Symmetric Processing of IBIS-AMI Redrivers & Handling of AMI_Init only Models

Alaeddin A. Aydiner

Errata & Comments Since Last Time

- (Thanks to Curtis!) When redriver TX is in PRE and BOTH modes taking the "past" into account, explicit convolutions by EDA tool with previous equalized impulse responses should not be necessary. Presumably, redriver TX would handle them as it is passed both its equalized upstream and unequalized downstream.
- A reason why passing only an impulse to redriver TX AMI may be too restricting: It is unlikely for the mostly LTI TX AMI EQ; but redriver TX AMIs may want to select an LTI impulse out of a table of LTI responses to approximate their nonlinearity via AMI_Init calls.

Legal Disclaimer

Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

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.

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.

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.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

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.

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

Intel, Intel Interconnect Model Analyzer and Domain Converter (Intel IMADC), Intel Omni-Path Channel (Intel OP Channel), Intel Omni-Path Technology (Intel OP Technology), Intel Omni-Path Host Fabric Interface (Intel OP HFI), Intel Omni-Path Architecture (Intel OPA), 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.

Outline

Review of TX to RX Signaling

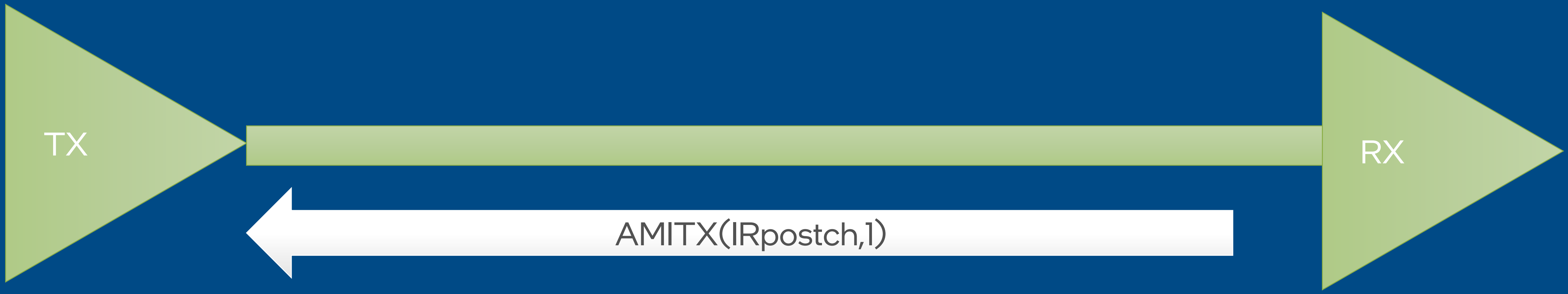
AMI_Init of Redrivers under Transient Signaling

Missing AMI_GetWave under Transient Signaling

TX to RX

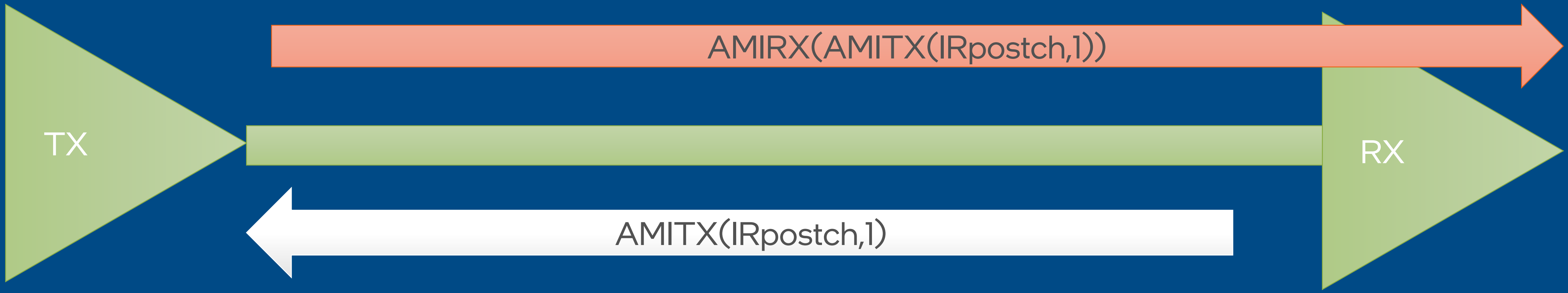
- Beginning with a channel connection from TX to RX…

TX to RX



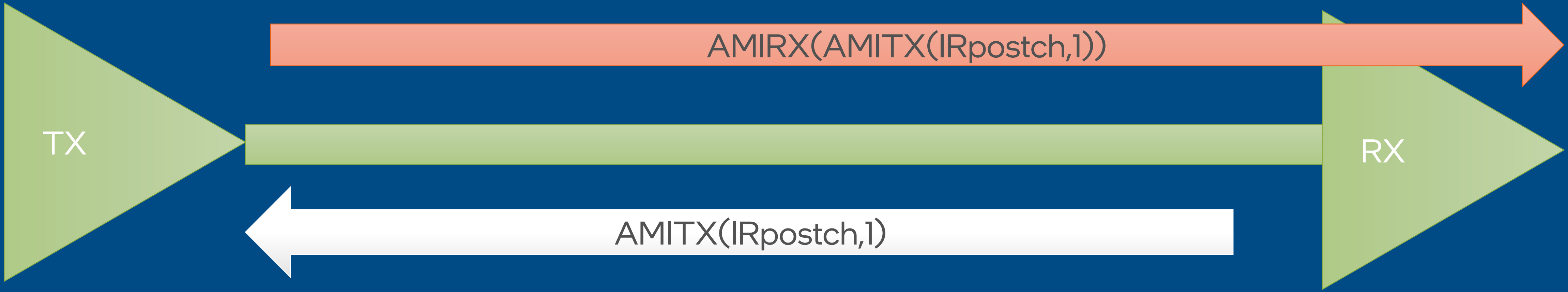
- TX AMI_Init takes its unequalized "post-"channel.

TX to RX



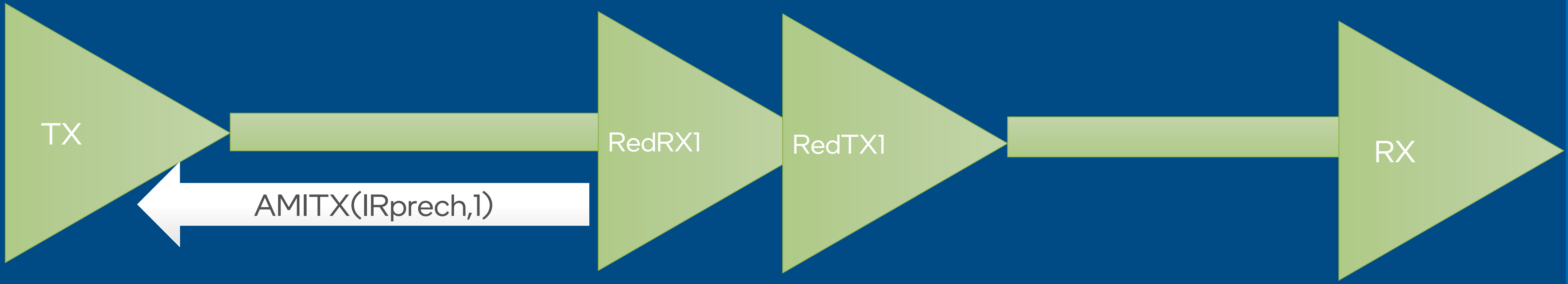
- TX AMI_Init takes its unequalized “post-”channel. RX takes its potentially TX-equalized “pre-” channel indicated by the pink background.

TX to RX



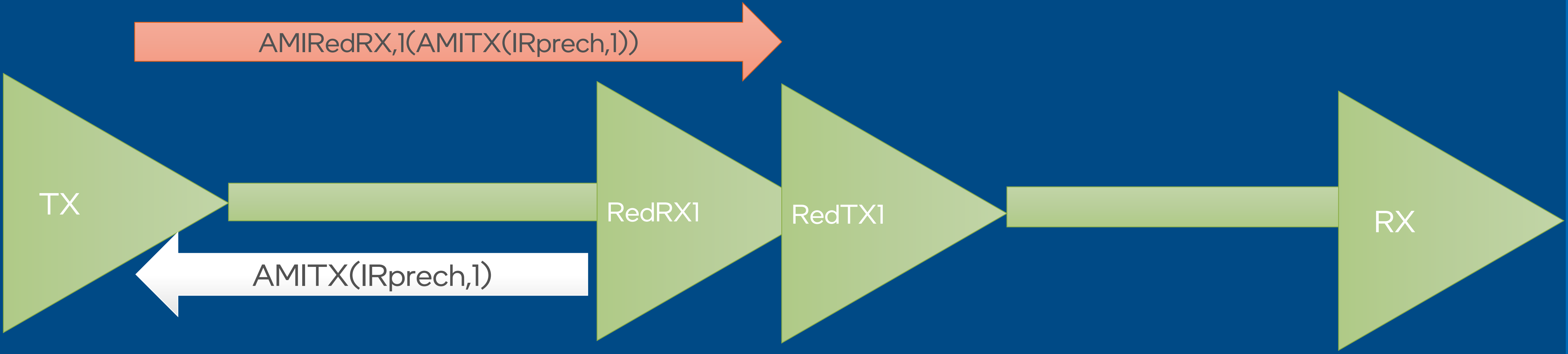
- TX AMI_Init takes its unequalized “post-”channel. RX takes its potentially TX-equalized “pre-” channel indicated by the pink background.
- Looking forward toward the RX, the “post-”channel is provided unequalized to TX. Therefore, the AMI_Init of TX and RX can be invoked in succession and only once each.

Redriver with Both GetWave & Impulse-Based Init



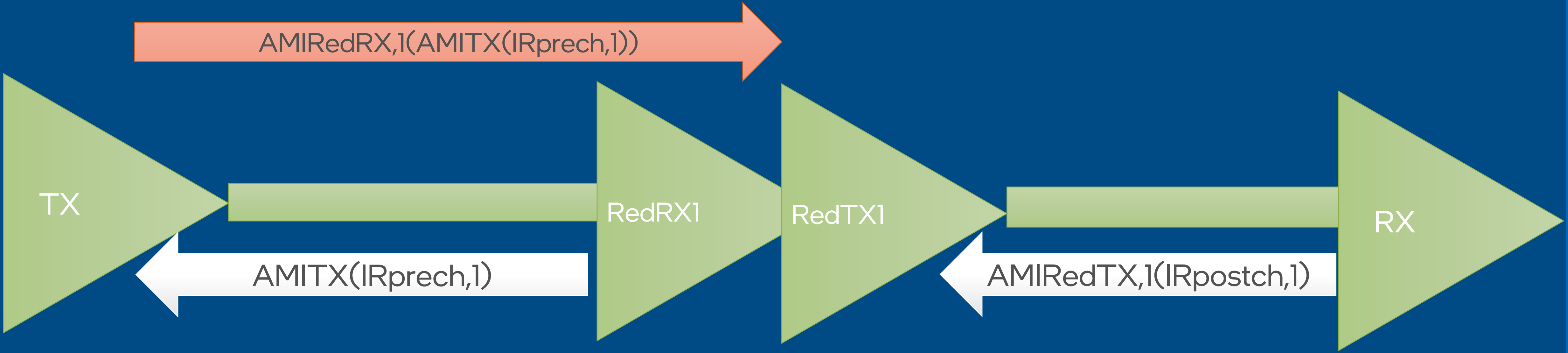
- Pre- and post- are determined with respective redriver index above, starting with #1.

Redriver with Both GetWave & Impulse-Based Init



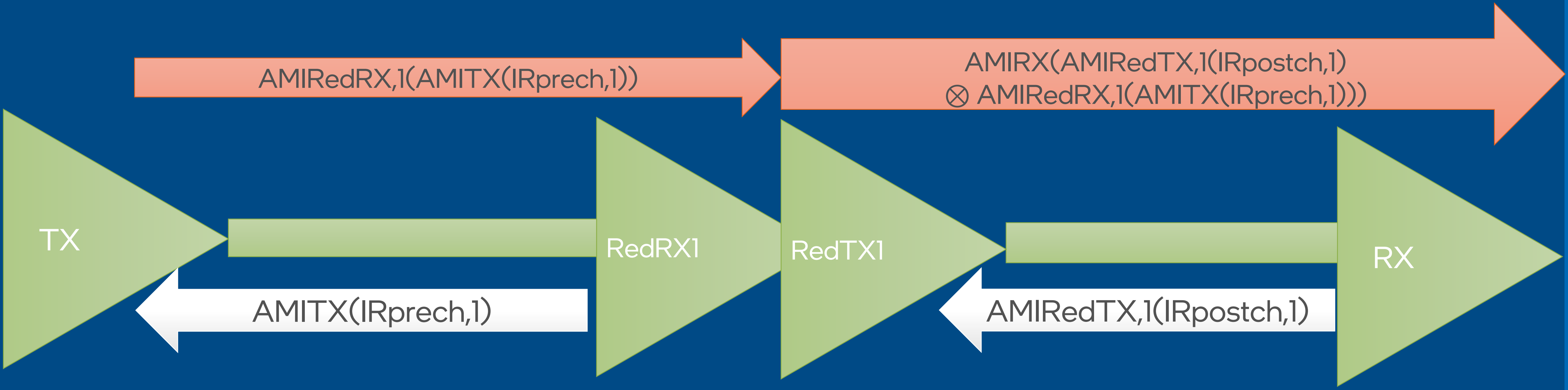
- Pre- and post- are determined with respective redriver index above, starting with #1.

Redriver with Both GetWave & Impulse-Based Init



- Pre- and post- are determined with respective redriver index above, starting with #1.
- Redriver TX deals with the post-channel by default.

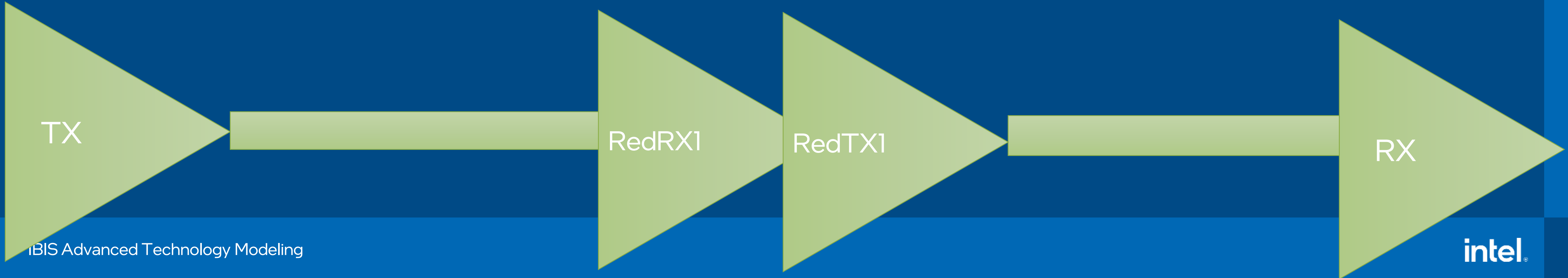
Redriver with Both GetWave & Impulse-Based Init


AMIRedRX,1(AMITX(IRprech,1))
AMIRX(AMIRedTX,1(IRpostch,1)
⊗ AMIRedRX,1(AMITX(IRprech,1)))
TX
RedRX1
RedTX1
RX
AMITX(IRprech,1)
AMIRedTX,1(IRpostch,1)

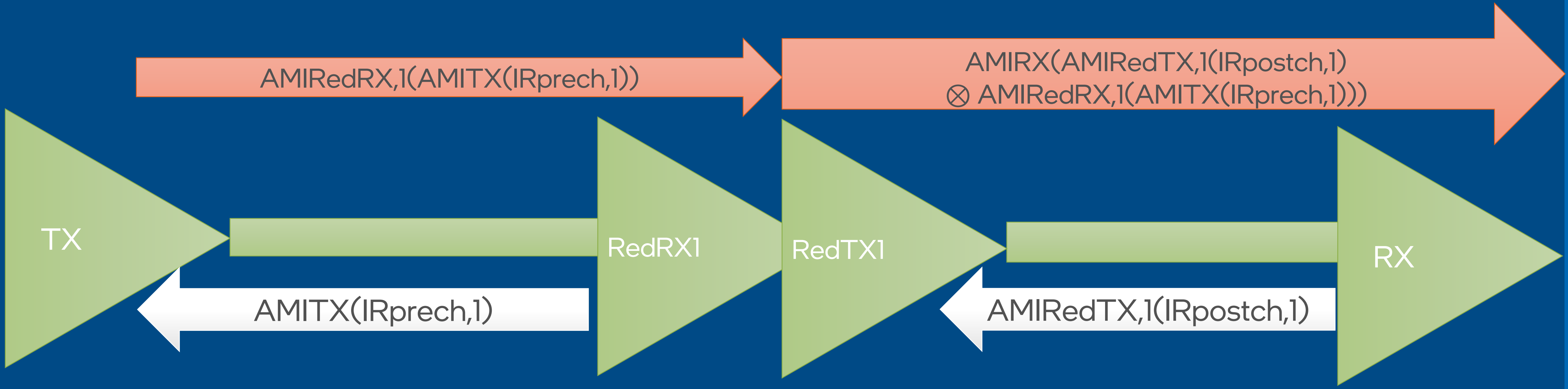
- Pre- and post- are determined with respective redriver index above, starting with #1.
- Redriver TX deals with the post-channel by default.
- Next redriver(s) if any continue with the pink equalized arrows.
- RX is given the entire equalized upstream to be able to run fewer ignore bits with a better initial EQ setup given the effective upstream impulse response.

When AMI_GetWave is missing in a redriver pair, AMI_TX or AMI_RX during empirical signaling:

- Simply disabling such usage would exclude existing & future such AMIs and render them unusable under empirical signaling.
- The simple convolution can always be carried out on the nearest channel to update the channel at hand.
- Basic idea: Empirical signaling will still convolve the channels that can be updated per AMI_Init(s) when AMI_GetWave is unavailable. So, simply merge them into nearest channel and update the channel with the output(s) of AMI_Init call(s) by AMIs that cannot do AMI_GetWave.
- **If TX-AMI is missing AMI_GetWave: IRprech,1' = AMITX(IRprech,1)**
- **If RedRX1 is missing AMI_GetWave: IRprech,1" = AMIRedRX,1(AMITX(IRprech,1'))**
- **If RedTX1 is missing AMI_GetWave: IRpostch,1' = AMIRedTX,1(IRpostch,1)**
- **If RX-AMI is missing AMI_GetWave: IRpostch,1" = AMIRX(IRpostch,1')**
- Invocations are left-to-right and only a subset of the above can be applied per the particular scenario.
- The (pathological) case of combined TX-RX redriver can be left unspecified in the standard. That combination is becoming less common as far as we can tell.



Redriver with both GetWave & Impulse-Based Init


AMIRedRX,1(AMITX(IRprech,1))
AMIRX(AMIRedTX,1(IRpostch,1)
⊗ AMIRedRX,1(AMITX(IRprech,1)))
TX
RedRX1
RedTX1
RX
AMITX(IRprech,1)
AMIRedTX,1(IRpostch,1)

- Pre- and post- are determined with respective redriver index above, starting with #1.
- Redriver TX deals with the post-channel by default.
- The next redriver(s) ,if any, continue with the pink equalized arrows.
- RX is given the entire equalized upstream to enable running fewer ignore bits with a better initial EQ setup given the effective upstream impulse response.
- But this setup does not handle the complicated EQ adaptation needs of a sophisticated redriver.

Solution: AMI_RE_TX/RX_EQ_MODE Keywords

- As we have seen, by default, a TX “sees” its unequalized post-channel; and an RX “sees” its equalized pre-channel or complete upstream. Keywords can be defined to express this default behavior:

  AMI_RE_TX_EQ_MODE = POST

  AMI_RE_RX_EQ_MODE = PRE

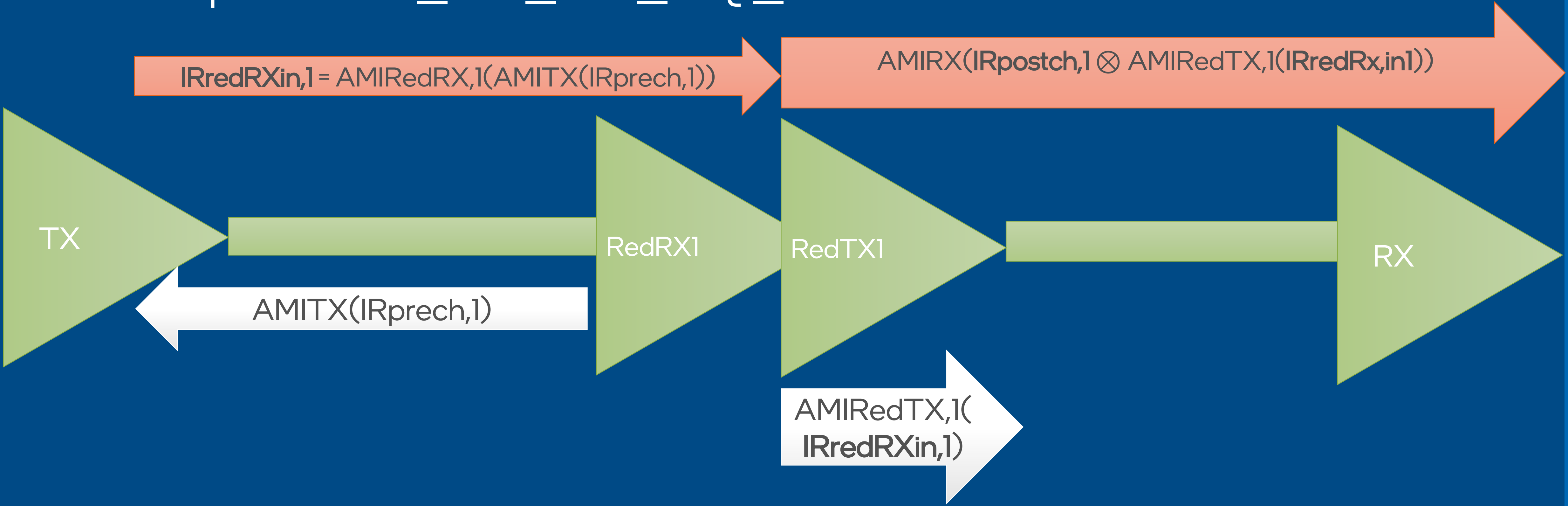
- We can, however, enable redrivers with complicated equalization adaptation in different ways. Expanding the keywords to cover all possibilities:

  AMI_RE_TX_EQ_MODE = { POST, PRE, BOTH }

  AMI_RE_RX_EQ_MODE = { PRE, POST, BOTH }

- Anything that requires a post-channel will be given an unequalized response thereof, to ensure a single AMI_Init call for all AMI components from leftmost TX to rightmost RX.
- Such AMIs must enable both AMI_Init impulse and AMI_GetWave!

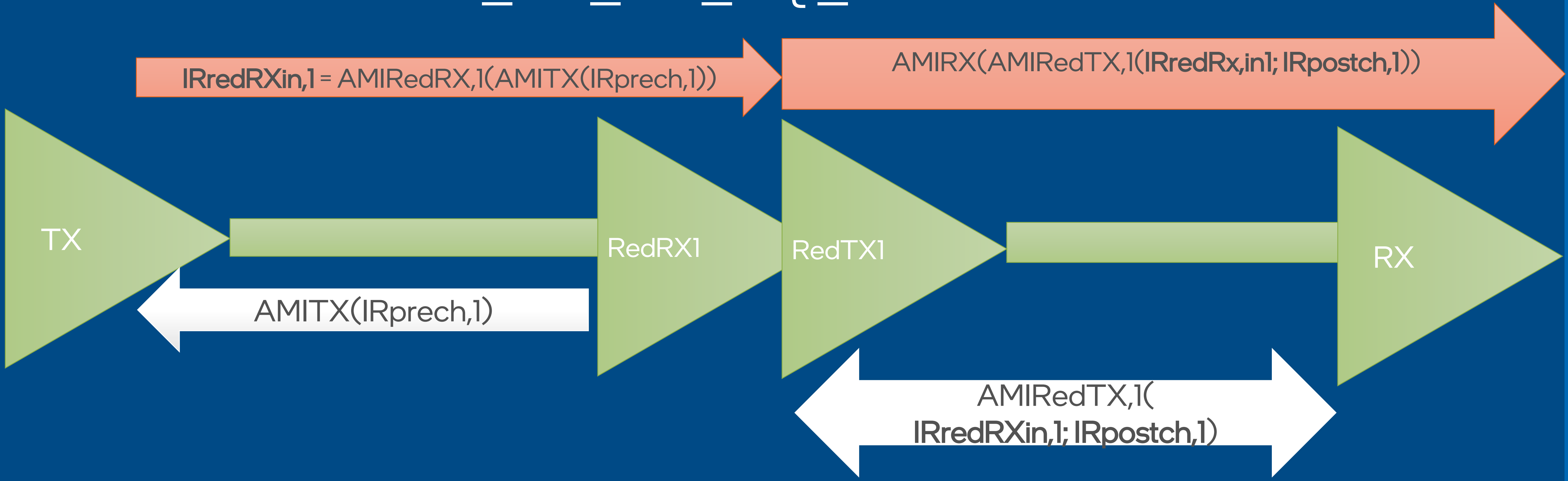
Example: AMI_RE_TX_EQ_MODE = PRE



IRredRXin,1 = AMIRedRX,1(AMITX(IRprech,1))
AMIRX(**IRpostch,1** ⊗ AMIRedTX,1(**IRredRx,in1**))
TX
RedRX1
RedTX1
RX
AMITX(IRprech,1)
AMIRedTX,1(
IRredRXin,1)

- Note, in this setup, the redriver TX takes the redriver RX equalized pre-channel, IRredRXin,1.
- Because the post-channel is not accounted for in the final RX, it needs to be explicitly convolved into the impulse argument for RX AMI.
- Compare the boldfaced items with the earlier slide!

What about AMI_RE_TX_EQ_MODE=BOTH?


IRredRXin,1 = AMIRedRX,1(AMITX(IRprech,1))
AMIRX(AMIRedTX,1(IRredRx,in1; IRpostch,1))
TX
RedRX1
RedTX1
RX
AMITX(IRprech,1)
AMIRedTX,1(
IRredRXin,1; IRpostch,1)

- Note, in this setup, the redriver TX takes both the redriver RX equalized pre-channel, IRredRXin,1 and unequalized post-channel. It becomes a double-argument function!
- Because the post-channel is accounted for the final RX, it does not need to be explicitly convolved into the impulse argument for RX AMI.
- Compare the boldfaced items with the earlier slide!

Backup – Earlier Slides

Synthesis Proposal

- The notions of pre- and post- can refer to any individual redriver in a redriver chain.
- IRpostch,k = IRprech,k+1 for the kth and k+1th redrivers, if applicable.
- Positive index k of the redriver refers to the AMI component from TX to RX, excluding TX and RX.

Symbol or Function	Definition
IRpostch,k	Post-channel IR of kth redriver
IRprech,k	Pre-channel IR of kth redriver
AMIredRX,k(arg)	Analytical/AMI_Init modification of argument IR by kth redriver RX, identity operator if either redriver RX AMI or its returned IR does not exist
AMIredTX,k(arg)	Analytical/AMI_Init modification of argument IR by kth redriver TX, identity operator if either redriver TX AMI or its returned IR does not exist
AMIredTXRX,k(arg)	Analytical/AMI_Init modification of argument IR by combined kth redriver TX-RX, identity operator if either redriver TX-RX AMI or its returned IR does not exist
AMITX(arg)	Analytical/AMI_Init modification of argument IR by TX, identity operator if either TX AMI or its return IR does not exist
IRredRXin,k	The upstream response that the kth redriver RX would "see": IRprech or AMITX(IRprech) or AMIredTX,1(IRprech) for redriver #1 or cascaded cross-convolved forms of these like AMIredRX,1(AMITX(IRprech,1)) ⊗ AMIredTX,1(IRpostch,1) … ⊗ AMIredRX,k(IRredRXin,k) ⊗ AMIredTX,k (IRpostch,k)) . The individual terms will change with certain switches discussed next.

Solution Proposal – Two Optional Reserved Keywords

1. Optional AMI_RED_TX_EQ_MODE => { POST (default), PRE, BOTH }
2. Optional AMI_RED_RX_EQ_MODE => { PRE (default), POST, BOTH }

- Default: Much like an AMI_TX and AMI_RX take their post and pre-channel, the latter possibly equalized by earlier TX, in a symmetric fashion that would be the default AMI behavior.

- Setting AMI_RE_TX_EQ_MODE to PRE would pass it pre-channel, possibly equalized by earlier TX and redriver RX, instead of post-channel:
  - Instead of AMIredTx,k(IRpostch,k), we'd have AMIredTx,k(IRredRxin,k).

- Setting AMI_RE_RX_EQ_MODE to POST would pass it unequalized post-channel instead of pre-channel, possibly equalized by earlier TX:
  - Instead of AMIredRx,k(IRredRxin,k), we'd have AMIredRx,k(IRpostch,k).

- Setting either to both would require an additional column in the input IR matrix. (We should pass even the additional cross-talks for completeness.) Associated single-argument functions now become double-argument. Note that each IR argument is actually a bundle consisting of its data and cross-talking lanes:
  - Instead of AMIredTx,k(IRpostch,k), we'd have AMIredTx,k(IRpostch,k, IRredRxin,k).
  - Instead of AMIredRx,k(IRredRxin), we'd have AMIredRx,k(IRredRxin,k, IRpostch,k).

Solution Proposal – Tabular Form

AMI_RED_TX_EQ_MODE	AMI_RED_RX_EQ_MODE	Input IR to kth redriver TX, i.e., arg(s) of AMIredTX,k()	Input IR to kth redriver RX, i.e., arg(s) of AMIredRX,k()	Upstream IR to final RX assuming k is last redriver
POST (default)	PRE(default)	IRpostch,k	IRredRXin,k	AMIredRX,k(IRredRXin,k) ⊗ AMIredTX,k(Irpostch,k)
PRE	PRE(default)	IRredRXin,k	IRredRXin,k	
BOTH	PRE(default)	IRpostch,k,IRredRxin,k	IRredRXin,k	*One can complete with explicit final channel convolution as needed…*
POST (default)	POST	IRpostch,k	IRpostch,k	
PRE	POST	IRredRXin,k	IRpostch,k	
BOTH	POST	IRpostch,k,IRredRxin,k	IRpostch,k	
POST(default)	BOTH	IRpostch,k	IRredRxin,k,Irpostch,k	
PRE	BOTH	IRredRXin,k	IRredRxin,k,Irpostch,k	
BOTH	BOTH	IRpostch,k,IRredRxin,k	IRredRxin,k,Irpostch,k	

intel®