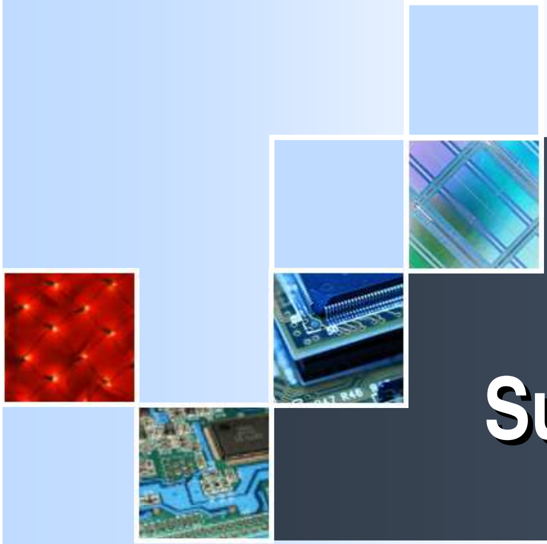


BIRD Proposal: Extending IBIS-AMI to Support Back-Channel Communications

IBIS-ATM Committee
July 2011

Marcus Van Ierssel – Snowbush IP (Gennum)
Kumar Keshavan - Sigrity
Ken Willis – Sigrity
Walter Katz - SISoft

Overview

- Assumptions
- Proposed modifications to support back-channel
- New Reserved_Parameters
- Back-channel BCI file
- Flow changes
- 802.3KR example

Assumptions

- Back-channel functionality will be supported for time domain simulations only. Back-channel functionality does not affect statistical analysis.
- This functionality will be implemented in the AMI_GetWave function (and the Rx AMI_Init function)

Modifications Required for BIRD

- Enhance AMI_GetWave to allow “AMI_parameters_out” to be taken both into as well as out of the AMI model (covered in BIRD 128)
- New Reserved_Parameters
- Definition of Back-Channel Interface .bci file format
  - Includes new Reserved_Parameters
  - New section for Protocol_Specific parameters
- Flow Changes
  - Add back-channel training flow before standard simulation flow

New Reserved_Parameters

- Training
  - Turns back-channel training on/off
- Backchannel_Protocol
  - Tells user which back-channel protocols are supported
  - User selects which protocol to use
  - Protocol selection points to .bci file with the back-channel protocol information details
  - Both Tx and Rx must point to same file for back-channel communication to occur

We Are Signal Integrity

Reserved_Parameters > Training

(Training (Usage In) (Type String) (List “Off” “On”) (Default “Off”) (Description "Turns training on or off"))

Reserved_Parameters > Backchannel_Protocol

(Backchannel_Protocol (Type String) (Usage In)

(List “None” “PCIeG3” “802.3KR”)

(Description

"This Device can support backchannel training for PCIeGen3 and Ethernet 802.3KR standards. When “None” the models shall support the normal IBIS 5.x flows. When calling the Tx and Rx AMI_Init function, the EDA tool shall pass the value:

<full_path_to>/<protocol>.bci

The EDA tool is responsible for determining <full_path_to>. This file may be located in the same directory as the .ibs, .ami, dll files or may be located in library folders controlled by the EDA tool.”)

Reserved_Parameters > Back-Channel .bci File

- Training_Pattern > describes the bit stream used for training
  - Preamble
  - Data
    - PRBS or LFSR
    - Length
  - Postamble
- Max_Train_Bits
  - Max duration of training
- TrainingDone
  - Signifies that training is completed

Back-Channel .bci File Format

```
(802.3KR
        (Reserved_Parameters
                (Training_Pattern (Description "Defines the training pattern")
                        (Preamble (Usage Info) (Type String) (Value “b11111111111111110000000000000000 “)
                                                (Description "Leading preamble pattern."))
                    (Data (Usage Info) (Type String) (“LFSR 1,9,11 random 4096”)
                                                (Description “Training pattern."))
                    (Postamble (Usage Info) (Type String) (Value b00)
                                                (Description “Trailing postamble pattern."))
                        )
                (Max_Train_Bits (Usage In) (Type Integer) (Value 500000)
                                                (Description "Number of total training bits allowed"))
                (TrainingDone (Usage InOut) (Type Boolean) (List False True)
                                                (Description “If True then training is done"))
        )
        (Protocol_Specific
                    (-1 (Usage InOut) (Type Integer) (List -1 0 1) (Default 0)
                                (Description "Parameter name is standard-specific, and can be any legal Type"))
                    (0 (Usage InOut) (Type Integer) (List -1 0 1) (Default 0)
                                (Description "Parameter name is standard-specific, and can be any legal Type"))
                    (1 (Usage InOut) (Type Integer) (List -1 0 1) (Default 0)
                                (Description "Parameter name is standard-specific, and can be any legal Type"))
         )
 )
```

Example of Protocol_Specific Parameters

```
(TapIncDec (Description "When written by Rx, -1 means to decrement,
                         0 hold same value, and 1 means to increment.
                         When written by Tx, -1 means low limit has
                         been reached, 0 means the setting is
                         adjustable, and 1 means high limit has been
                         reached.")
 (-1 (Usage InOut) (Type Integer) (List -1 0 1) (Default 0))
 (0 (Usage InOut) (Type Integer) (List -1 0 1) (Default 0))
 (1 (Usage InOut) (Type Integer) (List -1 0 1) (Default 0))
)
```

- Protocol_Specific parameters are analogous to Model_Specific parameters, but are associated with a particular backchannel protocol
- They are governed by the same AMI syntax
- Both the Tx and Rx AMI models must handle these Protocol_Specific parameters

Defining Bit Patterns

- We should be able to easily define as a concatenated list of waveform snippets for Preamble/Data/Postamble.
- Example using Format Table:

```
(Data (Type String) (Usage Info)
  (Table
    (“b01111111111111111000000000000000 5”)
    (“h0123456789ABCDEF0123456789ABCDEF 10”)
    (“o01234567012345670123456701234567 10”)
    (“File abc.bpi 3”)
    (“PRBS 11 b11110000111 1”)
    (“LFSR “1,9,11” random 4096”)
  (Description “
    Strings that begin b,h,o, denote Binary, Hex, Octal.
    These bit patterns are followed by a repeat count.
    The default is 1, which means the pattern is added once.
    Strings that begin with PRBS generate a Pseudo Random Binary Sequence using a Linear Feedback Shift Register.
        PRBS is followed by 3 fields: <duty cycle> <seed> <repeat count>
                <duty cycle> A positive, integer number. The PRBS patter will repeat every 2^<duty cycle> bits.
                <seed> A non-negative integer number, can be represented as b… or “random for random seed
                <repeat count> is non-negative integer number. The number of times this bit pattern is to be inserted into the stimulus.”))
        LFSR is followed by 3 fields: <taps> <seed> <data_len>
                <taps> lfsr taps
                <seed> A non-negative integer number, can be represented as b… or “random for random seed
                <data len> is optional non-negative integer number. The length of the data pattern generated by this lfsr in bits. if the value is ‘R’ run it
      forever ”))

     Strings that begin with File reference a file that contains a sequence of binary, octal or hex numbers.
        File is followed by two fields: <file name> <repeat count>))
```

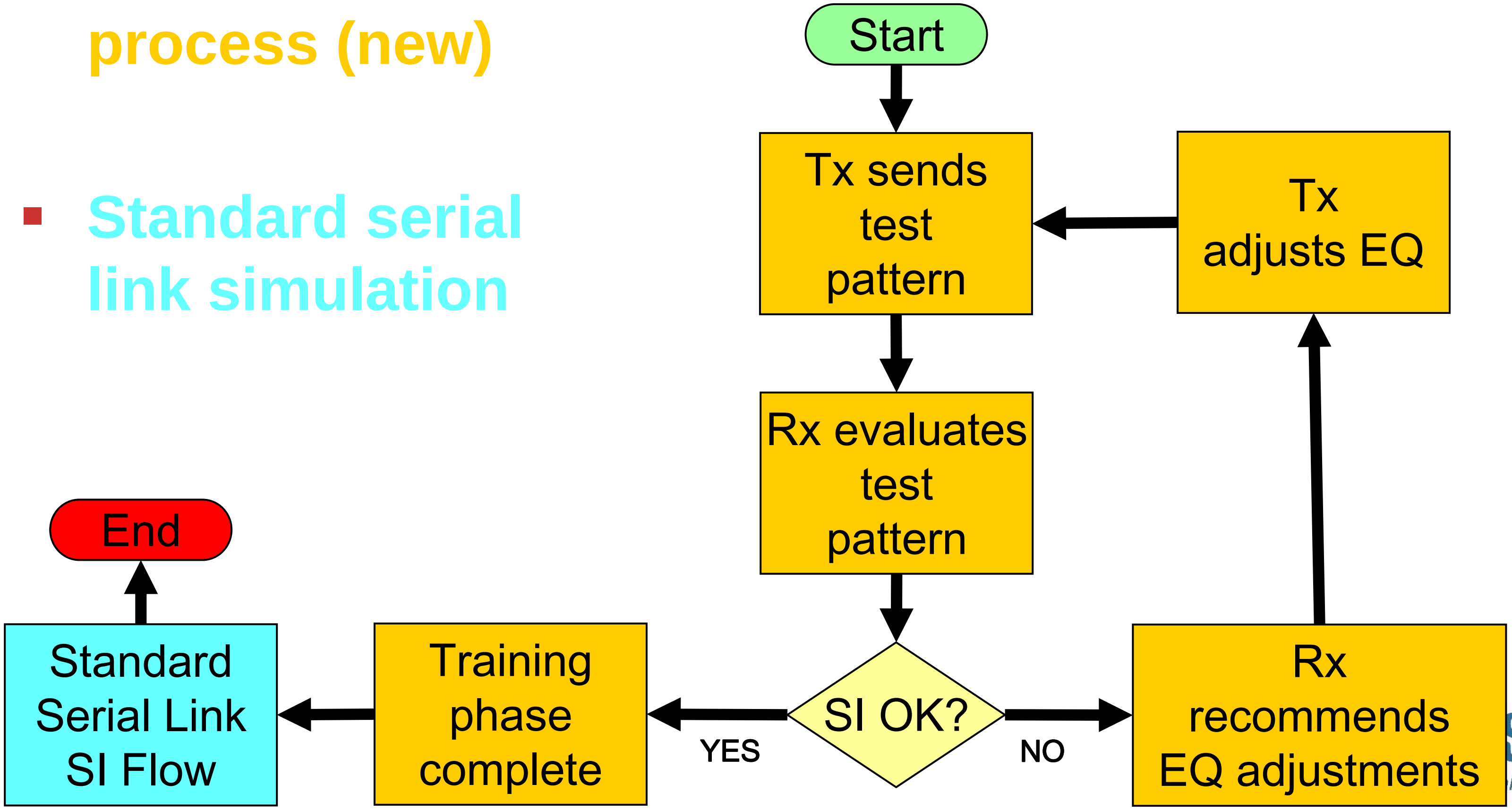
Example Bit Pattern File

Contents of “a_bit_pattern.bpi” could be:

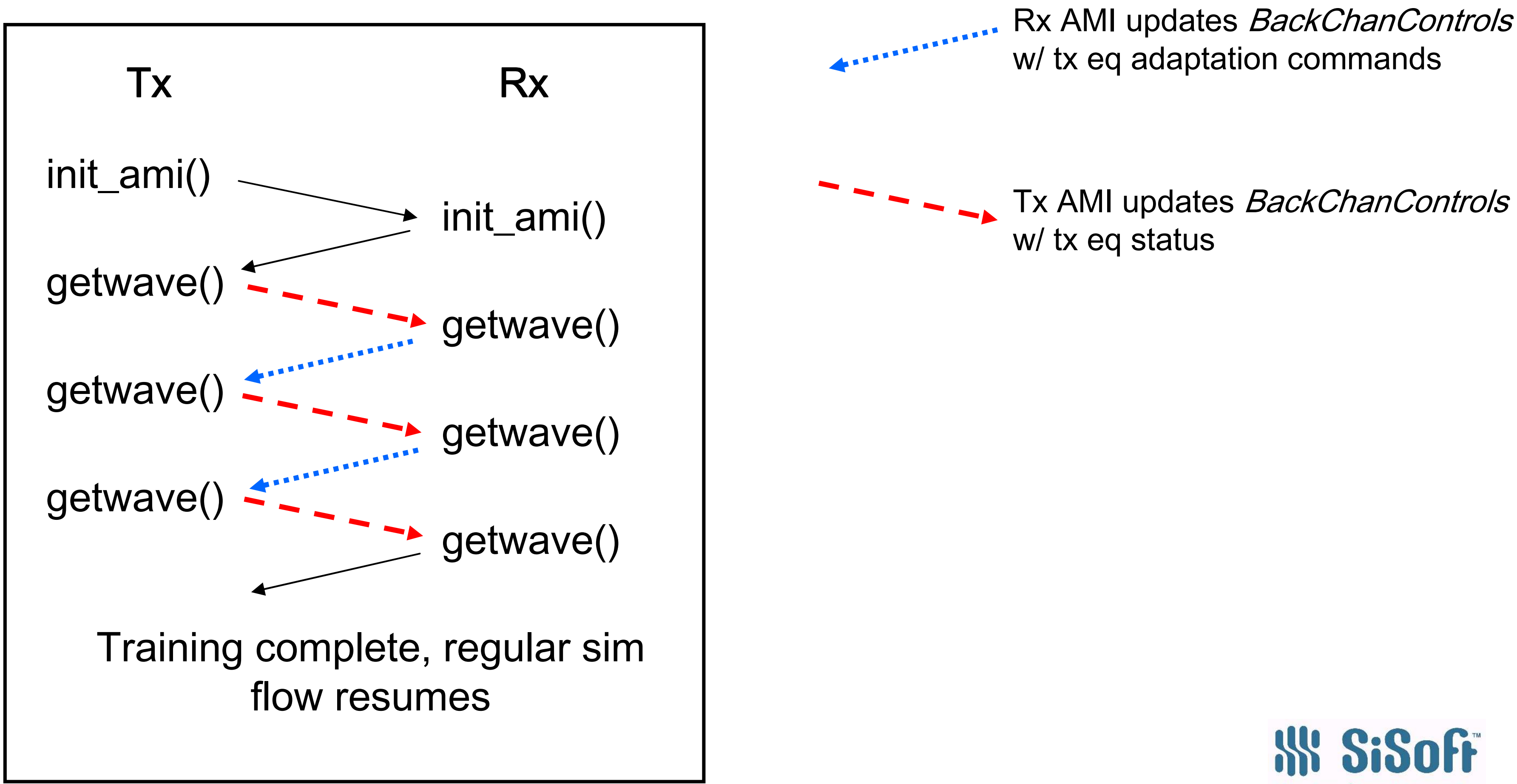
h555555555555555000000001111111100001000f0f0f0f0f
555555555 555555000000001111
000011111111000010000f0f0

Flow Changes

- Back-Channel process (new)
- Standard serial link simulation


Start
Tx sends test pattern
Tx adjusts EQ
Rx evaluates test pattern
End
SI OK?
YES
NO
Rx recommends EQ adjustments
Training phase complete
Standard Serial Link SI Flow

Back-Channel Flow Detail



Tx

Rx

init_ami()

init_ami()

getwave()

getwave()

getwave()

getwave()

getwave()

getwave()

Training complete, regular sim flow resumes

Rx AMI updates *BackChanControls* w/ tx eq adaptation commands

Tx AMI updates *BackChanControls* w/ tx eq status

802.3KR Back-Channel .bci File Example

```
(802.3KR
 (Reserved_Parameters
   (Training_Pattern (Description
                “On first call to Tx GetWave when (Training On) input stimulus to Tx GetWave shall be Preamble
        followed by Data repeated until Rx GetWave sets (Training Off). When Rx GetWave sets (Training
        Off), the next input stimulus will be Postamble then simulation stimulus”)
       (Preamble (Usage Info) (Type String)
              (Value “b11111111111111110000000000000000 1”)
              (Description “Training Preamble sent once”))
        (Data (Usage Info) (Type String) (“LFSR 1,9,11 random 4096”)
                                                  (Description “Training pattern."))
        (Postamble (Usage Info) (Type String) (Value “b00 1”)
                        (Description “Training postamble pattern, repeated 5 times”)))
       (Max_Train_Bits (Usage In) (Type Integer) (Value 500000) (Default 500000)
                       (Description "Number of total training bits allowed"))
       (TrainingDone (Usage InOut) (Type Boolean) (List False True)
                       (Description “If True then training is done"))
  )
)
```

802.3KR Back-Channel .bci File Example (cont)

```
 (Protocol_Specific
      (Description “From Rx
                         -n Decrements tap by n
                          0 Tap is unchanged
                          n Increments tap by n
                     From Tx
                         -1 Low limit has been reached
                          0 Setting is adjustable
                          1 High limit has been reached”)
      (-1    (Usage InOut) (Type Integer) (Range 0 -1 1)
                     (Description “Pre-cursor tap control"))
      (0     (Usage InOut) (Type Integer) (Range 0 -1 1)
                     (Description “Main tap control"))
      (1     (Usage InOut) (Type Integer) (Range 0 -1 1)
                     (Description “First post-cursor tap control"))
  )
)
```

Example Parameter Tracing

AMI_parameters_in and _out

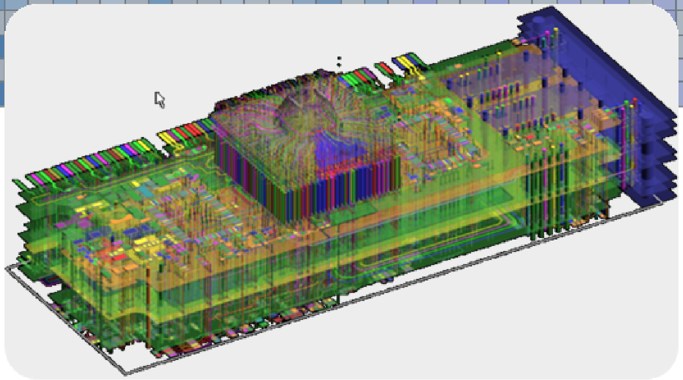
Note: Using AMI_parameters_io to describe usage of AMI_parameters_out (BIRD 128).

```
AMI_parameters_in from simulator to Tx AMI_Init
  (tx_root (Backchannel_Protocol “C:/Library/IBIS/802.3KR.bci”) (Training “On”))
AMI_parameters_in from simulator to Rx AMI_Init
  (rx_root (Backchannel_Protocol “C:/Library/IBIS/802.3KR.bci”) (Training “On”)))
             (802.3KR (TrainingDone False)(Max_Train_Bits 500000))
AMI_parameters_io from Rx AMI_Init
  (rx_root …)(802.3KR (TrainingDone False) (-1 0) (0 0) (1 0))
AMI_parameters_io to Tx AMI_GetWave
             (802.3KR (TrainingDone False) (-1 0) (0 0) (1 0))
AMI_parameters_io from Tx AMI_GetWave
  (tx_root …)(802.3KR (-1 0)(0 0)(1 0))
AMI_parameters_io to Rx AMI_GetWave
             (802.3KR (-1 0)(0 0)(1 0))))
AMI_parameters_io from Rx AMI_GetWave
   (rx_root …)(802.3KR (TrainingDone True) (-1 -1) (0 +1) (1 -1))
AMI_parameters_io to Tx AMI_GetWave
              (802.3KR (TrainingDone True) (-1 -1) (0 +1) (1 -1))
```

Issues

- The content of .bci files will contain Protocol_Specific parameters
- AMI models that utilize these .bci files will need to support these parameters
- We need to formalize a process by which these .bci files are developed, reviewed, approved,a nd posted to the AMI modeling community

Thank You!



SiSoft™
We Are Signal Integrity