IBIS-AMI Time-Domain Reference Flow

–James Zhou, QLogic Corp.
Presented at IBIS Quality Task Group
Aug. 16, 2011

Background and Objectives

- The subject of IBIS-AMI cookbook was raised by Mike LaBonte, Chairman of IBIS Quality Task Group, during the Aug 9, 2011 teleconference.
- The AR was to create a "starter" presentation summarizing the current status of IBIS-AMI reference flows and modeling approaches for the purpose of exploring end-user interests and concerns on IBIS-AMI.
- With the end goal being to create materials for end-user education and training, the feedbacks and comments generated during this process may also help to identify issues in the Specification requiring clarification or modification.
- This presentation only covers IBIS-AMI time-domain reference flow.

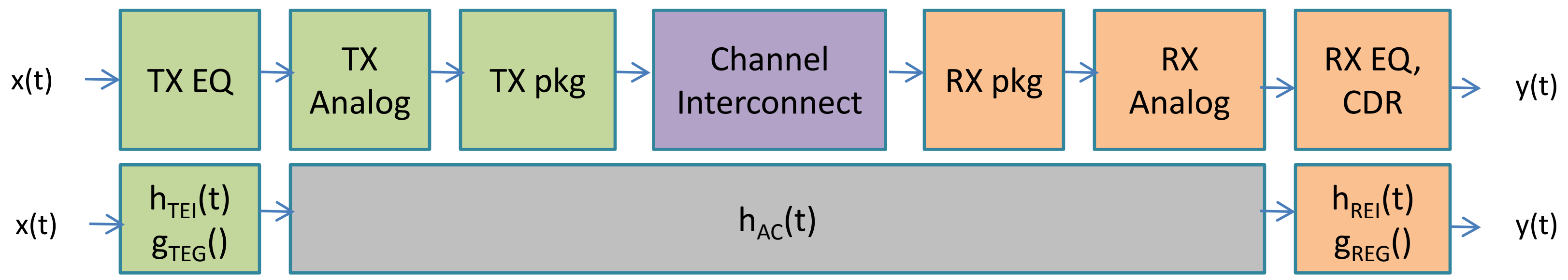
Beginning Questions

- Some “typical” questions that an end-user might ask about IBIS-AMI modeling:
  - Locations of AMI and analog data
  - Navigation of IBIS-AMI analysis flows
  - Engineering and mathematical (vs. programmatic) definitions of IBIS-AMI functions (AMI_Init, AMI_Getwave) and quantities
  - Assumptions made in the IBIS-AMI models and analysis flow
  - Mandatory vs. optional features, parameters and operations

AMI Reference Flow – Brief History

- BIRD 104.1, (10/2007)
  - First public proposal 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

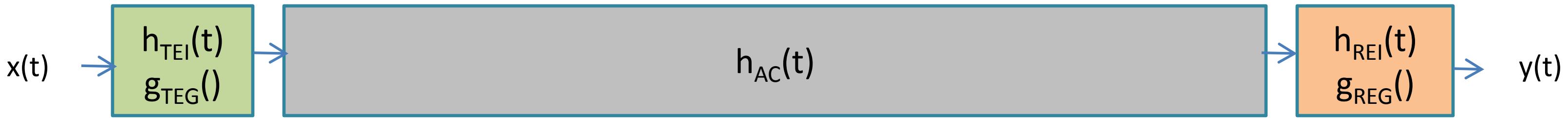
Reference Flow – IBIS 5.0



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

- $h_{AC}(t)$ is the end-to-end analog channel impulse response
- $b(t)*p(t)$ is the input waveform to Tx AMI block
- [Note] Naming conventions for impulse and AMI_GetWave functions in this presentation follow that of DAC 2009 IBIS Summit Presentation by W. Katz

Reference Flow – IBIS 5.0



Step 5a: $h_{5a}(t) = \text{AMI\_GetWaveTX}[h_4(t)]$	(Tx GetWave_Exists = TRUE)
Step 5b: $h_{5b}(t) = h_4(t)$	(Tx GetWave_Exists = FALSE)
Step 6a: $h_{6a}(t) = \text{AMI\_GetWaveRX}[h_5(t)]$	(Rx GetWave_Exists = TRUE)
Step 6b: $h_{6b}(t) = h_5(t)$	(Rx GetWave_Exists = FALSE)
Step 7: $y(t) = h_6(t)$	

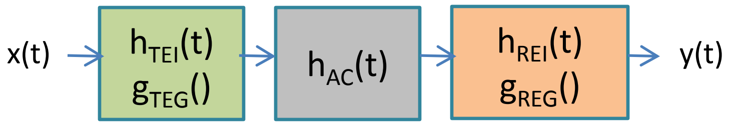
- It is not obvious how to map this process to system equations relating output to input
- Init_Returns_Impulse plays no role in the flow

Reference Flow Diagram – IBIS 5.0

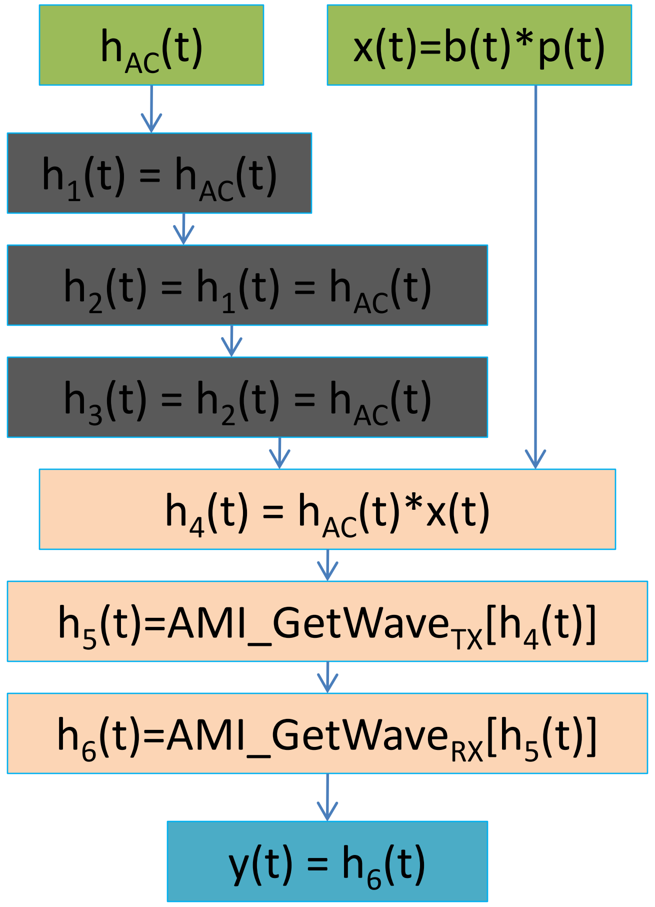
- Two-phased process. Output of Init phase convolves with stimulus to become the input of GetWave phase.
- Analog channel $h_{AC}(t)$ participates in both Init phase and GetWave phase.
- Use_Init_Output and Tx/Rx GetWave_Exists branchs are contained within local steps

$$\begin{tikzcd}
\boxed{h_{AC}(t)} \arrow[d] & \boxed{x(t)=b(t)*p(t)} \arrow[dddd, start anchor=south, end anchor=north east] \\
\boxed{h_1(t) = h_{AC}(t)} \arrow[d] & \\
\boxed{h_2(t)=AMI\_Init_{TX}[h_1(t)]} \arrow[d] & \\
\boxed{h_3(t)=AMI\_Init_{RX}[h_2(t)]} \arrow[d] & \\
\boxed{h_4(t) = h_3(t)*x(t)} \arrow[d] & \\
\boxed{h_5(t)=AMI\_GetWave_{TX}[h_4(t)]} \arrow[d] & \\
\boxed{h_6(t)=AMI\_GetWave_{RX}[h_5(t)]} \arrow[d] & \\
\boxed{y(t) = h_6(t)} &
\end{tikzcd}$$

Reference Flow Diagram – IBIS 5.0



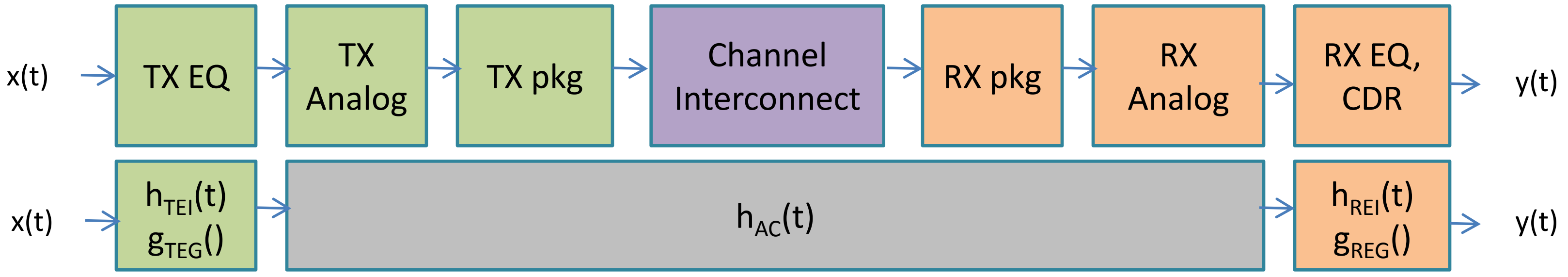
- If Use_Init_Output = FALSE, Init phase is bypassed
- Convolving x(t) directly with $h_{AC}(t)$ without including the Tx AMI block (Step 4) makes this flow invalid for NLTV Tx AMI block


$h_{AC}(t)$
$x(t)=b(t)*p(t)$
$h_1(t) = h_{AC}(t)$
$h_2(t) = h_1(t) = h_{AC}(t)$
$h_3(t) = h_2(t) = h_{AC}(t)$
$h_4(t) = h_{AC}(t)*x(t)$
$h_5(t)=AMI\_GetWave_{TX}[h_4(t)]$
$h_6(t)=AMI\_GetWave_{RX}[h_5(t)]$
$y(t) = h_6(t)$

Double-Counting

- The double-counting issue was originated from the fact that the output of the init phase serves as input to the getwave phase sequentially. There is some ambiguity on the definition and functionality of input and output variables of AMI_GetWave calls.
- 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, GetWave_Exists must be dealt with in a consistent manner.

Reference Flow - BIRD 120.1



Step 1: $h_1(t) = h_{AC}(t)$

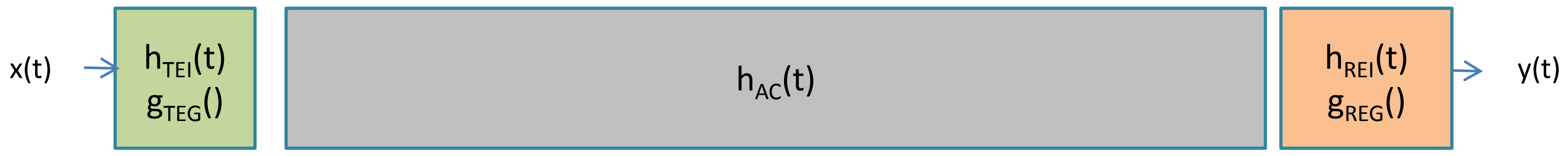
Step 2: $h_2(t) = \text{Tx\_AMI\_Init}[h_1(t)] = h_{TEI}(t)*h_{AC}(t)$

Step 3: $h_3(t) = \text{Rx\_AMI\_Init}[h_2(t)] = h_{REI}(t)*h_{TEI}(t)*h_{AC}(t)$

Step 4: $h_4(t) = x(t) = b(t)*p(t)$

- Showing time-domain flow only.

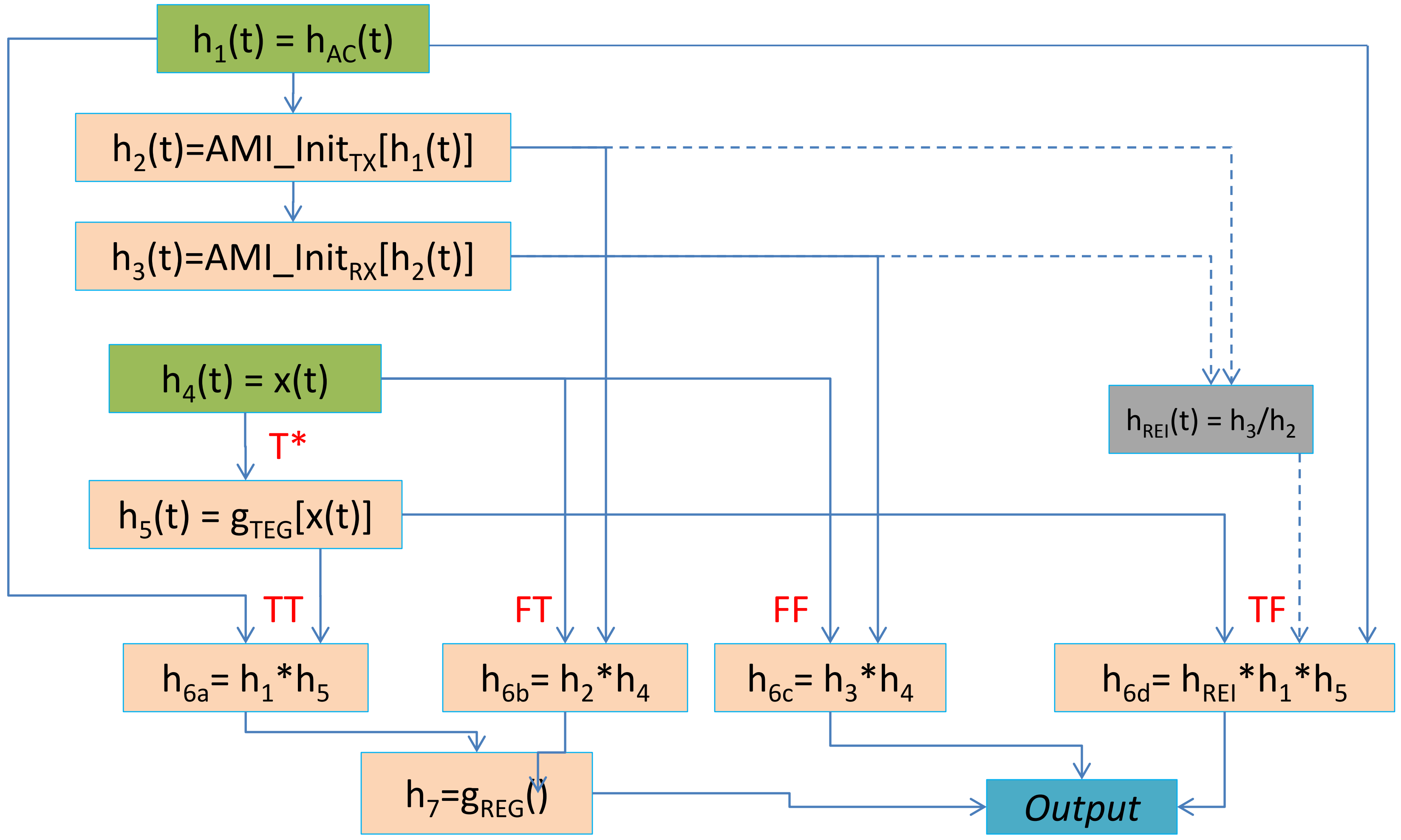
Reference Flow - BIRD 120.1 (cont.)



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:	$h_8(t) = \{h_{7a}(t), h_{7b}(t), h_{6c}(t), h_{6d}(t)\}$	

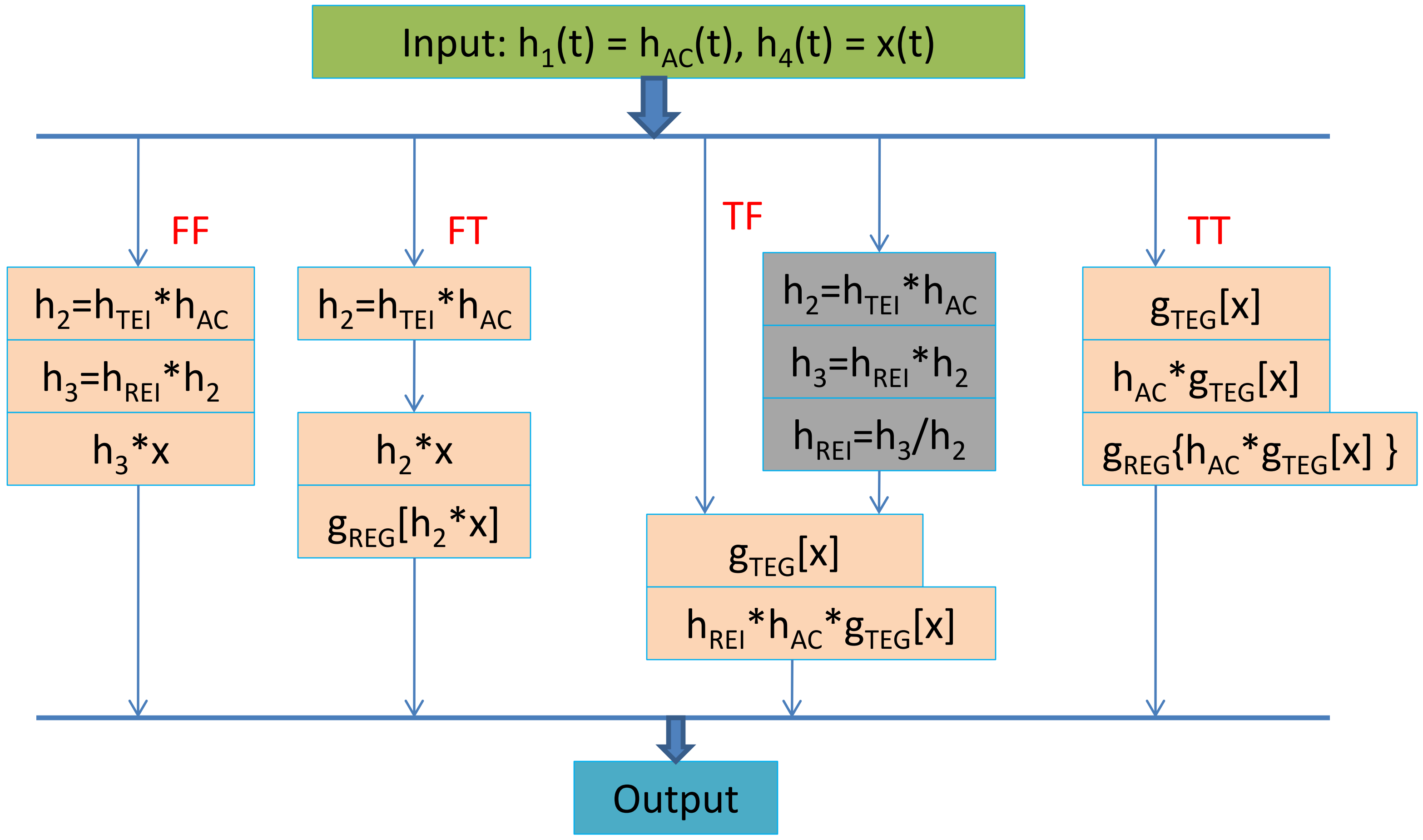
- [Note]: TxGE is TX GetWave_Exists; RxGE is RX GetWave_Exists

Reference Flow Diagram - Original



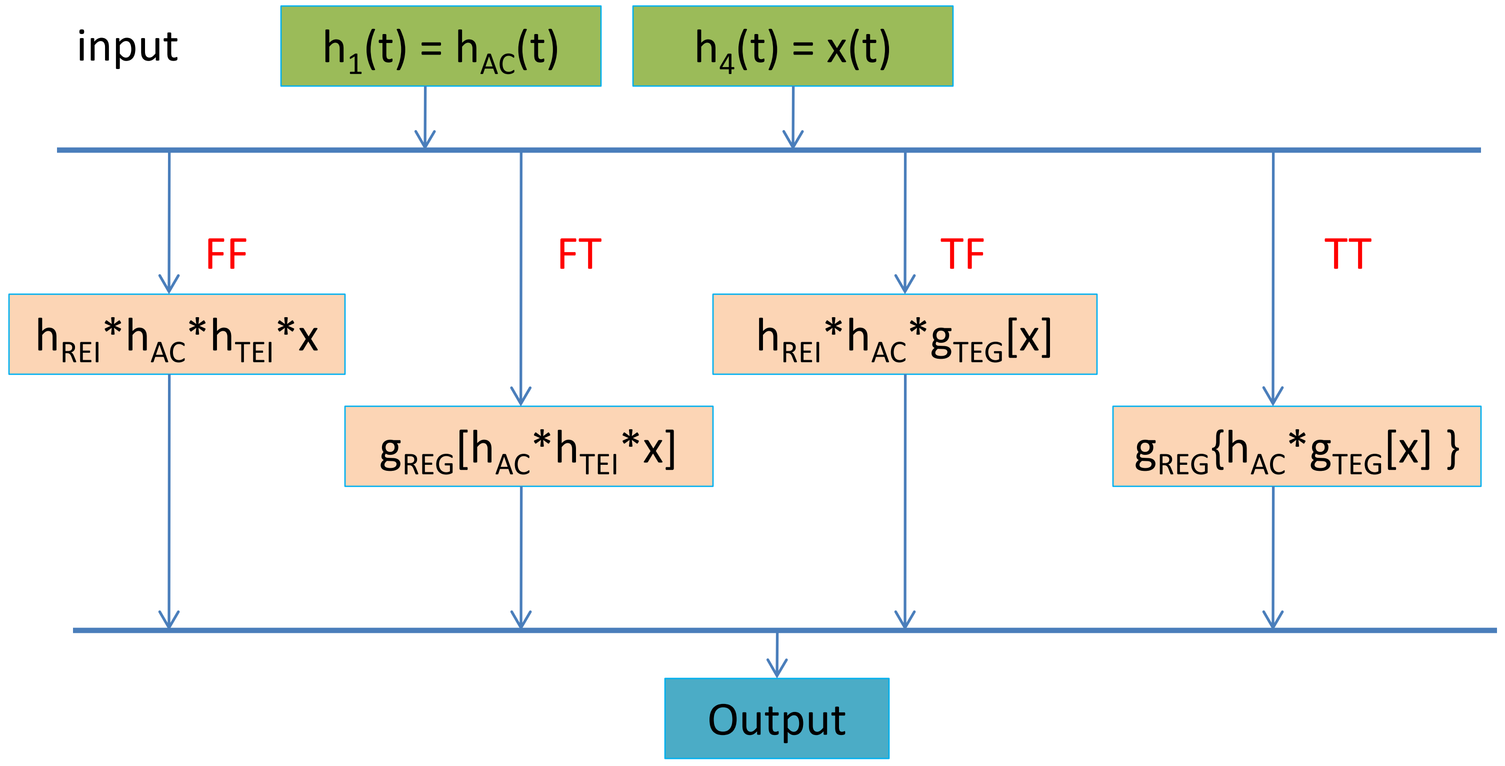
- Four possible combinations of Tx GetWave_Exists and Rx GetWave_Exists are: FF,FT,TF and TT

Reference Flow – Expanded



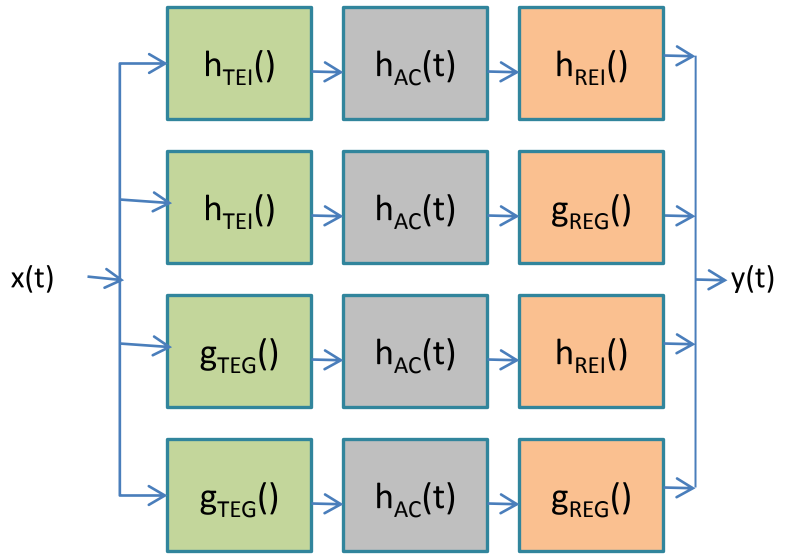
- This is equivalent to reference flow on previous page but easier to identify the four branches from start to finish

Reference Flow – Consolidated


input
$h_1(t) = h_{AC}(t)$
$h_4(t) = x(t)$
FF
FT
TF
TT
$h_{REI}*h_{AC}*h_{TEI}*x$
$g_{REG}[h_{AC}*h_{TEI}*x]$
$h_{REI}*h_{AC}*g_{TEG}[x]$
$g_{REG}\{h_{AC}*g_{TEG}[x]\}$
Output

- 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}

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 $h_3(t)$

Observations

- Four branch uni-phase flow
- 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 branch selection process.
- 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

Double-Counting

- “there is a possibility for 'double-counting' the equalization effects in the Tx model. 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.”
- Clarification is needed on this statement ‘s consistency with the reference flow
  - Reference flow always selects AMI_GetWave when GetWave_Exists=TRUE
  - If this intended to be an exception to the reference flow, it needs to be stated in the Specification

Conclusion

- BIRD 120.1 time-domain reference flow effectively resolved the double-counting issue in Specification 5.0
- Deprecation of Use_Init_Output simplified the workflow without comprising capability.
- Reference flow is presented in an equivalent way which might be logically easier to follow
- Inconsistencies exist between statements in BIRD 120.1 on double-counting and the reference flow
- The purpose and role of Init_Returns_Impulse in reference flow need to be clarified