Symmetric Synthesis of (the Essence of) Existing Re-driver Proposals

Alaeddin A. Aydiner

intel®

Legal Disclaimer

Outline

- Introduction
- Synthesis Proposal
- Discussion

Introduction

Rationales for this synthesis proposal:

- Keep the non-EQ-adapting case as the default for the current status as that is the case for 99% (if not 100%) of existing re-drivers; thus preserve simplicity of implementation
- Naturally extend the standard with a “symmetric” mindset; allow re-driver TX and RX that can EQ adapt to do so with different or extra input as in the earlier proposals
- Preserve the need to invoke AMI_Init only once avoiding multiple AMI_Close, AMI_Init sequences
- Let the multi-re-driver case get implemented readily with a loop enumerating re-drivers from initial TX to final RX

Synthesis Proposal

- The notions of pre- and post- can refer to any individual re-driver in a re-driver chain.
- IRpostch,k = IRprech,k+1 for the k^{th} and $k+1^{th}$ re-drivers, if applicable.
- Positive index k of the re-driver refers to the AMI component from TX to RX, excluding TX and RX.

Symbol or Function	Definition
IRpostch,k	Post-channel IR of k^{th} re-driver
IRprech,k	Pre-channel IR of k^{th} re-driver
AMIredRX,k(arg)	Analytical/AMI_Init modification of argument IR by k^{th} re-driver RX, identity operator if either re-driver RX AMI or its returned IR does not exist
AMIredTX,k(arg)	Analytical/AMI_Init modification of argument IR by k^{th} re-driver TX, identity operator if either re-driver TX AMI or its returned IR does not exist
AMIredTXRX,k(arg)	Analytical/AMI_Init modification of argument IR by combined k^{th} re-driver TX-RX, identity operator if either re-driver 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 k^{th} re-driver RX would "see": IRprech or AMITX(IRprech) or AMIredTX,1(IRprech) for re-driver #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 re-driver 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 k^{th} re-driver TX, i.e., arg(s) of AMIredTX,k()	Input IR to k^{th} re-driver RX, i.e., arg(s) of AMIredRX,k()	Upstream IR to final RX assuming k is last re-driver
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	
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	

One can complete with explicit final channel convolution as needed …

Discussion

- To check:
  - Complete and check final upstream IR in the table…
  - AMI_Init is indeed called once, right?
  - Make sure indeed simple evaluation with a loop enumerating over re-drivers from left to right work through "propagating" IRredRXin,k.
  - Fix fallacies if any, write down a few explicit expressions for IRredRXin,k, which is a function of each re-driver AMI's TX and RX AMI EQ modes…
- Assuming no major fallacies:
  - Are your most desired EQ adaptation features in this proposal?
- Also needed are the changes to handle of AMI_Init only re-driver/non-re-driver AMIs in empirical signaling by updating the linked channel accordingly (Unless it is a combined TX-RX re-driver, linked channel is well defined. We can keep that unspecified. IMO, combined one is conceptually broken.)


intel