Use_Init_Output was deprecated; AMI_GetWave is always called if GetWave_Exists = TRUE
- x(t) and $h_{AC}(t)$ are only processed once by AMI_Init or AMI_GetWave, systematically eliminating the double counting issue.
- The same reference flow applies to both LTI and NLTV AMI blocks.

Init_Returns_Impulse

- Init_Returns_Impulse does not participate in the reference flow branching decisions.
- After the deprecation of Use_Init_Output, the new flow always calls AMI_Getwave whenever Getwave_Exists = true, regardless of the value of Init_Returns_Impulse
- Outputs are generated by AMI_Init only if Getwave_Exists = false and in this case, the flow always calls AMI_Init regardless of the value of Init_Returns_Impulse
- Clarification is needed on the intended purpose, application, interpretation of Init_Returns_Impulse, and its role in the flow.

AMI_Init

- If Init_Returns_Impulse = TRUE, AMI_Init returns the convolution of input impulse response with impulse response of the equalization
- If Init_Returns_Impulse = FALSE, AMI_Init passes the input to output without changing it
  - the AMI block represents an all pass filter which impulse response is the Dirac delta function with unit amplitude.
- The output can always be interpreted as the convolution of the input with the impulse responses of the AMI block.

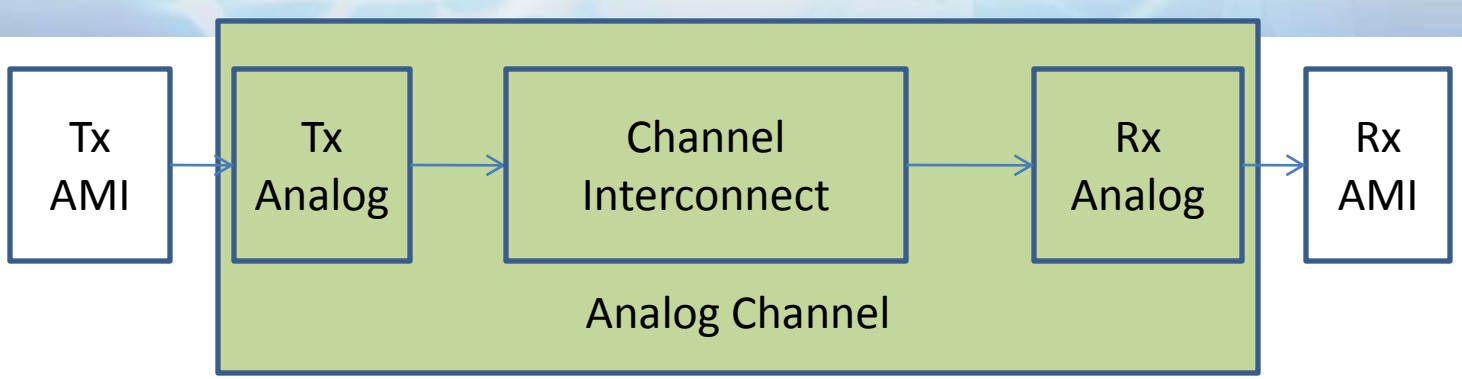
AMI_GetWave

- Only applies to time-domain flow; does not apply to statistical flow
- Can represent either NLTV or LTI AMI blocks
- In reference flow, AMI_GetWave always has higher precedence than AMI_Init
- Explicit relationship between output and input may not exist

Conclusion on Reference Workflow

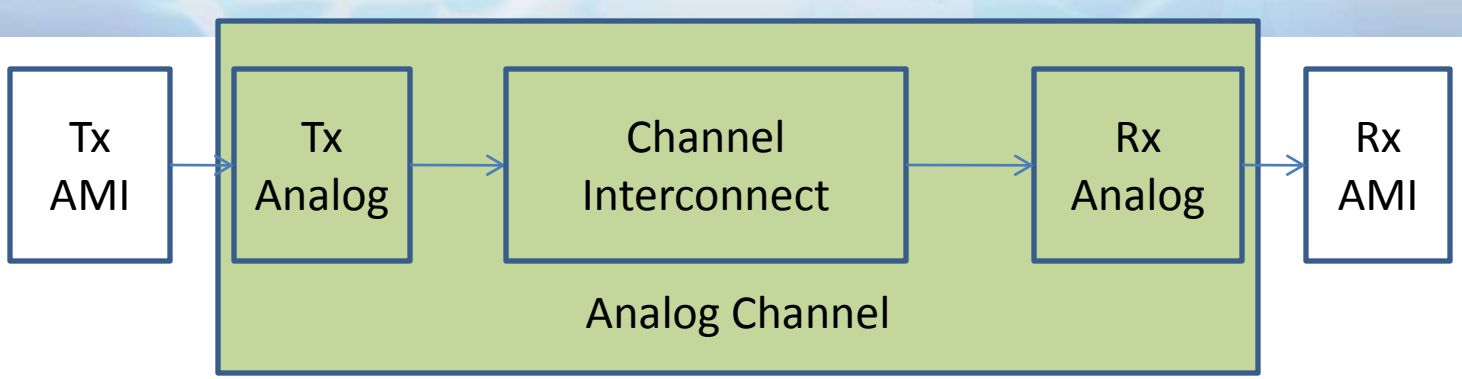
- IBIS 5.1 time-domain reference flow is a one-pass flow where only one of _Init or _GetWave is deployed at simulation time. This approach allows the flow to be mapped to system equations consistent with well-established signal processing principles.
- Double-counting remained an unresolved issue in 5.1 due to the fact that Tx and Rx are allowed to “optimize” independently
- Deprecation of Use_Init_Output simplified the workflow without comprising capability.
- Init_Returns_Impulse is a trivial case of degeneration. Its impact to flow and result is insignificant.

Introduction of Analog Modeling



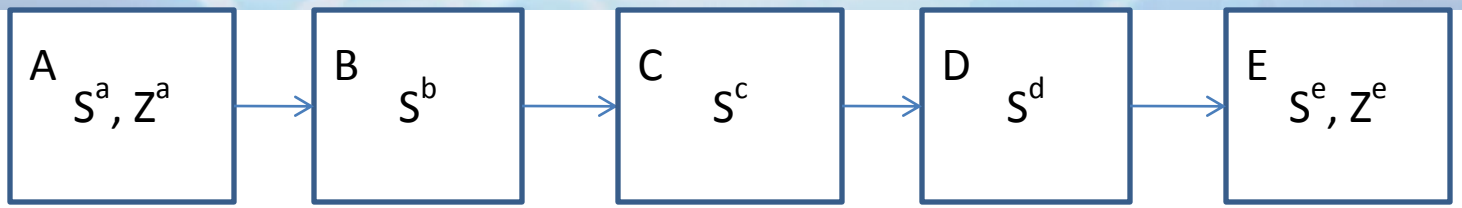
- IBIS 5.1 AMI channel model is shown above.
- All analog blocks are cascaded to form the analog channel with impulse response hAC(t)
- There are ambiguities in the impedance conditions at the input and output of analog channel
- This presentation is intended to explain and clarify the issue of interface impedance conditions between the Tx/Rx AMI block and analog channels

What is in IBIS 5.1?



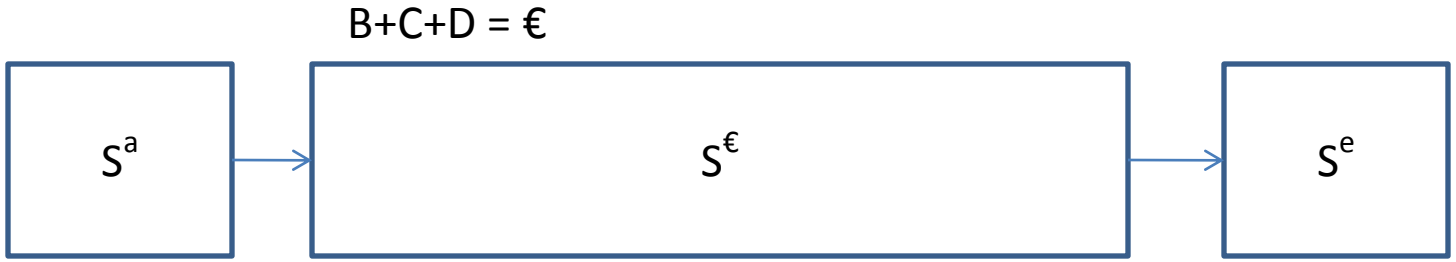
- "The transmitter equalization, receiver equalization and clock recovery circuits are assumed to have a high-impedance (electrically isolated) connection to the analog portion of the channel"
- At the interface of core (i.e. Tx AMI block) and analog circuit (i.e. Tx analog), there are two impedances, one is the output impedance of the AMI block and the other is the input of the analog clock. The wording in the spec did not say which is which. Are they both high impedance or only one of them is high impedance, and which one?

Network Cascade Problem



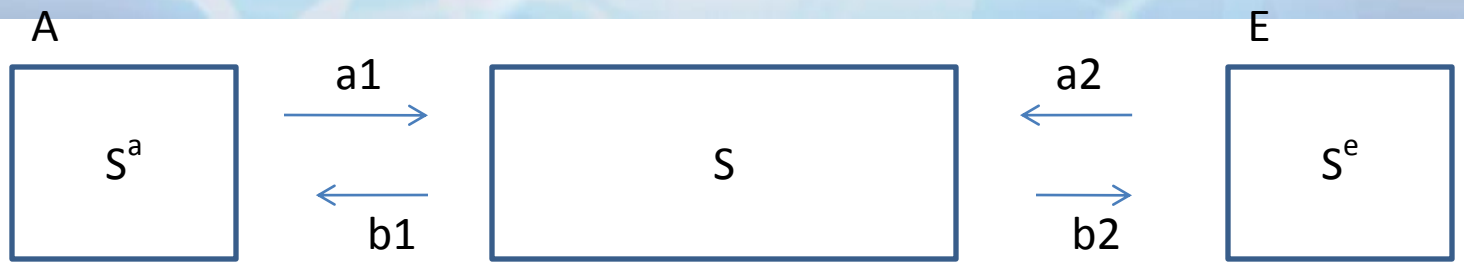
- The network has five blocks of A,B,C,D and E cascaded in sequence.
- Block A is the signal source which could be non-linear time-variant (NLTV). Its output impedance is assumed LTI and provided in S-matrix S^a
- Block A generates the waveform $v^a_{out}(t)$ S^a
- Block E is the signal sink which could be NLTV. Its input impedance is assumed LTI and provided in S-matrix S^e
- B, C and D are LTI networks and can be represented by S-matrices of S^b,S^c,S^d
- In this context, no restrictions or assumptions were made whatsoever on the ranges and values of the S-parameters. All the interfaces (A-B, B-C, C-D, D-E) are generally and arbitrarily MIS-matched.
- We wish to obtain the voltage at the input of block E, $v^e_{in}(t)$, taking into account of all the reflections/mismatches at all interfaces, rigorously (i.e., free of any assumptions that can cause systematic errors)

Step 1: cascade B,C,D into €



- cascade B,C,D into a single network €, the cascade sub-network B+C+D has S-matrix $S^€$
- The cascade formula are trivial exercises of textbook problems and are not listed here

Step 2: frequency domain equations



- By definition of S-parameter, we have

  $a_1 = S^a b_1 + v^a_{out}$

  $a_2 = S^e b_2$

  $b_1 = S_{11} a_1 + S_{12} a_2$

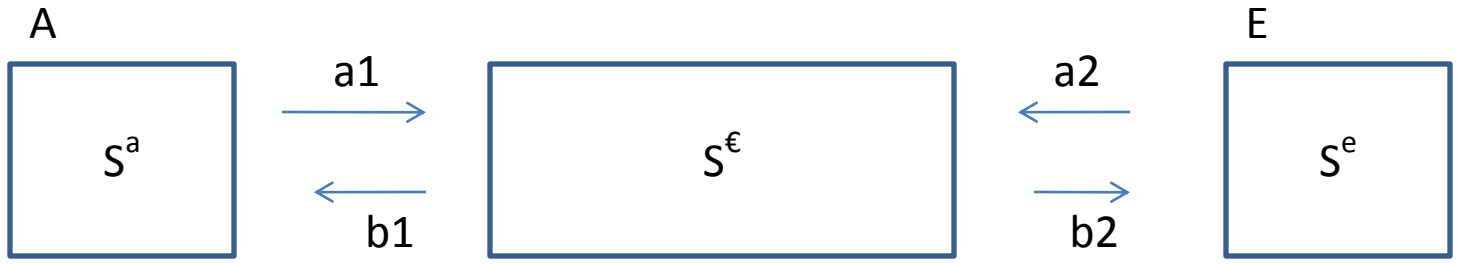
  $b_2 = S_{21} a_1 + S_{22} a_2$

- Solve for b_2, a_2

  $b_2 = \hat{S} \cdot v^a_{out}$, $a_2 = S^e \hat{S} \cdot v^a_{out}$

  $\hat{S} = [S_{22} S^a/(1-S_{11} S^a) + S_{22}]/[1-S_{22} S^e - S_{22} S^a S_{12} S^e/(1-S_{11} S^a)]$

Step 3: time domain equations



- Input voltage at block E is,

  $v^e_{in} = H(j\omega) \cdot v^a_{out}$

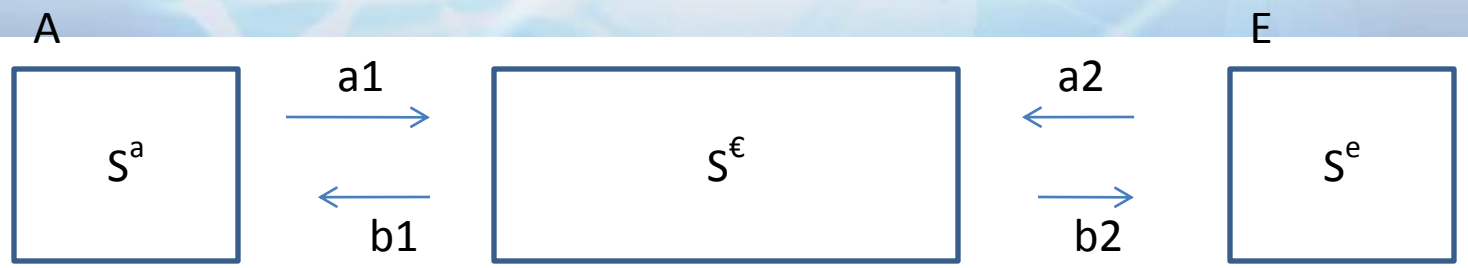
  where $H(j\omega) = (1+S^e)\,\hat{S}$

- Its time domain correspondence is,

  $v^e_{in}(t) = h(t) * v^a_{out}(t)$

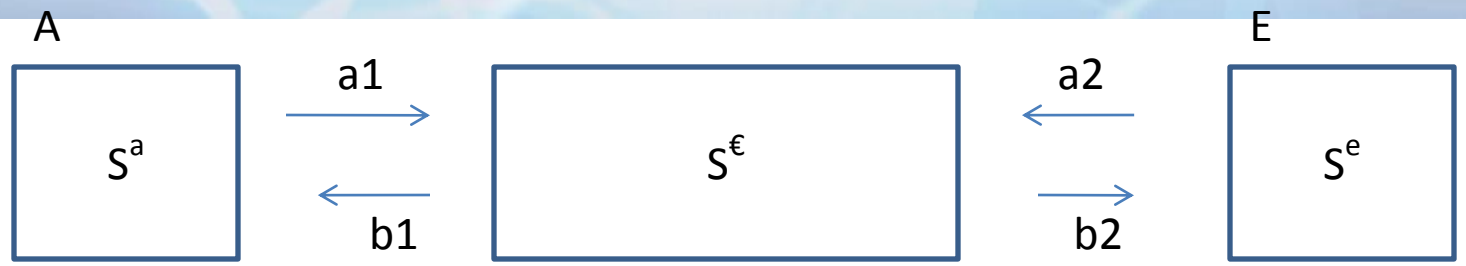
  where h(t) is the channel impulse response, $v^a_{out}(t)$ is the output waveform of the signal source (i.e. input to the channel) and, $v^e_{in}(t)$ is the input voltage at block E (i.e. output of the channel)

Conclusion



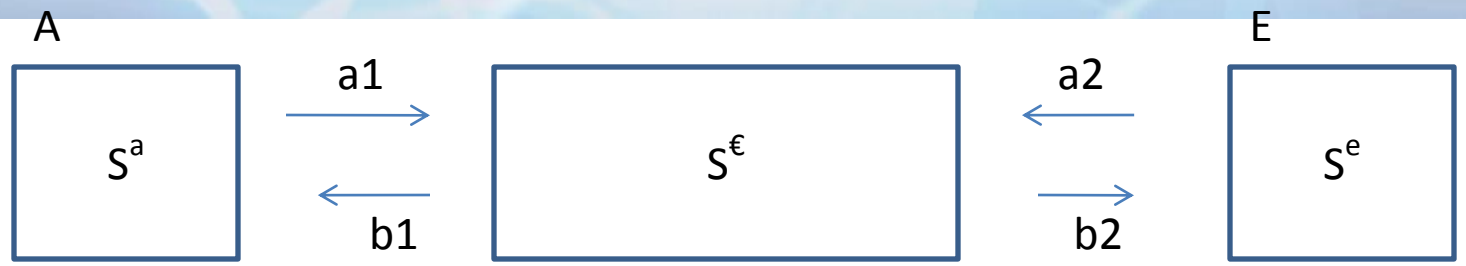
- The solution of the general network cascade problem takes into account all the mismatches in the system
- No assumptions are made on the values or ranges of any of the S-parameters
- Solution formula are straightforward and widely available from textbooks on network theory

IBIS 5.1 AMI



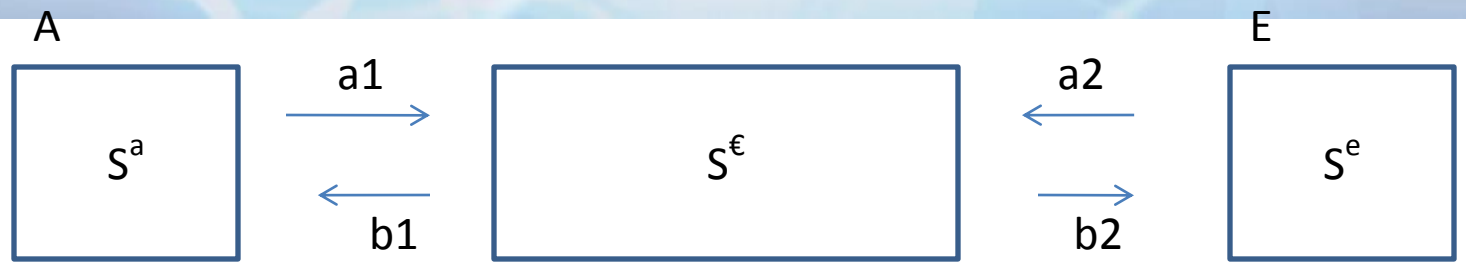
- IBIS 5.1 AMI 5.1 is a special case of the general network cascade problem by assuming that the source, sink and channel have ideal impedance values
  1) Signal source has zero output impedance ($S^a = -1$)
  2) Analog channel is perfectly matched at input ($S^b_{11} = 0$)
  3) Analog channel is reverse isolated ($S^b_{12} = 0$)
  4) Analog channel is perfectly matched at output ($S^d_{22} = 0$)
  5) Signal sink has infinite input impedance ($S^e = 1$)
     - some implementation set this to 50 ohm either by design or by accident ($S^e = 0$)

IBIS 5.1 AMI, Pros and Cons



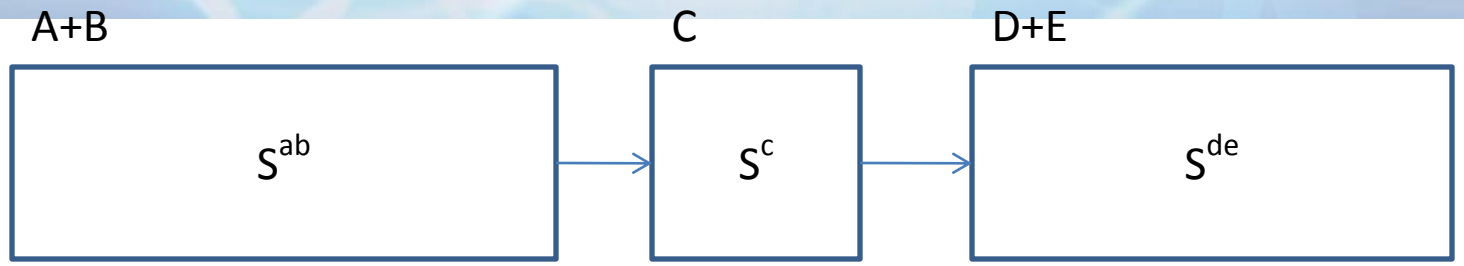
- The impedance conditions imposed by IBIS 5.1 AMI is a special ideal case of the general network cascade problem
- There has been cases where these ideal impedance conditions were mis-interpreted by either the model creator or the EDA tool, resulting in erroneous results.
- The five ideal impedance conditions maybe too restrictive for some model makers
- They could be confusing for some users

Recommendations



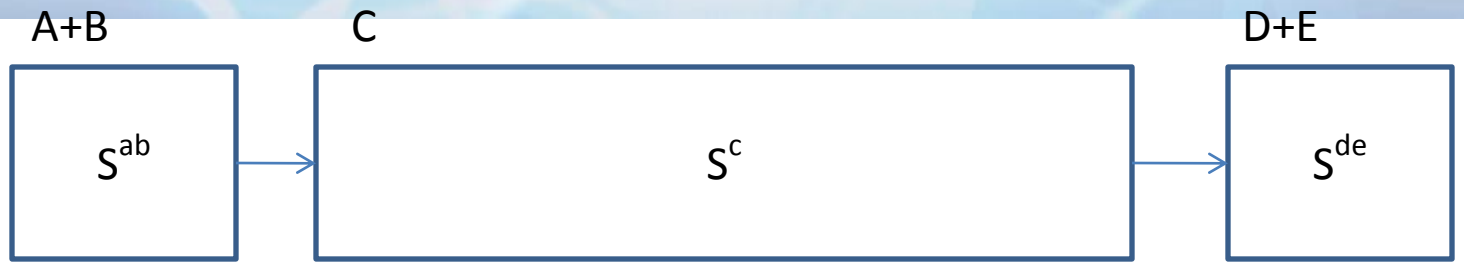
- IBIS 5.1 can be enhanced by removing the five ideal impedance conditions. They don't really offer much benefit, at the cost of restrictions to model creators and confusions to EDA tool users.
- The general network cascade problem is straightforward to implement. It provides all the freedom to model makers to model the circuits as they really are (real world signal sources and sinks have non-ideal impedances)
- The requirement to provide source and sink impedances (S^a and S^e) does not add extra burdens to the model makers. These data are already available in existing models. They actually reduce the burdens of model makers because the non-ideal network data do not need to be fitted to meet the five impedance conditions.
- The proposed enhancement does not deprecate or invalidate the five ideal impedance conditions in IBIS 5.1 AMI (i.e. they can still be used in models if desirable to some model makers)

Different ways to cascade



- The solution can be obtained by cascading the network in a different sequence; A and B are cascaded, D and E are cascaded to form the system shown above.
- It is obvious that this network has the exact same structure as the one shown in Step 1 (cascading B, C, D)
- It can be solved using the exact same process as before

Conclusion



- The network cascade problem can be solved by different sequences of cascading. The final answer should and will always be the same if done properly by using the same formula.
- By lifting the five impedance conditions in IBIS 5.1 AMI, the proposed scheme allows the model makers to put any circuits inside the signal source (aka AMI block) without changing the work flow.
- From a modeling perspective, it is irrelevant to debate whether the circuit inside the signal source (aka DLL block) is digital or analog, because it does not really matters (again, from a modeling perspective)