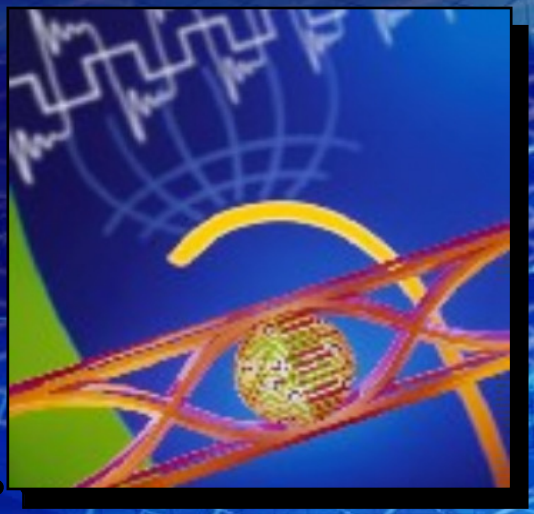
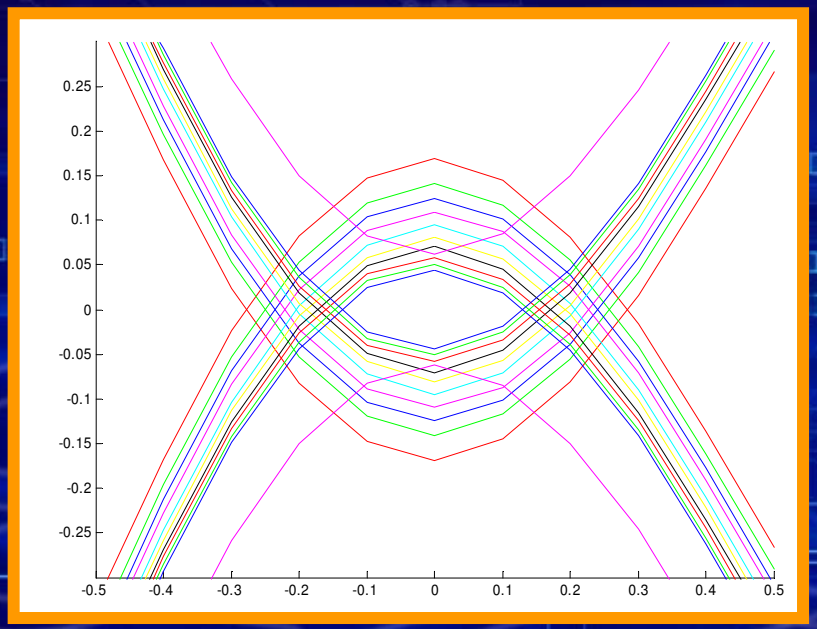
Current IBIS-AMI Support

IBIS Summit, DAC, June 2008

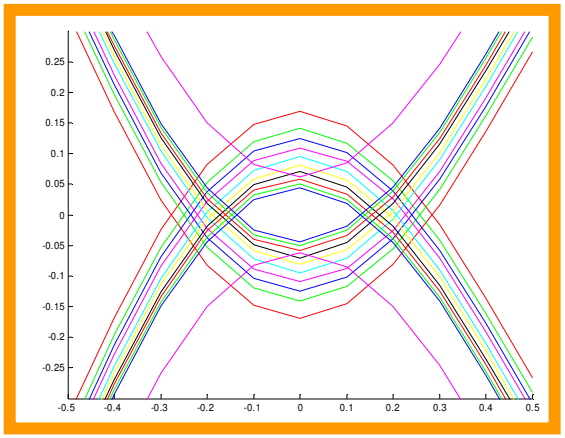




Arpad Muranyi

Mentor Graphics®

Current IBIS-AMI Support



IBIS Summit, DAC, June 2008

1. **Taking care of some unfinished business**
   - **Updated performance benchmarks**
   - **Sliding window algorithm in VHDL-AMS**
2. **Attributes in VHDL-AMS**
3. **An IBIS-AMI example using the FOREIGN attribute**
4. **Conclusions – future work**

Taking care of some unfinished business

- In my last presentation I promised that the source code of the VHDL-AMS and Matlab models will be posted on the IBIS web site shortly after the summit
  - http://www.vhdl.org/pub/ibis/summits/feb08/muranyi.pdf (pg. 46)
  - I apologize for not having this done yet
- The benchmark data of the above presentation (pg. 43) need some correction
  - running the GetWave function with a single call (using the entire waveform) or with multiple calls (using the sliding window technique) *has a significant effect on its execution time*
  - the benchmarks in the above presentation mixed multi/single calls to GetWave in the comparison, *putting the VHDL-AMS and Matlab implementations in an unfavorable light*

Recall – benchmarks from last time

PRBS register length:	**22**
Sampling time:	**25 ps**
Bit time:	**200 ps**
Stop time:	**20 µs**

Original code in ANSI C:	**10.0 sec**
Matlab (properly coded):	**3.5 sec**
Matlab (C clone):	**14:00.0 sec**
VHDL-AMS (simulator #1):	**23:33.0 sec**
VHDL-AMS (simulator #2):	**1:15:45.0 sec**

INCORRECT!

Updated benchmarks

PRBS register length:	**22**
Sampling time:	**25 ps**
Bit time:	**200 ps**
Stop time:	**20 µs**

	Single GW call	Multiple GW call	improvement factor
	h:mm:ss	h:mm:ss	
ANSI C code:	0:03:30	0:00:17	12.35
Matlab C-clone:	0:14:25	not implemented	216.25
Matlab "proper":	0:00:04	not implemented	
Octave C-clone:	hangs	not implemented	
Octave "proper":	0:01:47	not implemented	
VHDL-AMS (vendor 1):	0:20:15	0:00:04	303.75
VHDL-AMS (vendor 2):	1:14:02	0:00:22	201.91
Octave vs. Matlab:			26.75

Notes on new benchmarks

- These benchmarks are still not 100 % accurate and fair
  - they were done on two different machines (software to software)
    - 2.33 GHz Dell D820 Latitude laptop with 2 GB memory
    - 3.19 GHz Dell Dimension 8300 desktop with 1 GB memory
  - there are differences in how the tested software load themselves, the models, and the way the models are compiled and executed, making it hard to compare the specific aspects of performance
  - measurements were taken with a clock on the screen
- No far reaching conclusions should be made based on this data, since it is only a crude comparison
  - the tests should be redone under more rigorous control for more precise results
    - run them on the same machine
    - implement timing measurement points in the code
- *The performance of the VHDL-AMS IBIS-AMI model is competitive with the C and Matlab implementations*

Sliding window algorithm in VHDL-AMS

- The sliding window algorithm was added to the VHDL-AMS Tx model (for the sake of this test only)
  - normally the EDA tool would chop up the waveform and call GetWave multiple times
  - in this test, the test bench still calls GetWave only once, and the sliding window algorithm is applied to the waveform in the GetWave function
  - in terms of the end result, it really doesn't matter whether it is the caller of GetWave or GetWave itself that performs the sliding window algorithm
  - this was an easier modification to the existing code

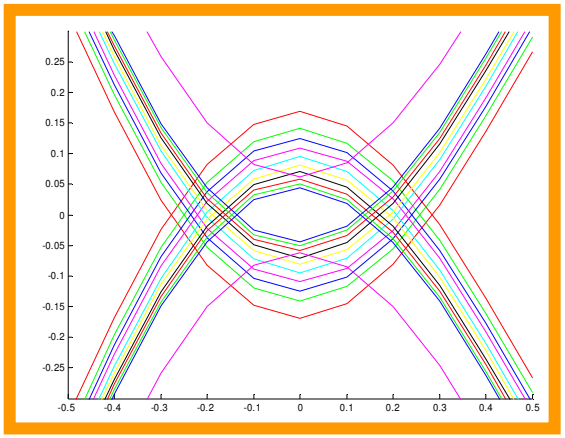
Sliding window code in Tx GetWave

original Tx code

```
i := 0;
while i < WaveSize loop
  if (ActualWindowSize > WaveSize-i) then
    ActualWindowSize := WaveSize - i;
  end if;

  for indx in 0 to ActualWindowSize-1 loop    -- Save the time of each edge of WaveIn in ClockTimes
    if (WaveIn(indx+i)*LastIn < 0.0 ) then
      ClockTimes(Clock_indx) := GwTime + (real(indx+i)-0.5) * SampleInterval;
      Clock_indx := Clock_indx + 1;
    end if;
    if (WaveIn(indx+i) /= LastIn) then            -- Add step response
      StepSize := WaveIn(indx+i) - LastIn;
      for yndx in 0 to RowSize-1 loop
        tmp_dbl(indx+yndx) := tmp_dbl(indx+yndx) + StepSize * StepResponse(yndx);
      end loop;
      for yndx in indx+RowSize to TempSize-1 loop
        tmp_dbl(yndx) := tmp_dbl(yndx) + StepSize * StepResponse(RowSize-1);
      end loop;
    end if;
    LastIn := WaveIn(indx+i);
    ReturnVec(indx+i) := tmp_dbl(indx);           -- Save the output
  end loop;

  ClockTimes(Clock_indx) := -1.0;                 -- Terminate the list of clock ticks

  for indx in 0 to RowSize-1 loop                 -- Save the remaining response for the next block of data
    tmp_dbl(indx) := tmp_dbl(indx+ActualWindowSize);
  end loop;
  for indx in RowSize-1 to tmp_dbl'right loop
    tmp_dbl(indx) := tmp_dbl(RowSize-1);
  end loop;

  GwTime := GwTime + real(ActualWindowSize) * SampleInterval;

  i := i + RequestedWindowSize;
end loop;
```

Current IBIS-AMI Support

IBIS Summit, DAC, June 2008



1. Taking care of some unfinished business
   - Updated performance benchmarks
   - Sliding window algorithm in VHDL-AMS
2. Attributes in VHDL-AMS
3. An IBIS-AMI example using the FOREIGN attribute
4. Conclusions – future work

A handy language feature of VHDL-AMS

- **Attributes**
  - there are two kinds: pre-defined and user-defined
- **Pre-defined attribute examples:**
  - T'left, T'right, T'low, T'high, etc...
  - A'left(n), A'right(n), A'low(n), A'high(n), etc…
  - S'delayed(t), S'stable(t), S'quiet(t), etc…
  - Q'tolerance, Q'dot, Q'integ, Q'above,

    Q'ltf(num, den), Q'ztf(num, den, t, delay), etc…
- **For more details, refer to any VHDL-AMS reference**

  P. J. Ashenden, G. D. Peterson, D. A. Teegarden: "The system designer's Guide to VHDL-AMS", Chapter 22

User-defined attributes in VHDL-AMS

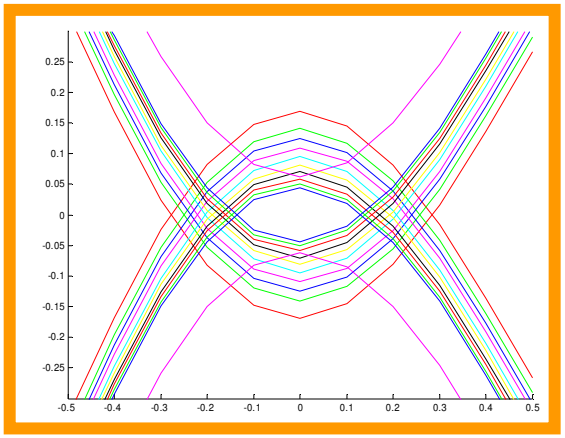
- A VHDL-AMS user can declare attributes and write attribute specifications "…nominating items which take on the attribute with particular values"
  - user defined attributes can be given to a wide variety of entity_classes, such as:
    - procedures, functions, packages,
    - architectures, natures, quantities, terminals,
    - constants, variables, signals,
    - etc… (long list)
- Mentor's VHDL-AMS implementation has a built-in user-defined attribute called "FOREIGN" acting as a C-code interface (which in turn can call practically any compiled code)
  - this capability existed for a long time and can be utilized to execute IBIS-AMI models within VHDL-AMS

Calling IBIS-AMI through VHDL-AMS

- **The VHDL-AMS model calls a C-code wrapper through the FOREIGN attribute**
  - **the C function argument types are mapped to VHDL-AMS types**
  - **obviously there are some limitations, but the wrapper function can take care of most of the type conversions if necessary**
  - **the wrapper function can take care of the IBIS requirements of the calling conventions of the Init, GetWave, and Close functions**
  - **any additional feature and capability can be programmed into the wrapper function(s)**
- **Once an "IBIS-AMI wrapper" has been developed for VHDL-AMS, any IBIS-AMI model can be executed in any of our tools that has VHDL-AMS capabilities**
  - ***all of this can be user written, no product changes are required***

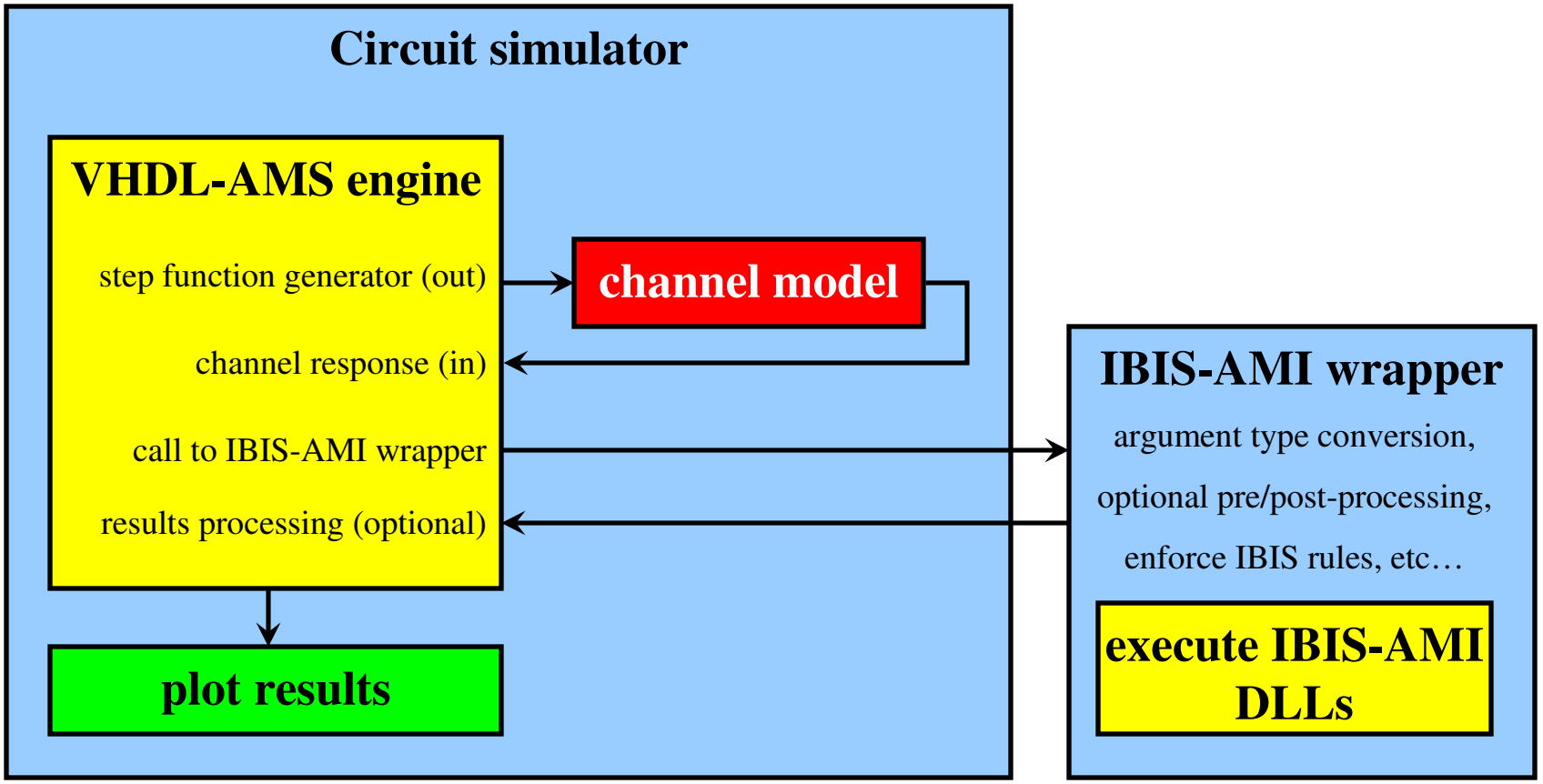
Current IBIS-AMI Support

IBIS Summit, DAC, June 2008



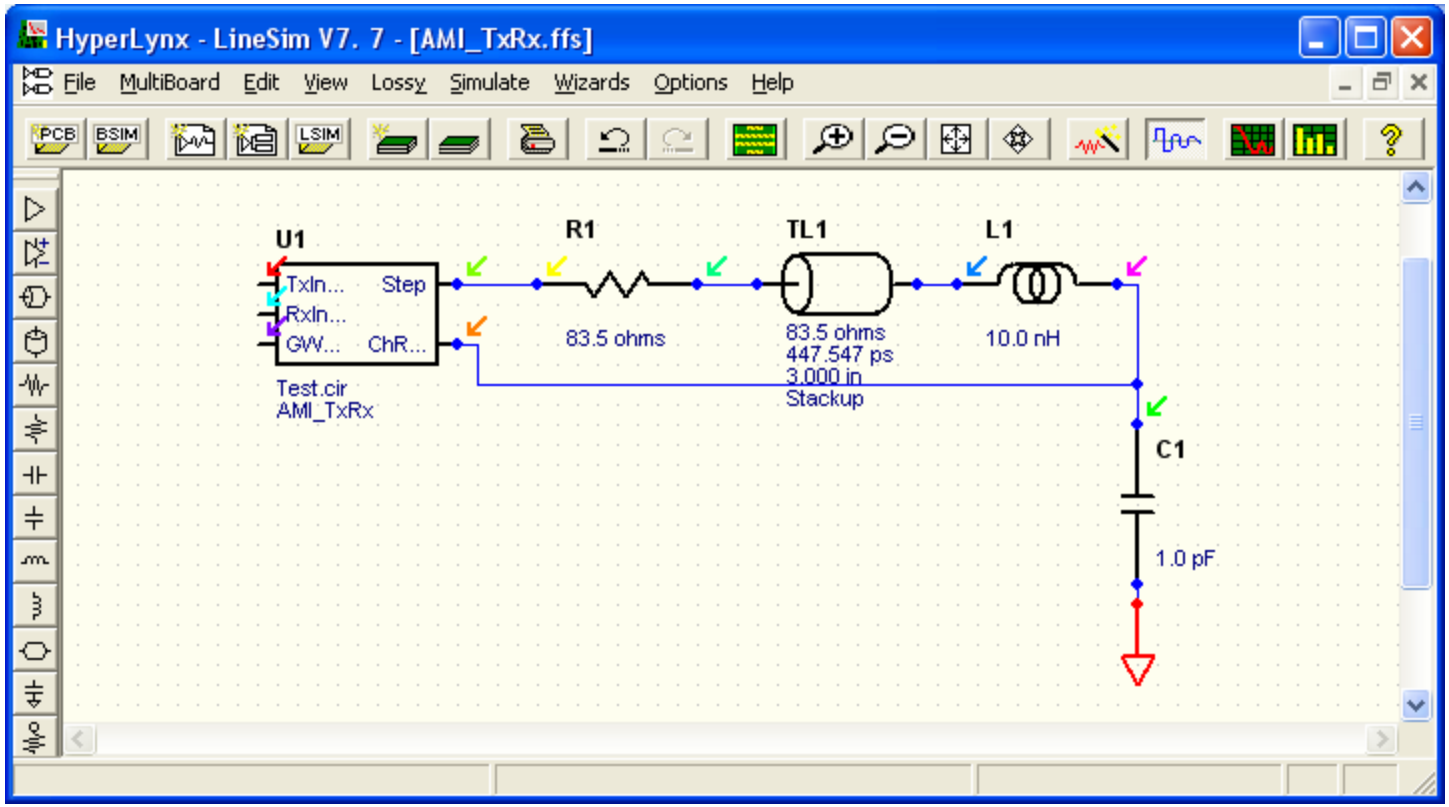
1. Taking care of some unfinished business
   - Updated performance benchmarks
   - Sliding window algorithm in VHDL-AMS
2. Attributes in VHDL-AMS
3. An IBIS-AMI example using the FOREIGN attribute
4. Conclusions – future work

Block diagram of the following example



This is only one of many possible ways of implementing IBIS-AMI support via VHDL-AMS

Example with IBIS-AMI Tx and Rx models



Example circuit description

- U1 contains a VHDL-AMS model
  - it generates a step function to excite the “channel”
- R1 represents a simple resistive driver impedance
- TL1, L1, C1 represents a T-line, package and input
  - the “channel” can be an arbitrary circuit, including S-parameter models, but it must include the Tx and Rx impedances
- The first part of the TD simulation generates a channel response
  - the length of this is a parameter in the VHDL-AMS model
- When the channel response is done, the AMI Tx and Rx models are executed using that channel response
  - the VHDL-AMS model includes a PRBS pattern generator
- After that, the results are returned to the simulator through the VHDL-AMS model for plotting

VHDL-AMS code segment for U1

```
begin
  ------------------------------------------------------------------------
  StepGen : process is
  begin
    wait for StepTime;
    sStep <= 1.0;
    wait;
  end process StepGen;
  ------------------------------------------------------------------------
  Run_AMI : process is
    variable  TxImpulseMatrix : real_vector(TxMatrixInOut'range) := (others => 0.0);
    variable  RxImpulseMatrix : real_vector(RxMatrixInOut'range) := (others => 0.0);
    variable  WaveIn          : real_vector(0 to WaveSize-1)     := (others => 0.0);
  begin
    wait for StepTime;
    if (RowSize <= ChRespSize) then
      TxMatrixInOut(RowSize) <= VwfmIn'dot;
      RowSize                <= RowSize + 1;
    else
      TxImpulseMatrix := TxMatrixInOut;
      RxImpulseMatrix := RxMatrixInOut;
      WaveIn          := AMIwaveInOut;
      computing : VHDL_IBIS_AMI(TxImpulseMatrix,   RowSize,
                                Aggressors,        StepTime,
                                BitTime,           UseTxInitOutput,
                                TxParameters,      TxDLLfileName,
                                WaveIn,            WaveSize,
                                RxImpulseMatrix,   UseRxInitOutput,
                                RxParameters,      RxDLLfileName);
      TxMatrixInOut(TxImpulseMatrix'range) <= TxImpulseMatrix;
      RxMatrixInOut(RxImpulseMatrix'range) <= RxImpulseMatrix;
      AMIwaveInOut(WaveIn'range)           <= WaveIn;
      wait;
    end if;
  end process Run_AMI;
  ------------------------------------------------------------------------
```

IBIS-AMI wrapper call (C-code)

VHDL-AMS code segment for U1 (cont'd)

```
------------------------------------------------------------------------
  Ticker : process is
  begin
    wait for StepTime;
    if (CountInit < ChRespSize) and (Now > ChRespDuration) then
      CountInit <= CountInit + 1;
    end if;
    if (CountGW < AMIwaveInOut'right) and (Now > ChRespDuration) then
      CountGW <= CountGW + 1;
    end if;
  end process Ticker;
  ----------------------------------------------------------------------
  break on  sStep, CountInit, CountGW;

  VwfmStep == sStep;
  if (CountInit > 0) use
    VwfmTxInit == TxMatrixInOut(CountInit);
    VwfmRxInit == RxMatrixInOut(CountInit);
  else
    VwfmTxInit == 0.0;
    VwfmRxInit == 0.0;
  end use;

  if (CountGW > 0) use
    VwfmGW   == AMIwaveInOut(CountGW);
  else
    VwfmGW   == 0.0;
  end use;

end architecture Call_TxRx;
--======================================================================
```

Copy IBIS-AMI results to quantities so that the tool can plot them

IBIS-AMI wrapper C-code sample (load DLLs)

```
//-------------------------------------------------------------------------
// Load DLL files and get the addresses of the AMI functions
//-------------------------------------------------------------------------
hTxDLLfile = LoadLibrary(Tx_DLL_file_name);
if (hTxDLLfile == NULL)
  printf("WARNING:  Tx DLL file [%s] not found.\n", Tx_DLL_file_name);
else {
  printf("INFO:  Tx DLL file [%s] loaded successfully.\n", Tx_DLL_file_name);

  AMI_Tx_Init    = (pAMI)GetProcAddress(hTxDLLfile, "AMI_Init");
  AMI_Tx_GetWave = (pAMI)GetProcAddress(hTxDLLfile, "AMI_GetWave");
  AMI_Tx_Close   = (pAMI)GetProcAddress(hTxDLLfile, "AMI_Close");

  if ((&AMI_Tx_Init == NULL) && (&AMI_Tx_GetWave == NULL) && (&AMI_Tx_Close == NULL))
    printf("WARNING:  Tx DLL file [%s] contains no AMI functions.\n", Tx_DLL_file_name);
  else {
    printf("INFO:  Tx DLL file [%s] contains:\n", Tx_DLL_file_name);
    if (AMI_Tx_Init != NULL)
      printf("      AMI_Init\n");
    if (AMI_Tx_GetWave != NULL)
      printf("      AMI_GetWave\n");
    if (AMI_Tx_Close != NULL)
      printf("      AMI_Close\n");
  }
}
…
…
…
```

IBIS-AMI wrapper C-code sample (Tx Init)

```
//-------------------------------------------------------------------------
// Run Tx AMI_init
//-------------------------------------------------------------------------
// Make the input of Tx_Init the input of Rx_Init
if (Use_Tx_Init_Output == 0)
  memcpy(Rx_impulse_matrix, Tx_impulse_matrix, row_size*(aggressors+1)*sizeof(double));

if (AMI_Tx_Init != NULL) {
  AMI_ReturnVal = 0;
  AMI_ReturnVal = AMI_Tx_Init(Tx_impulse_matrix,
                              row_size,
                              aggressors,
                              sample_interval,
                              bit_time,
                              Tx_parameters_in,
                              &Tx_parameters_out,
                              &Tx_memory_handle,
                              &Tx_msg);
}
else
  printf("WARNING:  The AMI_Init function doesn't exist in the Tx DLL file:  [%s]\n\n", Tx_DLL_file_name);

// Make the output of Tx_Init the input of Rx_Init
if (Use_Tx_Init_Output != 0)
  memcpy(Rx_impulse_matrix, Tx_impulse_matrix, row_size*(aggressors+1)*sizeof(double));
//-------------------------------------------------------------------------
```

IBIS-AMI wrapper C-code sample (Rx Init)

```
//-------------------------------------------------------------------------
// Run Rx AMI_init
//-------------------------------------------------------------------------
if (AMI_Rx_Init != NULL) {
  AMI_ReturnVal = 0;
  AMI_ReturnVal = AMI_Rx_Init(Rx_impulse_matrix,
                              row_size,
                              aggressors,
                              sample_interval,
                              bit_time,
                              Rx_parameters_in,
                              &Rx_parameters_out,
                              &Rx_memory_handle,
                              &Rx_msg);
}
else
  printf("WARNING:  The AMI_Init function doesn't exist in the Rx DLL file:  [%s]\n\n", Rx_DLL_file_name);
//-------------------------------------------------------------------------
// Convolve stimulus with (modified or unmodified) impulse response
//-------------------------------------------------------------------------
// Scale impulse response by sample_interval
if (Use_Rx_Init_Output == 0) {
  // Use output of Tx_Init
  for(i = 0; i < row_size; i++) {
    tmp_imp_resp[i] = sample_interval * Tx_impulse_matrix[i];
  }
}
else {
  // Use output of Rx_Init
  for(i = 0; i < row_size; i++) {
    tmp_imp_resp[i] = sample_interval * Rx_impulse_matrix[i];
  }
}
```

IBIS-AMI wrapper C-code sample (conv.)

```
// Do the rest of the convolution using the sliding window technique
ActualWindowSize = RequestedWindowSize;
for(i = 0;  i < wave_size;  i = i + RequestedWindowSize) {
  if (ActualWindowSize > wave_size - i) {
    ActualWindowSize = wave_size - i;
    printf("Last window size(%d) = %d\n\n", i/RequestedWindowSize+1, ActualWindowSize);
  }

  for(indx = 0; indx < ActualWindowSize; indx++) {
    for(yndx = 0; yndx < row_size; yndx++) {
      tmp_wave[indx+yndx] += wave_in[i+indx] * tmp_imp_resp[yndx];
    }
    last_in = wave_in[i+indx];
    wave_in[i+indx] = tmp_wave[indx];           //Save the output
  }

  //Save the remaining response for the next block of data
  for(indx = 0; indx < row_size; indx++) {
    tmp_wave[indx] = tmp_wave[indx+RequestedWindowSize];
  }
  for( ; indx < tmp_size; indx++) {
    tmp_wave[indx] = 0;
  }
} // End of FOR loop
```

IBIS-AMI wrapper C-code sample (Tx GetWave)

```
//-------------------------------------------------------------------------
// Run Tx AMI_GetWave
//-------------------------------------------------------------------------
if ((AMI_Tx_Init != NULL) && (AMI_Tx_GetWave != NULL)) {
  ActualWindowSize = RequestedWindowSize;
  AMI_ReturnVal = 0;
  for(i = 0;  i < wave_size;  i = i + RequestedWindowSize) {
    if (ActualWindowSize > wave_size - i) {
      ActualWindowSize = wave_size - i;
      printf("AMI_Tx_GetWave last window size(%d) = %d\n", i/RequestedWindowSize+1, ActualWindowSize);
    }
    AMI_ReturnVal = AMI_Tx_GetWave(&wave_in[i],
                                   ActualWindowSize,
                                   clock_times,
                                   &Tx_parameters_out,
                                   Tx_memory_handle);
    if (AMI_ReturnVal != 1) {
      printf("WARNING:  AMI_Tx_GetWave execution ended prematurely.\n\n");
      break;
    }
  } // End of FOR loop
}
else {
  if (AMI_Tx_GetWave == NULL)
    printf("WARNING:  The AMI_GetWave function doesn't exist in the Tx DLL file:  [%s]\n\n", Tx_DLL_file_name);
  else
    printf("WARNING:  The AMI_GetWave function couldn't be executed because the AMI_Init function doesn't exist
 in the Tx DLL file:  [%s]\n\n", Tx_DLL_file_name);
}
```

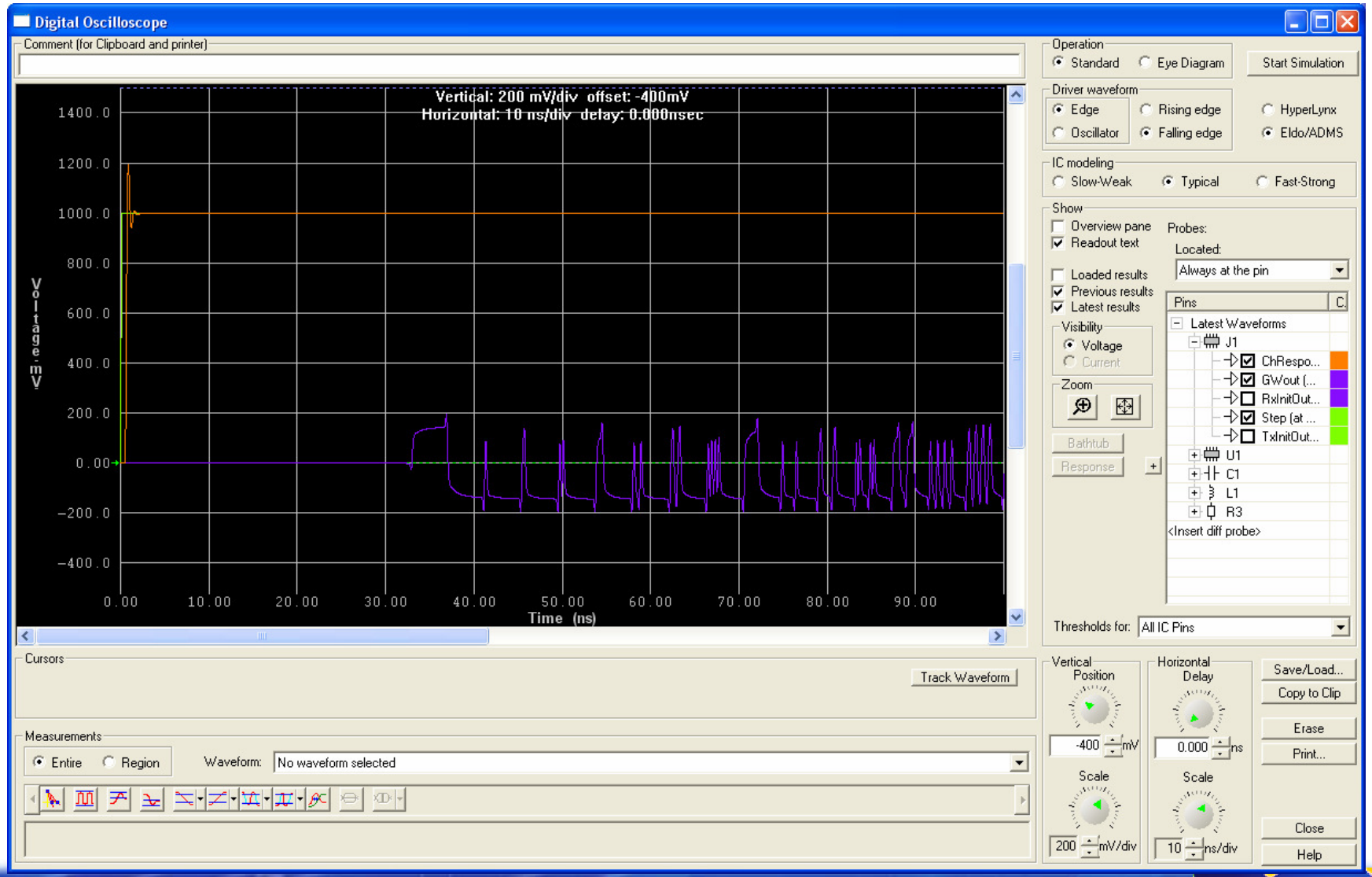
IBIS-AMI wrapper C-code sample (Rx GetWave)

```
//-------------------------------------------------------------------------
// Run Rx AMI_GetWave
//-------------------------------------------------------------------------
if ((AMI_Rx_Init != NULL) && (AMI_Rx_GetWave != NULL)) {
  ActualWindowSize = RequestedWindowSize;
  AMI_ReturnVal = 0;
// This IF statement is here for debugging purposes, because this Rx model's
// GetWave function cannot be called with the sliding window algorithm.
  if (0)
  {
    for( i = 0;  i < wave_size;  i = i + RequestedWindowSize ) {
      if (ActualWindowSize > wave_size - i) {
        ActualWindowSize = wave_size - i;
        printf("AMI_Rx_GetWave last window size(%d) = %d\n", i/RequestedWindowSize+1, ActualWindowSize);
      }
      AMI_ReturnVal = AMI_Rx_GetWave(&wave_in[i],
                                     ActualWindowSize,
                                     clock_times,
                                     &Rx_parameters_out,
                                     Rx_memory_handle);
      if (AMI_ReturnVal != 1) {
        printf("WARNING:  AMI_Rx_GetWave execution ended prematurely.\n");
        break;
      }
    } // End of FOR loop
  }
  else
  {
    AMI_ReturnVal = AMI_Rx_GetWave(wave_in,
                                   RxWaveSize,
                                   clock_times,
                                   &Rx_parameters_out,
                                   Rx_memory_handle);
  }
}
```

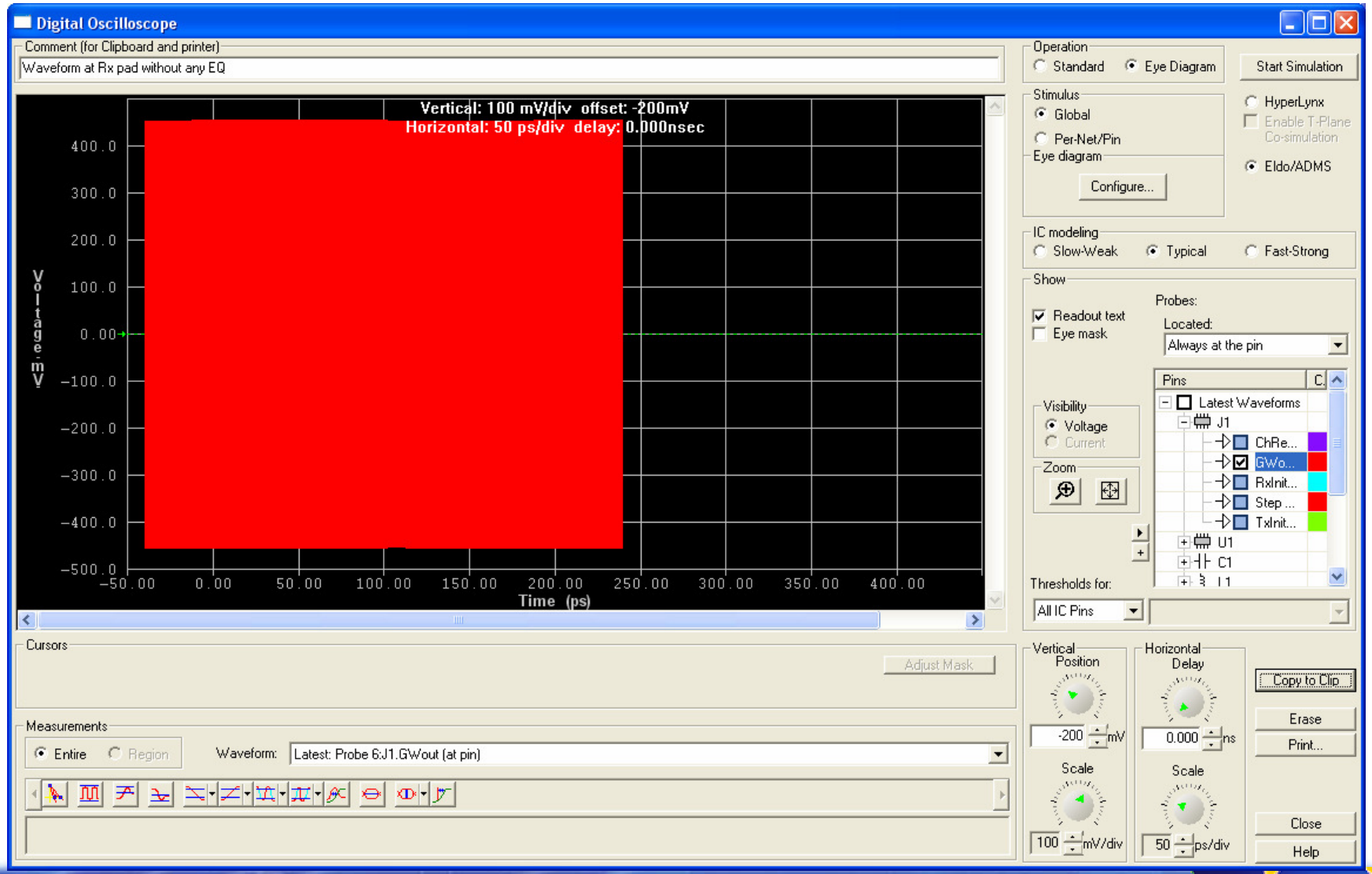
IBIS-AMI wrapper C-code sample (Close)

```
else {
  if (AMI_Tx_GetWave == NULL)
    printf("WARNING:  The AMI_GetWave function doesn't exist in the Rx DLL file:  [%s]\n\n", Rx_DLL_file_name);
  else
    printf("WARNING:  The AMI_GetWave function couldn't be executed because the AMI_Init function doesn't exist
 in the Rx DLL file:  [%s]\n\n", Rx_DLL_file_name);
}
//-------------------------------------------------------------------------
// Run Tx AMI_Close
//-------------------------------------------------------------------------
if ((AMI_Tx_Init != NULL) && (AMI_Tx_Close != NULL)) {
  AMI_ReturnVal = 0;
  AMI_ReturnVal = AMI_Tx_Close(Tx_memory_handle);
}
else {
  if (AMI_Tx_Close == NULL)
    printf("WARNING:  The AMI_Close function doesn't exist in the Tx DLL file:  [%s]\n\n", Tx_DLL_file_name);
  else
    printf("WARNING:  The AMI_Close function couldn't be executed because the AMI_Init function doesn't exist
 in the Tx DLL file:  [%s]\n\n", Tx_DLL_file_name);
}
//-------------------------------------------------------------------------
// Run Rx AMI_Close
//-------------------------------------------------------------------------
if ((AMI_Rx_Init != NULL) && (AMI_Rx_Close != NULL)) {
…
…
…
}
```

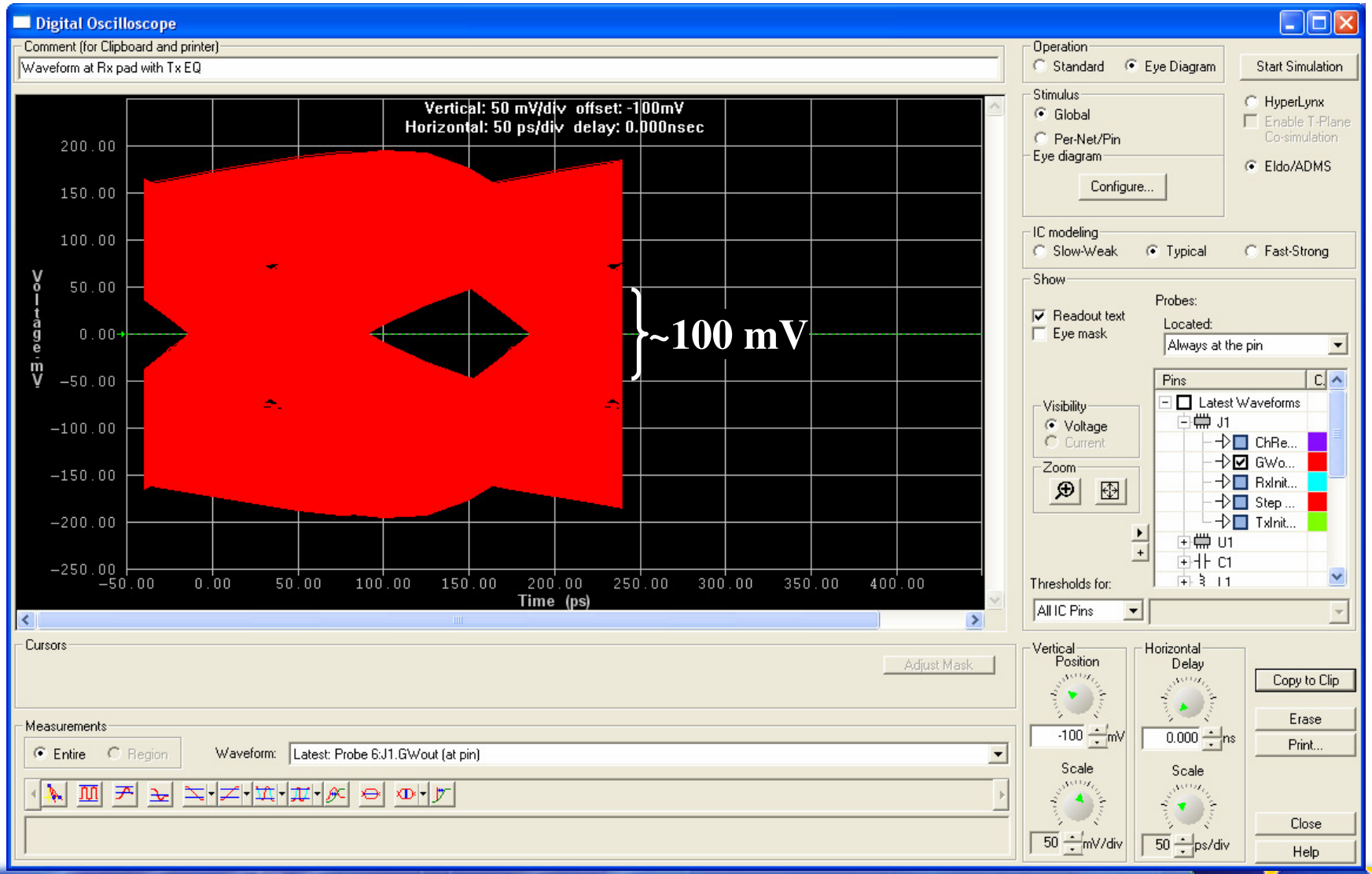
Waveform results of the example



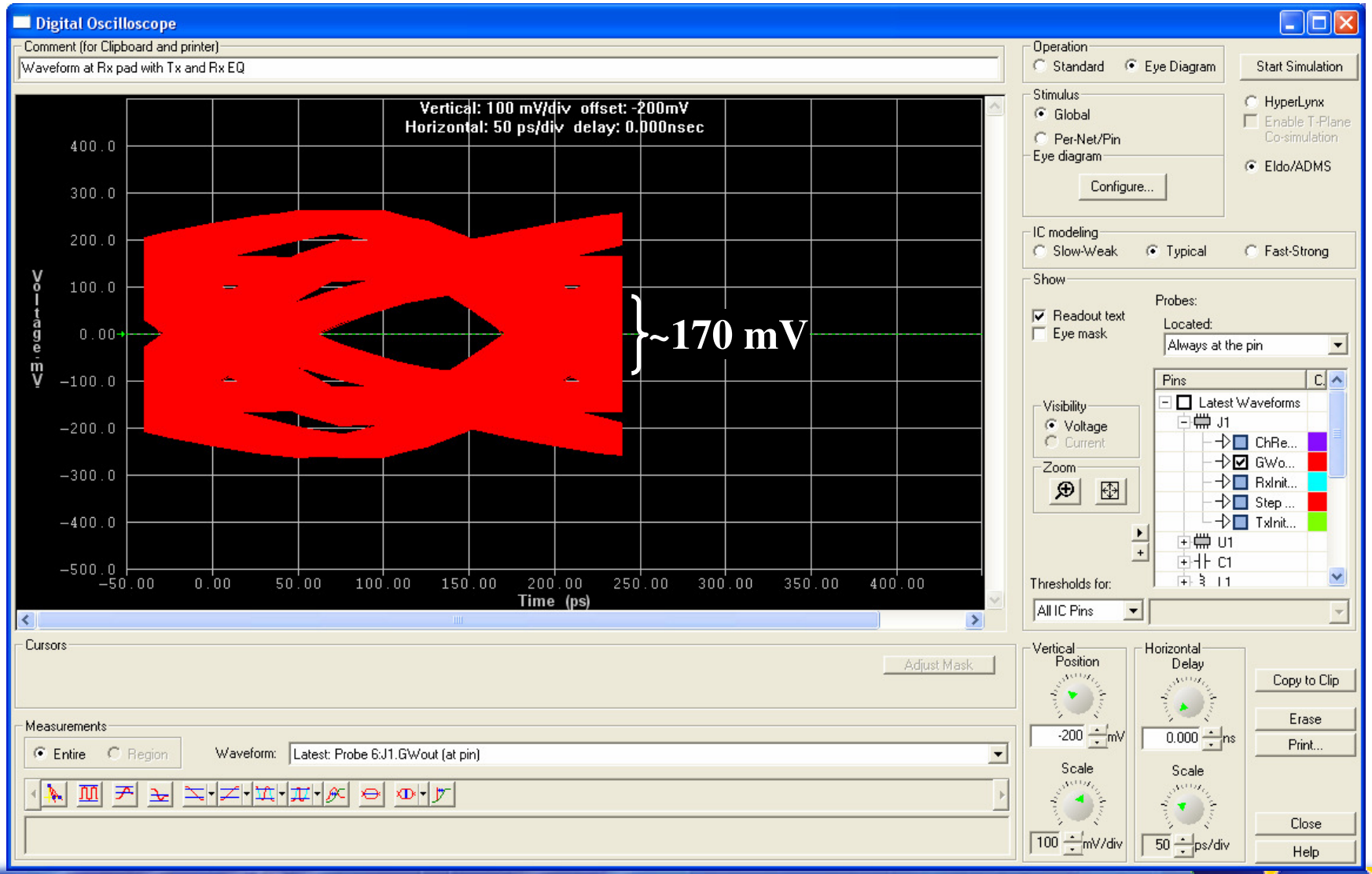
100k bits of PRBS 22 over 20 µs – no EQ



100k bits of PRBS 22 over 20 µs – Tx EQ

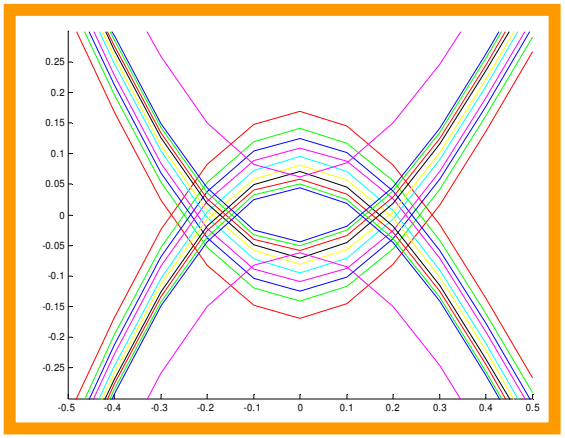


100k bits of PRBS 22 over 20 µs – Tx & Rx



Current IBIS-AMI Support

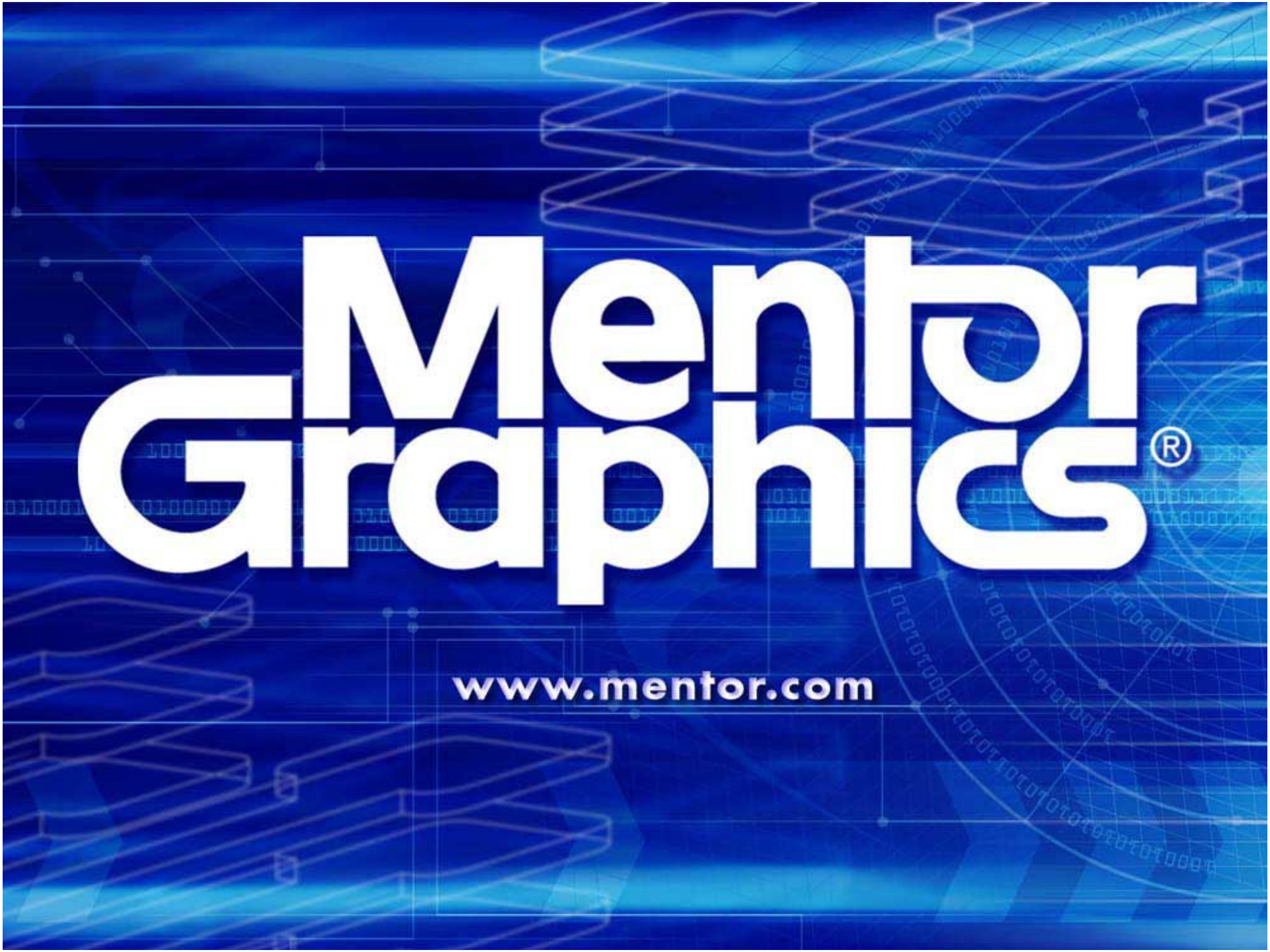
IBIS Summit, DAC, June 2008



1. Taking care of some unfinished business
   - Updated performance benchmarks
   - Sliding window algorithm in VHDL-AMS
2. Attributes in VHDL-AMS
3. An IBIS-AMI example using the FOREIGN attribute
4. Conclusions – future work

Conclusions – future work

- VHDL-AMS is a viable option for algorithmic modeling
  - its performance is fast, and the language is flexible
- IBIS-AMI models (DLLs) are fully supported in any Mentor tool that has VHDL-AMS capabilities
  - direct execution of the compiled IBIS-AMI models
- Any programming language, capable of producing executables can be supported through VHDL-AMS
  - Matlab, Visual Basic, Perl, you name it
  - execution speed of the compiled code does not suffer any degradation through VHDL-AMS since it is executed externally
- An IBIS (v5.0) parser will be needed to automate the IBIS parameter extraction for the IBIS-AMI models
  - this will happen most likely after the IBIS v5.0 spec has been ratified


Mentor Graphics®
www.mentor.com