

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

Alaeddin A. Aydiner



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.

Copyright © 2020, Intel Corporation. All Rights Reserved.

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.
- $IR_{postch,k} = IR_{prech,k+1}$ 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
$IR_{postch,k}$	Post-channel IR of k^{th} re-driver
$IR_{prech,k}$	Pre-channel IR of k^{th} re-driver
$AMI_{redRX,k}(arg)$	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
$AMI_{redTX,k}(arg)$	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
$AMI_{redTXRX,k}(arg)$	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
$AMI_{TX}(arg)$	Analytical/AMI_Init modification of argument IR by TX, identity operator if either TX AMI or its return IR does not exist
$IR_{redRXin,k}$	The upstream response that the k^{th} re-driver RX would "see": IR_{prech} or $AMI_{TX}(IR_{prech})$ or $AMI_{redTX,1}(IR_{prech})$ for re-driver #1 or cascaded cross-convolved forms of these like $AMI_{redRX,1}(AMI_{TX}(IR_{prech,1})) \otimes AMI_{redTX,1}(IR_{postch,1}) \dots \otimes AMI_{redRX,k}(IR_{redRXin,k}) \otimes AMI_{redTX,k}(IR_{postch,k})$. 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., $\text{arg}(s)$ of $\text{AMlredTX},k()$	Input IR to k^{th} re-driver RX, i.e., $\text{arg}(s)$ of $\text{AMlredRX},k()$	Upstream IR to final RX assuming k is last re-driver
POST (default)	PRE(default)	$\text{IRpostch},k$	$\text{IRredRXin},k$	$\text{AMlredRX},k(\text{IRredRXin},k) \otimes \text{AMlredTX},k(\text{IRpostch},k)$
PRE	PRE(default)	$\text{IRredRXin},k$	$\text{IRredRXin},k$	
BOTH	PRE(default)	$\text{IRpostch},k, \text{IRredRXin},k$	$\text{IRredRXin},k$	
POST (default)	POST	$\text{IRpostch},k$	$\text{IRpostch},k$	<i>One can complete with explicit final channel convolution as needed...</i>
PRE	POST	$\text{IRredRXin},k$	$\text{IRpostch},k$	
BOTH	POST	$\text{IRpostch},k, \text{IRredRXin},k$	$\text{IRpostch},k$	
POST(default)	BOTH	$\text{IRpostch},k$	$\text{IRredRXin},k, \text{IRpostch},k$	
PRE	BOTH	$\text{IRredRXin},k$	$\text{IRredRXin},k, \text{IRpostch},k$	
BOTH	BOTH	$\text{IRpostch},k, \text{IRredRXin},k$	$\text{IRredRXin},k, \text{IRpostch},k$	

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®