Contents

5. On-chip Bus: AHB-Lite

5.1. Overview

On-chip bus is the communication backbone in SoCs. Data and control messages are passed through on-chip bus to all computational units and peripheral devices connected on the bus. This lab demonstrates on-chip bus using a simple AHB system as an example. This single-master AHB system is similar to the AHB system which resides on Logic Module. You will learn the followings in this lab:

1. Understand how a simple AHB system works.
2. Be able to add a new AHB slave to an AHB system.
3. Know the check list to check bus protocol compliance

5.2. Background information

5.2.1. Typical AMBA AHB system

The Advanced Microcontroller Bus Architecture (AMBA) specification defines an on-chip communications standard for designing high-performance embedded microcontrollers.

Three distinct buses are defined within the AMBA specification:

- The Advanced High-performance Bus (AHB)
- The Advanced System Bus (ASB)
- The Advanced Peripheral Bus (APB).

AMBA AHB implements the features required for high-performance, high clock frequency systems including:

- burst transfers
- split transactions
- single-cycle bus master handover
- single-clock edge operation
- non-tristate implementation
- wider data bus configurations (64/128 bits).

The APB is part of the AMBA hierarchy of buses and is optimized for minimal power consumption and reduced interface complexity. The AMBA APB appears as a local secondary bus that is encapsulated as a single AHB or ASB slave device. APB provides a low-power extension to the system bus which builds on AHB or ASB signals directly. A typical AMBA AHB system design contains the following components:

• AHB master

A bus master is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.

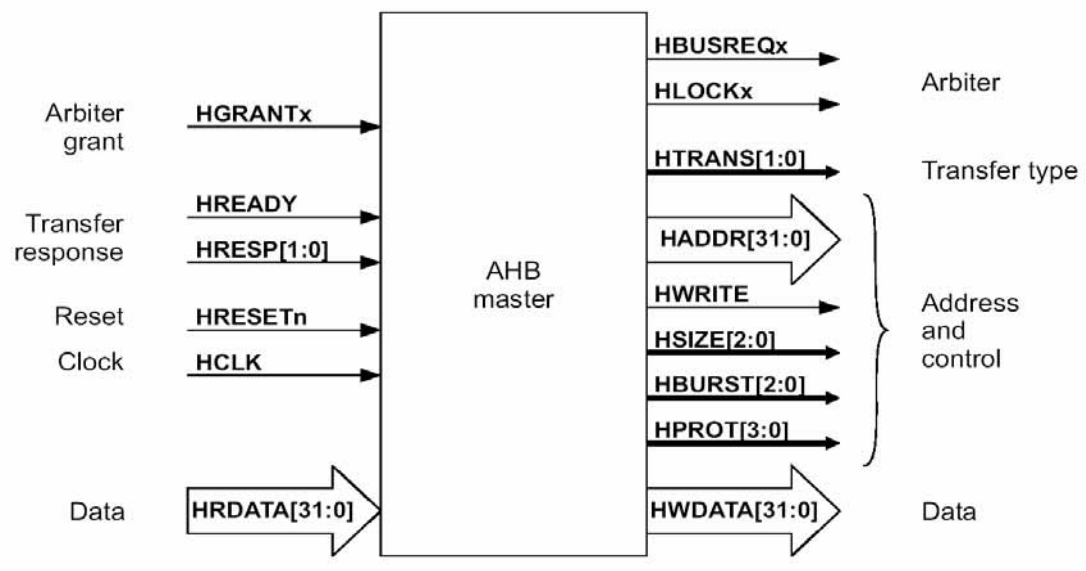


Figure 1 AHB bus master interface

• AHB slave

A bus slave responds to a read or write operation within a given address-space range. The bus slave signals back to the active master the success, failure or waiting of the data transfer.

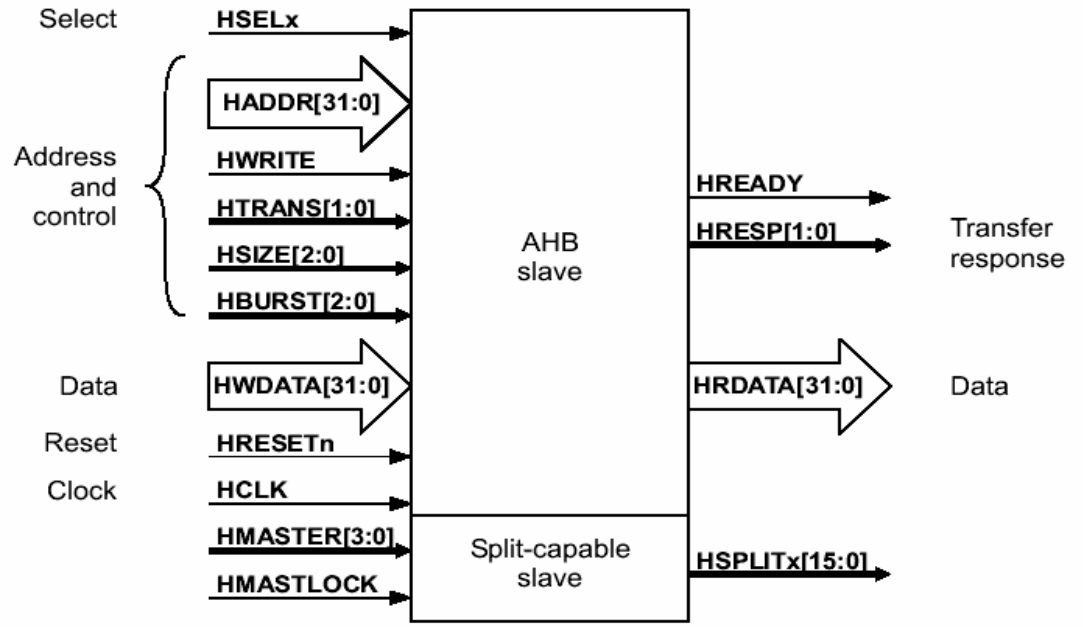


Figure 2 AHB bus slave interface

• AHB arbiter

The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers. Even though the arbitration protocol is fixed, any arbitration algorithm, such as highest priority or fair access can be implemented depending on the application requirements. An AHB would include only one arbiter, although this would be trivial in single bus master systems.

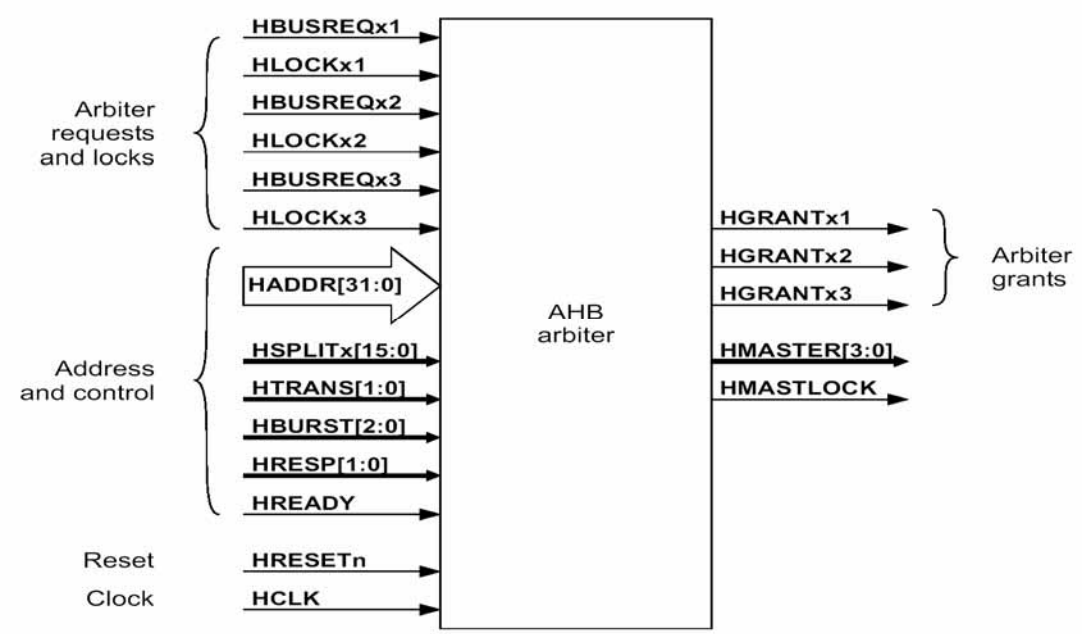


Figure 3 AHB arbiter interface diagram

• AHB decoder

The AHB decoder is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer. A single centralized decoder is required in all AHB implementations.

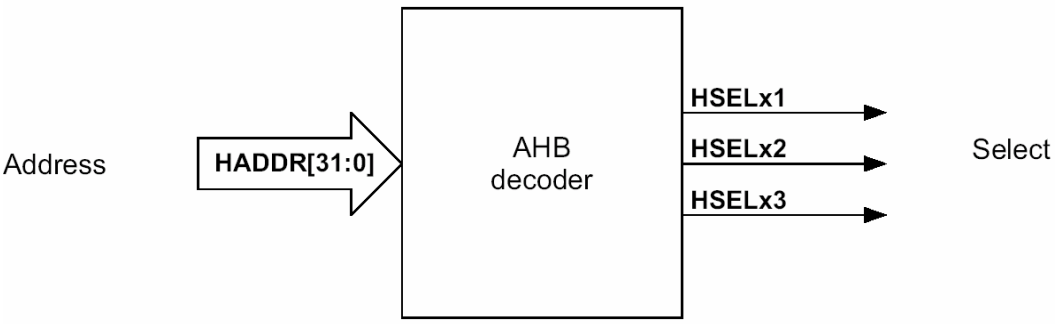


Figure 4 AHB decoder interface diagram

The interconnections of multiplexor and decoder for HADDR, HWDATA, HRDATA, and HSEL_x signals are illustrated in Figure 5 and Figure 6.

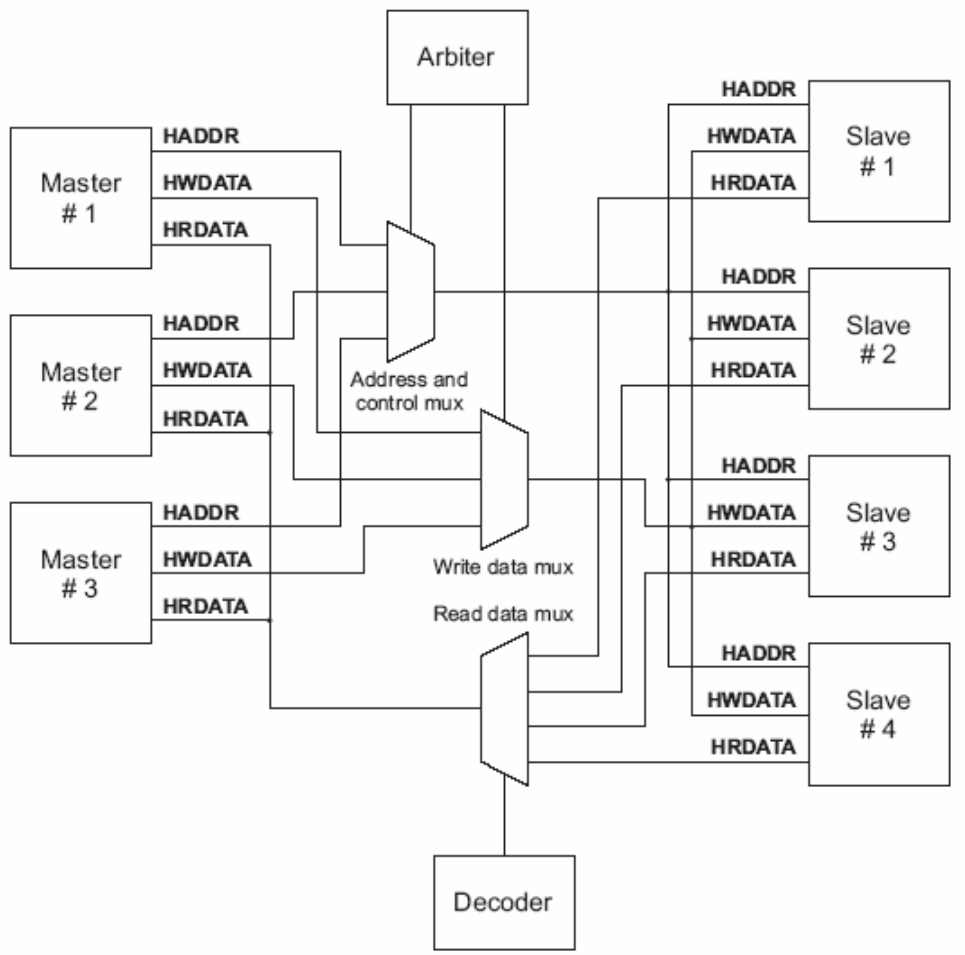


Figure 5 Multiplexor interconnections of address and data

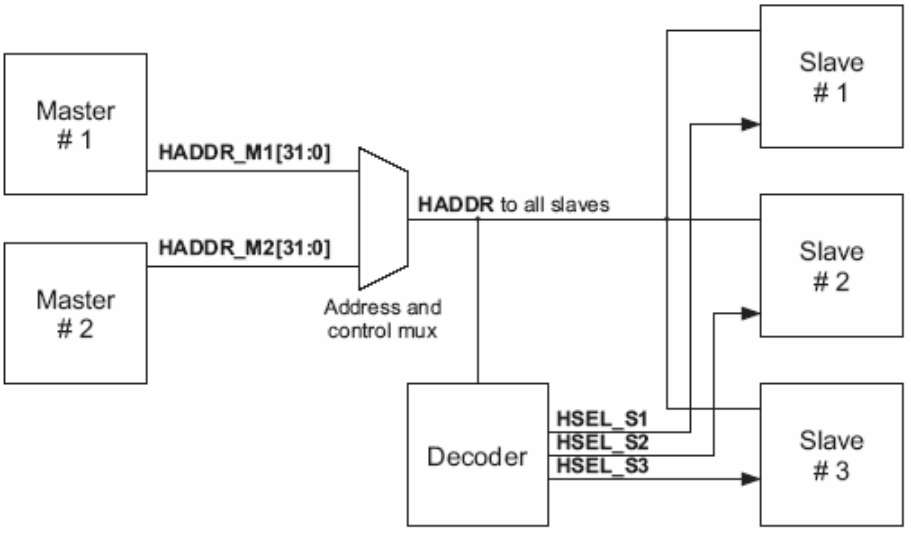


Figure 6 Address decoding system and the slave select signals

5.2.2. Single-master AHB system: AHB-Lite

A simple single-master AHB system (AHB-Lite) is modified from the AHB system on ARM Logic Module as an example in this lab. The original tri-state bidirectional signals (SDATA, HDATA, HREADY) in Logic Module's AHB system are modified into pairs of input and output signals. AHB-Lite does not support burst transfer because it was originally designed to use only ZBT(Zero Bus Turn-around) SRAM, therefore burst transfer is not necessary. Moreover, since there's only one single master in AHB-Lite, therefore there is no arbiter module. Hence bus request and bus grant signal are also not necessary in single-master systems like AHB-Lite.

The modules in AHB-Lite and their descriptions are listed in Table 1. All modules except AHB_HC_master, SRAM_8X4X4096, and RA1SH are within AHBAHBTop. AHBAHBTop is modified from the original AHB system adopted in ARM Logic Module. Figure 7 shows the block diagram of AHB-Lite.

Table 1 Modules in AHB-Lite

Modules	Description
AHB_HC_master	Master behavioral module in which test pattern is written. AHB read and write accesses are initiated in this module.
AHBAHBTop	The top-level HDL that instantiate all of the high-speed peripherals, decoder, and all necessary support and glue logic to make a working system. The files are named so that, for example, ASBAHBTop.v is the top level for AHB peripherals connected to an ASB system bus.
AHBDecoder	The decoder block provides the high-speed peripherals with select lines. These are generated from the address lines and the module ID (position in stack) signals from the motherboard. The decoder blocks also contain the default slave peripheral to simplify the example structure.
AHBMuxS2M	This is the AHB multiplexor that connects the read data buses from all of the slaves to the AHB master(s).
AHBZBTRAM	High-speed peripherals require that SSRAM controller block supports word, halfword, and byte operations to the SSRAM on the logic module.
SRAM_8X4X4096	This is the SRAM module of 16KB size. It has 4 SRAM banks where each bank is 8-bit in width and has 4096 entries. This SRAM module is connected to AHBZBTRAM memory controller.
RA1SH	SRAM behavioral module for each SRAM bank. The size is 8-bit x 4096.
AHB2APB	This is the bridge blocks required to connect APB peripherals the the high-speed AMBA AHB bus. They produce the peripheral select signals for each of the APB peripherals.
AHBAPBSys	The components required for an APB system are instantiated in this block. These include the bridge and the APB peripherals. This file also multiplexes the APB peripheral read buses and concatenates the interrupt sources to feed into the interrupt controller peripheral.
APBRegs	The AOB register peripheral provides memory mapped registers that you can use to: Configure the two clock generators Write to the user LEDs Read the user switch inputs.

	It also latches the pressing of the push button to generate an expansion interrupt.
APBIntcon	The APB interrupt controller contains all of the standard interrupt controller registers and has an input port for four APB interrupts. Four software interrupts are implemented.

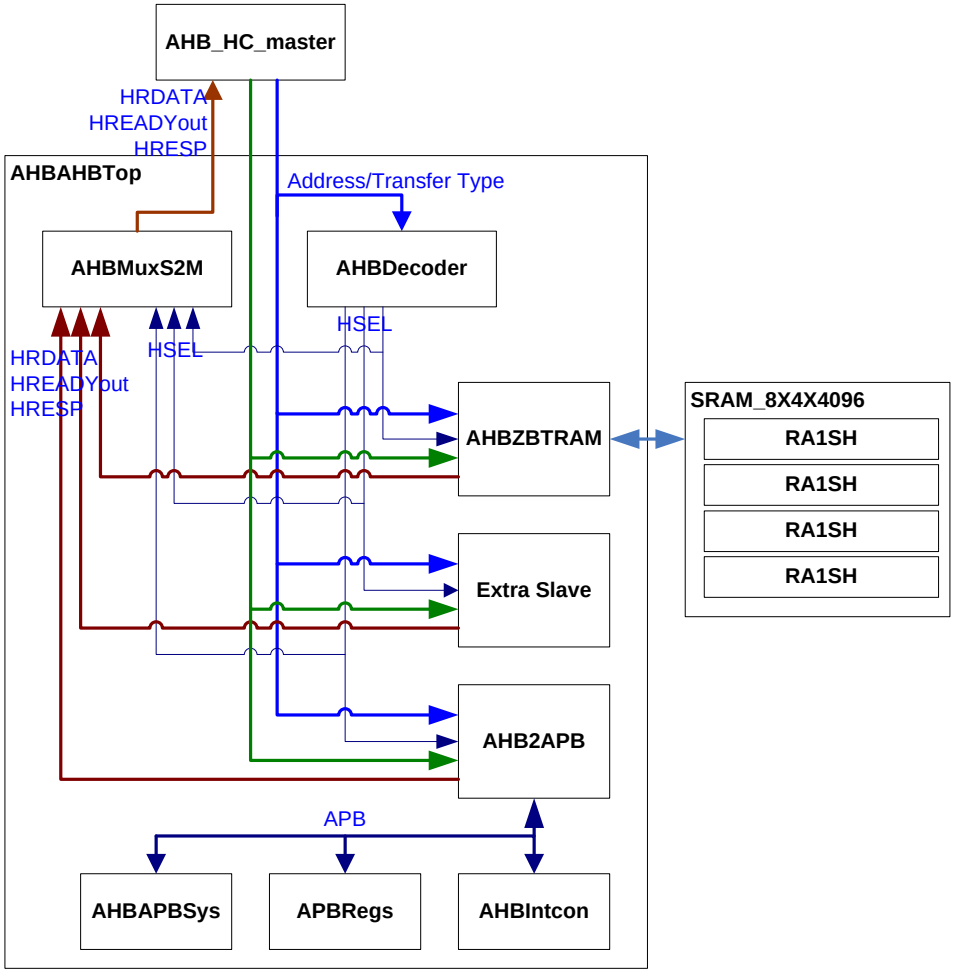


Figure 7 AHB-Lite block diagram

When an access is initiated by master, the control information containing address and transfer type are first sent to AHBDecoder in control phase. AHBDecoder decodes the address and identifies which slave device is to be accessed according to predefined system memory map. The slave device to be accessed will be signaled by HSEL signal generated by AHBDecoder. Once the slave device is selected, it decodes the address offset within its memory space to determine which register or memory location in the device is to be accessed. If the selected device is ready to output the read data or input the write data, data phase starts. To return information from slave to master, read data and response from selected slave device are sent to AHBMuxS2M for multiplexing. AHBMuxS2M multiplexes the read data and responses from multiple slave devices into one read data HRDATA and one response HRESP.

HSEL signals from AHBDecoder are used in AHBMuxS2M to determine which read data and response are sent to master.

To add a new slave into an AHB system, the memory range to map this new slave must be first be defined. After the new memory map is defined, AHBDecoder and AHBMuxS2M must be modified to fit the new memory map.

5.2.3. AHB Compliance Verification

Upon having a device with AHB slave interface, it is important to verify its compliance with AHB bus protocol. The simplest way and also the least reliable way to check a device's compliance is to install it into an already verified AHB system. If nothing went wrong during the test, we might "claim" its compliance. However, checking for AHB compliance in this manner requires a clear, well defined, and complete check list. Otherwise doubts in the claim of compliance would arise due to lack of proof.

Currently there are several commercial EDA tools to help verifying the compliance. These compliance checking tools either adopt formal verification techniques such as state-traversal or adopt simulation-based checking. The advantage of using formal techniques is the 100% confidence of protocol compliance due to their exhaustive nature. However, the overhead of formal techniques are the space required to store intermediate results. Simulation-based compliance checking evaluates the coverage of protocol check list from simulation of various test patterns. Once the coverage reaches 100%, claiming the device is AHB compliant is more persuasive. However, 100% coverage of the check list does not 100% guarantee that there won't be error, since some corner cases might not be tested.

Synopsys Designware provides Verification IP (VIP) written with Vera and verilog which can be used for simulation-based compliance testing. Designware VIP provides AHB-monitor which observes whether the transaction on the bus comply with AMBA AHB protocol. It also supports functional coverage which enables verification engineer to know which transactions are not verified.

Synopsys Designware is supported by CIC. You can install Designware VIP and Vera in your workstations. Please refer related documents provided in Designware and Vera.

In this lab, the check list will be manually written. You'll be ask to write the check list for a typical AHB-Lite system and a typical AHB-system without split/retry.

5.3. Instructions

The Verilog code describes AHB-Lite on the first Logic Module (HRID=4'b1110). The paths and descriptions of each file in this lab are listed in Table 2.

The test pattern from master to drive read/write access to AHB-Lite is written in AHB_HC_master module. If you intend to modify test pattern from master, modify the content of AHB_HC_master.

SRAM_8X4X4096.v defines SRAM memory module. AHBZBTSRAM memory controller is connected to this 16KB SRAM module. This memory ranges from 0xC2000000 to 0xC2000FFF, but the memory space from 0xC2001000 to 0xC20FFFFF is 16KB aliases of the original 16KB address. For instance, 0xC2001000 ~ 0xC2001FFF is alias to 0xC2000000 ~ 0xC2000FFF.

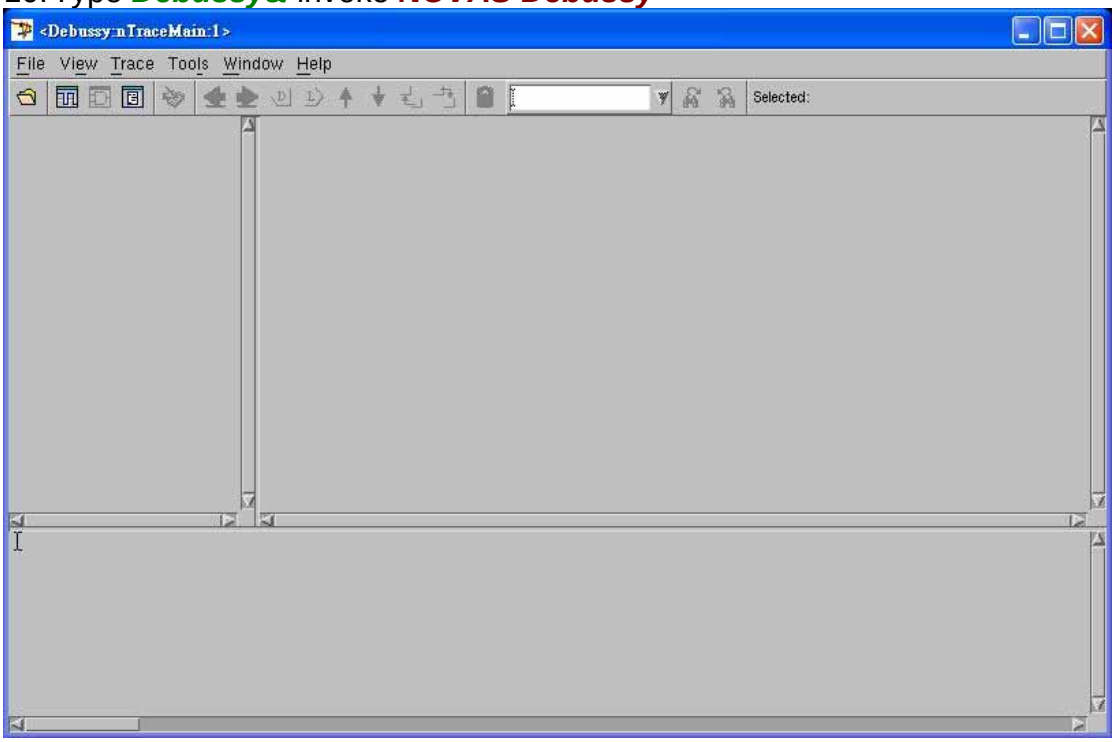
Table 2 File path and description

Files	Path	Description
AMBA_declare.v	/beh	AMBA related predefined keywords
AHB_HC_master.v	/beh	AHB_HC_master behavioral module
SRAM_8X4X4096.v	/beh	16KB SRAM module
RA1SH.v	/beh	SRAM behavioral model
AHB_Testbench.v	/beh	AHB_Testbench top module including AHB_HC_master, SRAM_8X4X4096, and RA1SH
AHBAHBTop.v	/rtl	AHBAHBTop modified module
AHBDecoder.v	/rtl	AHBDecoder module
AHBMuxS2M.v	/rtl	AHBMuxS2M module
AHBZBTRAM.v	/rtl	AHBZBTRAM module
AHB2APB.v	/rtl	AHB2APB module, it is also a slave
AHBAPBSys.v	/rtl	AHBAPBSys module
APBRegs.v	/rtl	APBRegs module
APBIntcon.v	/rtl	APBIntcon module
MyIP.v	/rtl	Slave to be added into AHB-Lite
AHB_Testbench.f	/verif	File lists of the whole test bench
LM_AHBAPB.f	/verif	File lists of AHBAHBTop and its sub-modules

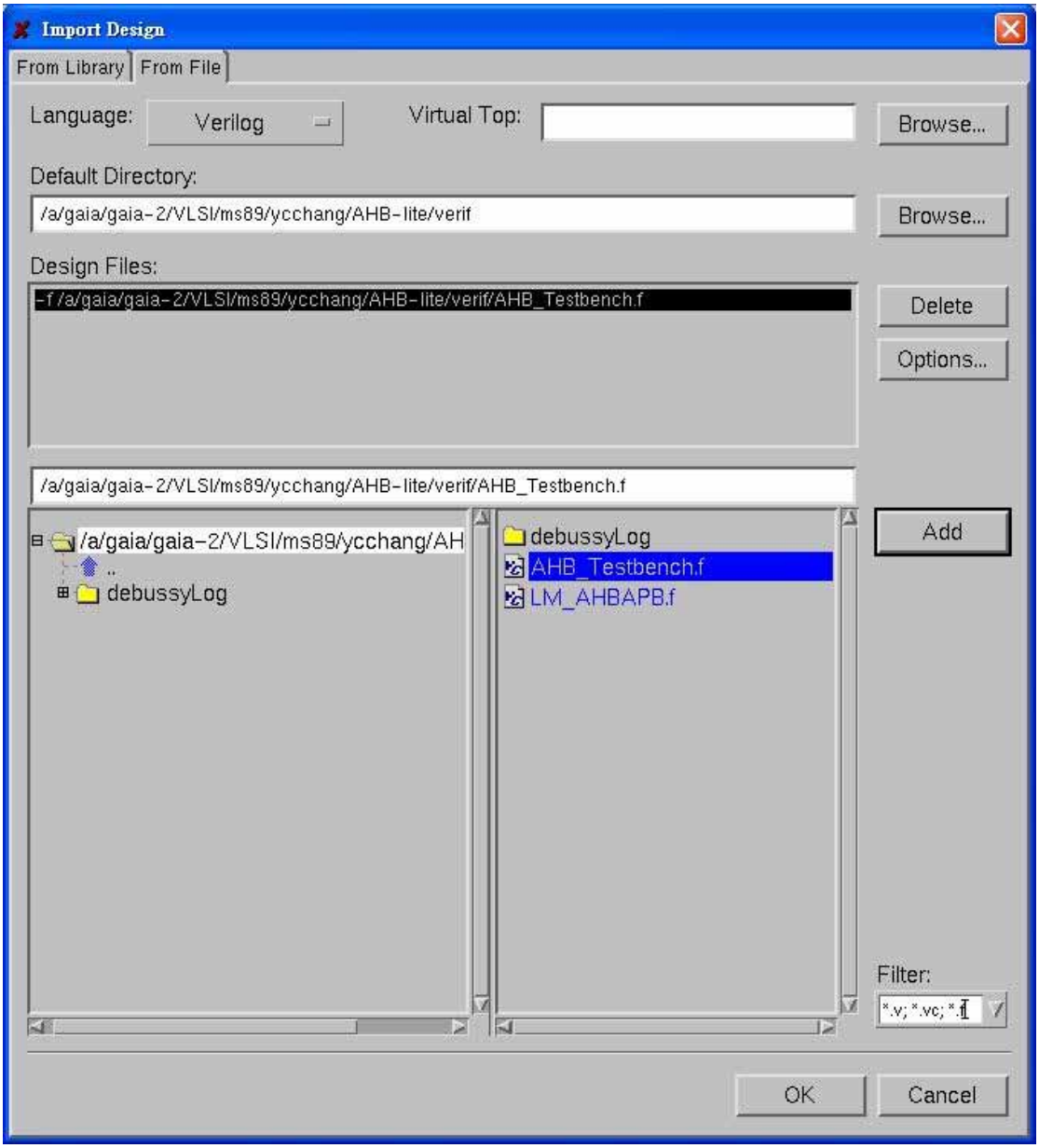
5.3.1. Running AHB-Lite example

1. In WindowsXP, Telnet ***eeMM.ee.nctu.edu.tw***, MM can be 01~31
2. Login using: **soclabNN**, NN is your team number. Set the password for your account.
3. Download **AHB-Lite.tar.Z** and put into your account directory.
4. Restart into Linux, and login using your account and password.
5. In your account directory, type ***tar –zxvf AHB-Lite.tar.Z*** to extract the package.
6. After extraction, a directory **AHB-Lite** is created.
7. Type ***cd AHB-Lite***, type ***ls*** and you'll see 3 directories : **beh**, **rtl**, **verif**

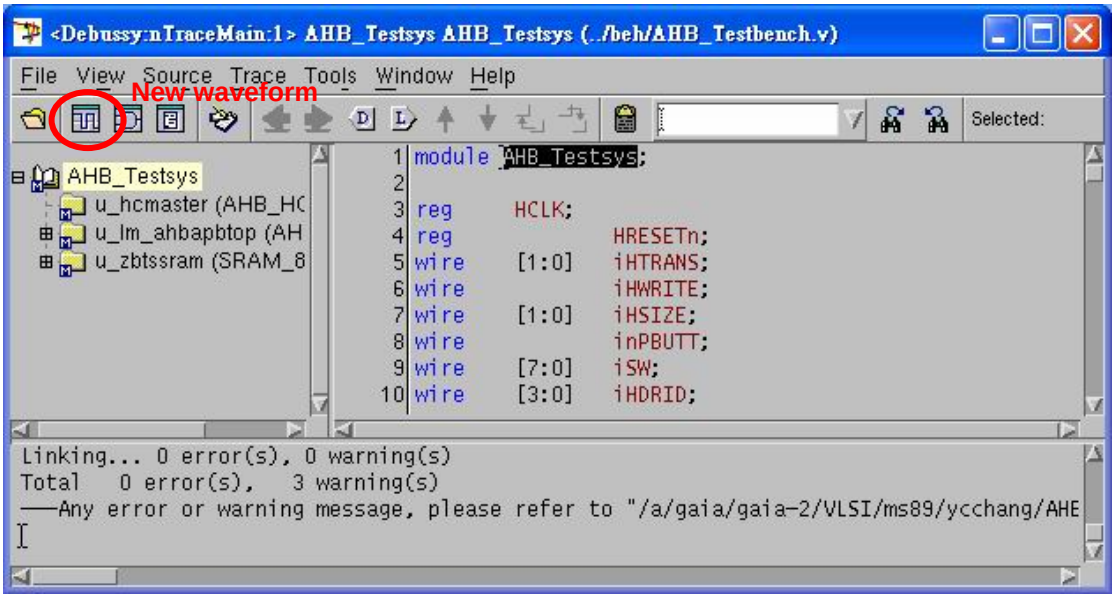
8. Type ***cd verif*** to go into **verif** directory, type ***ls*** to list the files in this directory.
9. Type ***verilog -f AHB_Testbench.f*** to run the simulation. After simulation, you will find **LM_AHBAPB_test.fsdb**, this is the waveform dump file after simulation.
10. Type ***Debussy&*** invoke ***NOVAS Debussy***



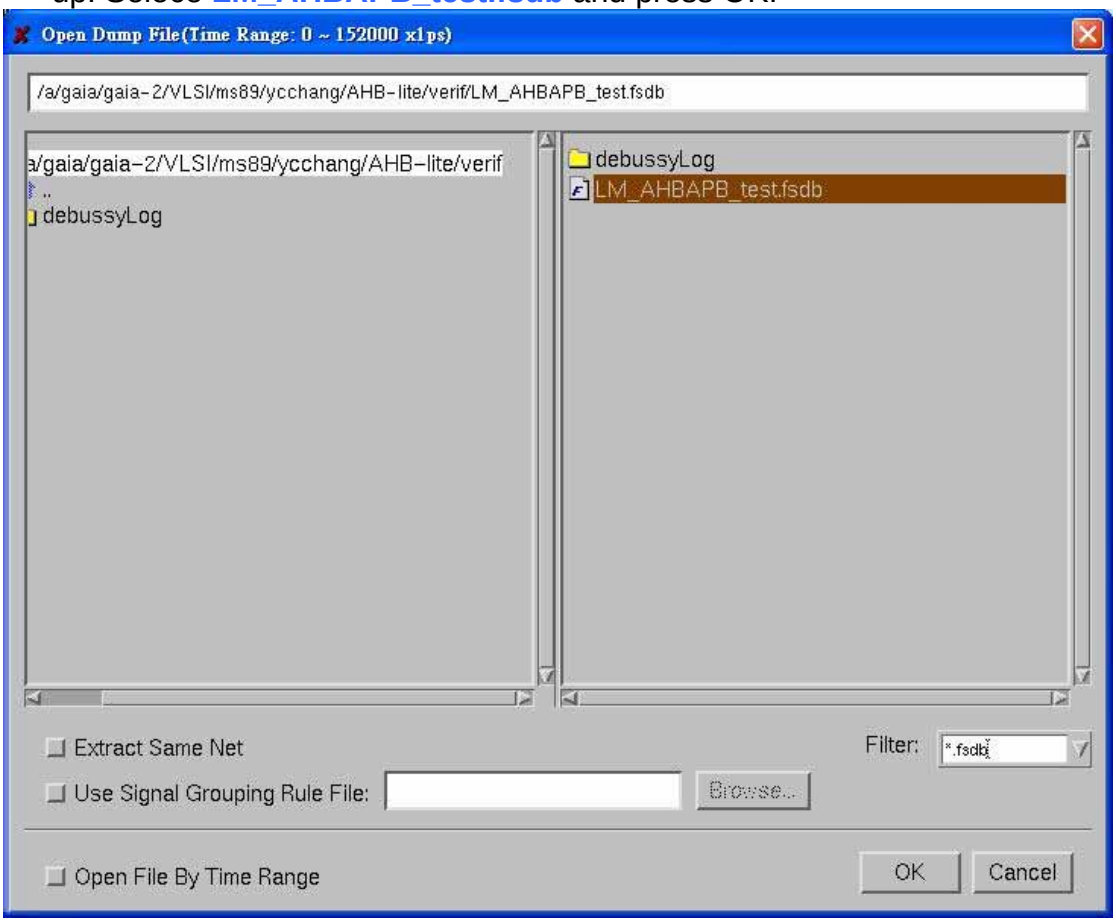
11. In Debussy, ***File→Import Design***, an ***Import Design*** window appears.
12. Click ***From File*** tab, then select **AHB_Testbench.f** and click ***Add*** button. The selected **AHB_Testbench.f** will be added into Design File field. Press ***OK*** when you are done.



13. Click on ***New Waveform*** button in Debussy. **Waveform Viewer** window will appear.

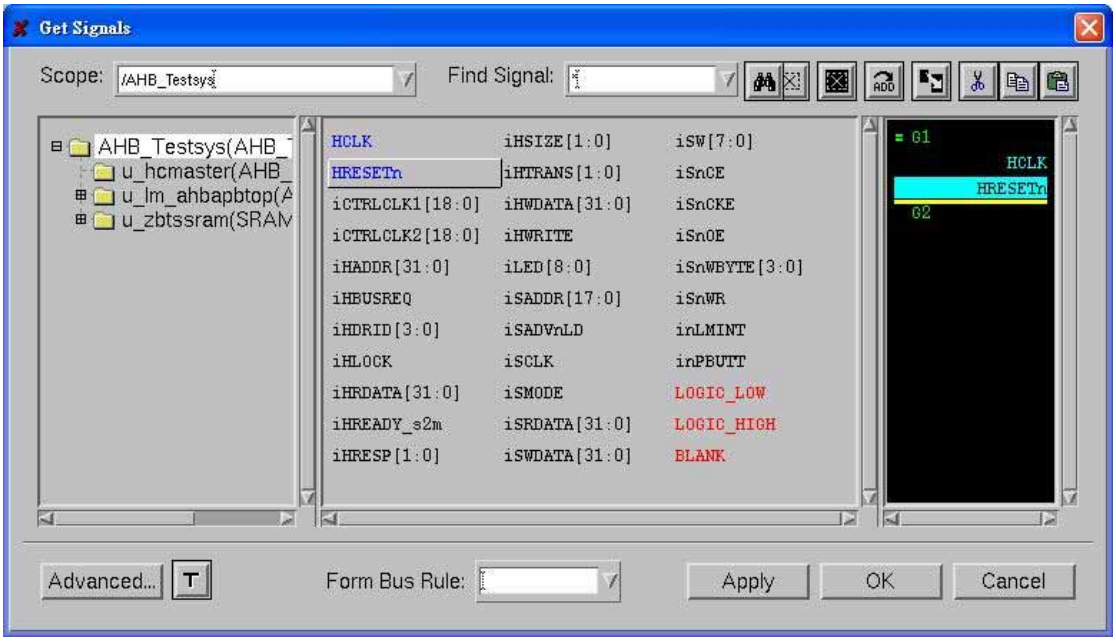


14. In Waveform Viewer, ***File→Open***, an ***Open Dump File*** window will show up. Selece **LM_AHBAPB_test.fsdb** and press OK.

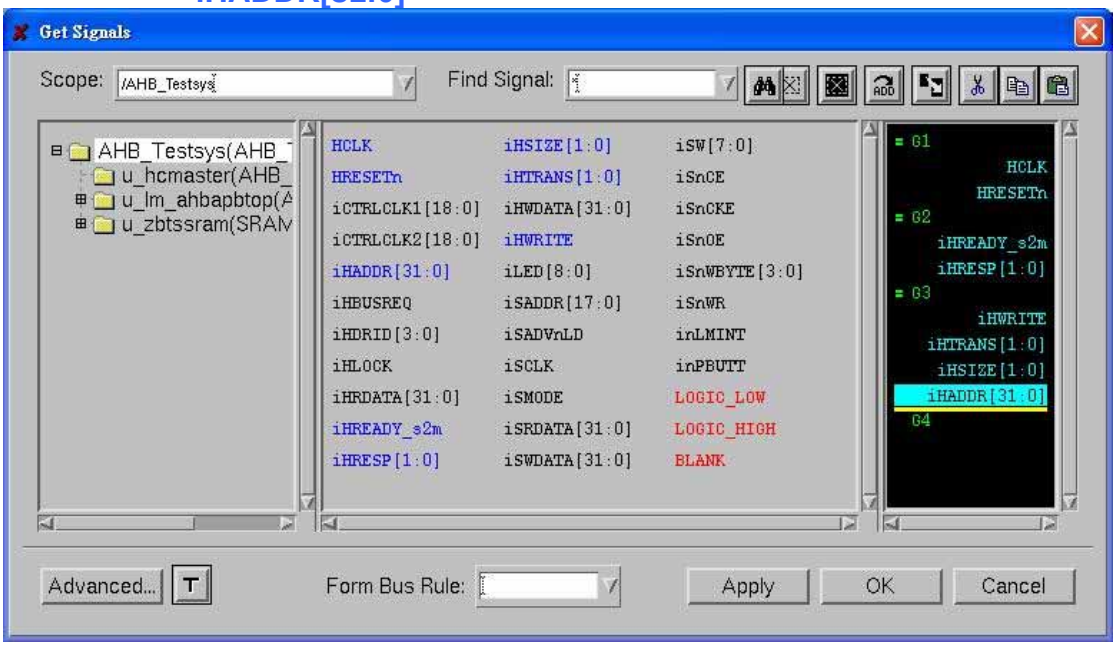


15. ***Signal→Get signals*** to add signals. A ***Get Signals*** window appears, double click on the following signals to add to waveform.

    HCLK
    HRESETn

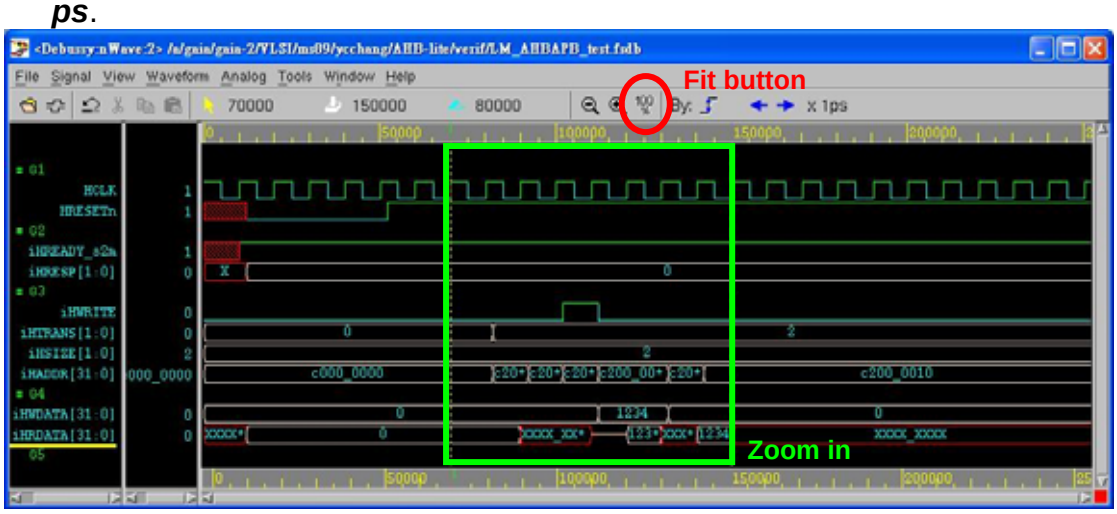


16. Middle mouse click on ***G2***. Use middle mouse button to change to different group. Add following signals to **G2**.
    iHREADY_s2m
    iHRESP[1:0]
17. Add following signals to ***G3***
    iHWRITE
    iHTRANS[1:0]
    iHSIZE[1:0]
    iHADDR[31:0]



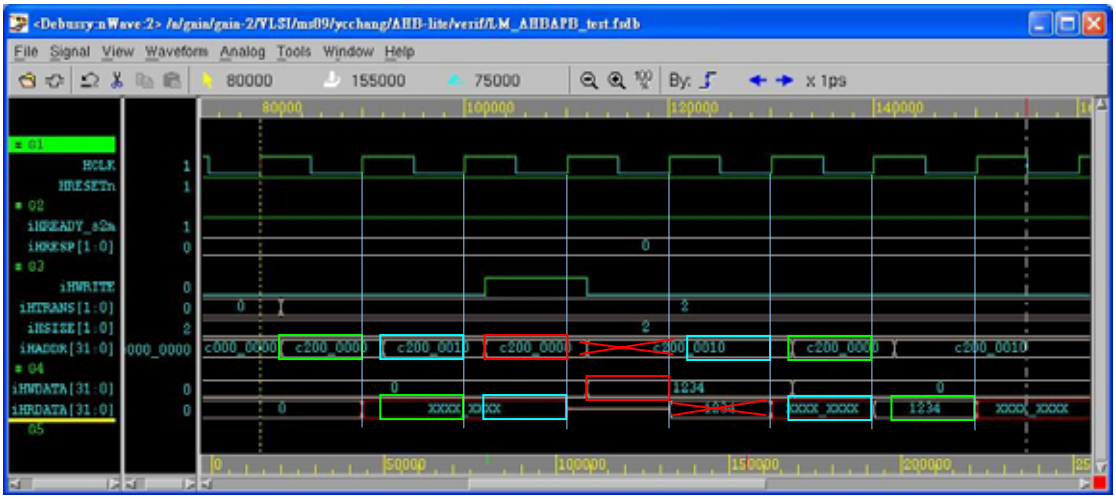
18. Add following signals to ***G4*** and then press OK. The waveform of selected signals will appear.
    iHWDATA
    iHRDATA

19. Press ***'f'*** or the ***fit button***. Then zooming in from ***t=70000 ps*** to ***t=150000 ps***.



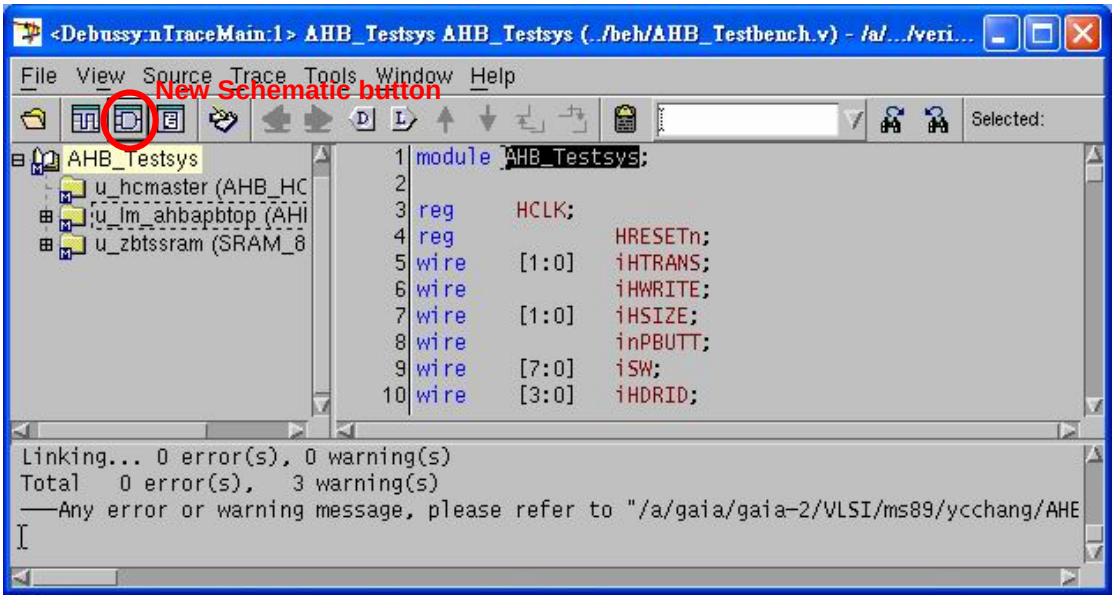
20. Observe read and write accesses. The memory accesses are listed below. Note that the SRAM model in this lab does not support read access immediately after a write (dead cycle is introduced unless ZBT SRAM is used), therefore the ***4th access*** cannot acquire the requested data.

No.	HWRITE	HSIZE	HADDR	HRDATA/ HWDATA	Remarks
1	Read	32-bit	0xc2000000	0xXXXXXXXX	
2	Read	32-bit	0xc2000010	0xXXXXXXXX	
3	Write	32-bit	0xc2000000	0x00001234	
4	Read	32-bit	0xc2000010	0x00001234	SRAM cannot read immediately after write for ZBT SRAM controller.
5	Read	32-bit	0xc2000010	0xXXXXXXXX	
6	Read	32-bit	0xc2000000	0x00001234	
7	Read	32-bit	0xc2000010	0xXXXXXXXX	



21. You can use ***Debussy Schematic Viewer*** to view the schematic of AHB-Lite. In Debussy Main window, click **AHB_Testsys** to select the top module, then click ***New Schematic button*** or ***Tools→New Schematic***.

How to use Debussy is beyond the scope of this lab. If you are interested in learning Debussy, refer to Debussy related documents.



5.4. Exercise

Add a new slave MyIP from **/rtl/MyIP.v** into AHB-Lite system. The base address for MyIP is ***0xc2100000***.

Then ***make a check list*** to check AHB-Lite compliance of MyIP. Please type it down and store as a text file. Please ask the TA to review your check list. Two checks are provided as an example:

- ☐ Single 32-bit read.
- ☐ Single 32-bit read after write

Write test pattern by modifying **AHB_HC_master** to check for AHB-Lite compliance according to the check list you made.

Optional: List the check list for a regular AHB slave device.

5.5. Reference

1. AMBA Specification, Rev. May, 2.0, 1999.
2. High-Speed Single-Port SRAM (HS-SRAM-SP) Generator User Manual, Artisan Components Inc., Release 4.0, Aug. 2000.
3. Debussy User Guide and Tutorial, NOVAS Software Inc., Sept. 2002.
4. Compatibility of Network SRAM and ZBT SRAM, Mitsubishi LSIs Application Note (AP-S001E), Rev. C, Renesas Tech. Corp., Sept. 2002.
5. DesignWare AHB Verification IP Databook, ver. 2.0a, Synopsys Inc., July 2002.
6. VMT User Manual, Release 2.0a, Synopsys Inc., July, 2002.
7. Vera User Guide, ver. 5.1, Synopsys Inc., June, 2002.

5.6. Appendix

AHB signals are listed in Table 3.

Table 3 AMBA AHB signals

Name	Source	Description
HCLK Bus clock	Clock source	This clock times all bus transfers. All signal timings are related to the rising edge of **HCLK.**
HRESETn Reset	Reset controller	Reset controller. The bus reset signal is active LOW and is used to reset the system and the bus. This is the only active LOW signal.
HADDR[31:0] Address bus	Master	The 32-bit system address bus.
HTRANS[1:0] Transfer type	Master	Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.
HWRITE Transfer direction	Master	When HIGH this signal indicates a write transfer and when LOW a read transfer.
HWRITE Transfer direction	Master	When HIGH this signal indicates a write transfer and when LOW a read transfer.
HSIZE[2:0] Transfer size	Master	Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit). The protocol allows for larger transfer sizes up to a maximum of 1024 bits.
HBURST[2:0] Burst type	Master	Indicates if the transfer forms part a burst. Four, eight and sixteen beat bursts are supported and the burst may be either incrementing or wrapping.
HPROT[3:0] Protection control	Master	The protection control signals provide additional information about a bus access and are primarily intended for use by any module that wishes to implement some level of protection. The signals indicate if the transfer is an opcode fetch or data access, as well as if the transfer is a privileged mode access or user mode access. For bus masters with a memory management unit these signals also indicate whether the current access is cacheable or bufferable.
HWDATA[31:0] Write data bus	Master	The write data bus is used to transfer data from the master to the bus slaves during write operations. A minimum data bus width of 32 bits is recommended. However, this may

		easily be extended to allow for higher bandwidth operation.
HSELx Slave select	Decoder	Each AHB slave has its own slave select signal and this signal indicates that the current transfer is intended for the selected slave. This signal is simply a combinatorial decode of the address bus.
HRDATA[31:0] Read data bus	Slave	The read data bus is used to transfer data from bus slaves to the bus master during read operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HREADY Transfer done	Slave	When HIGH the **HREADY** signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer. Note: Slaves on the bus require **HREADY** as both an input and an output signal.
HRESP[1:0] Transfer response	Slave	The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY and SPLIT.