AMBA AXI Design

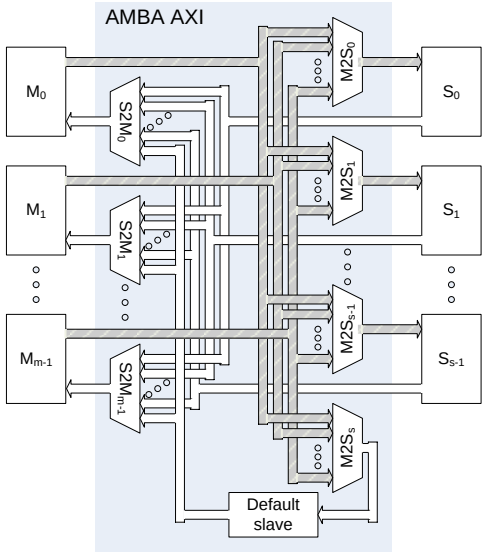
2013 – 2017

Ando Ki
(adki@future-ds.com)

Agenda

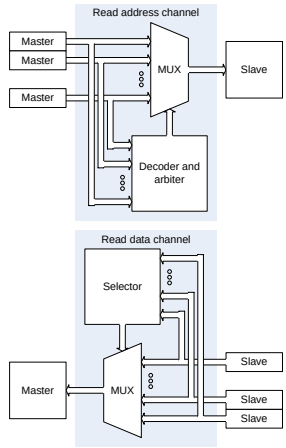
- Structure of AMBA AXI
- Read channels
- Write channels
- Forward channels
- Backward channels
- AMAB AXI two-master and two-slave case

Structure of AMBA AXI

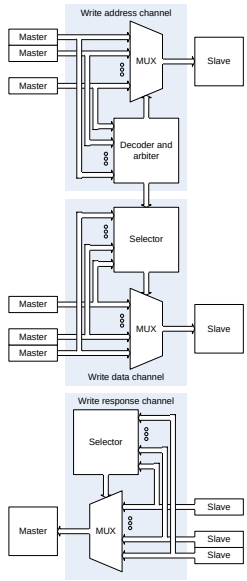


- Channel-based protocol
- 5 independent channels
- Single-clock edge operation: ACLK
- Two-way flow-control mechanism for each channel
- Valid/ready handshake mechanism
- Separate address/control and data phases
- Registers slices
- Easy to add register stages since each AXI channel transfers information in only one direction
- Burst-based protocol
- One address for burst
- Variable-length burst
- 1 to 16 data transfers per burst for AXI3; up to 256 for AXI4
- Multiple outstanding bursts
- Out-of-order transaction completion
- Data interleaving
- Lock and exclusive for atomic access in AXI3; only exclusive for atomic in AXI4

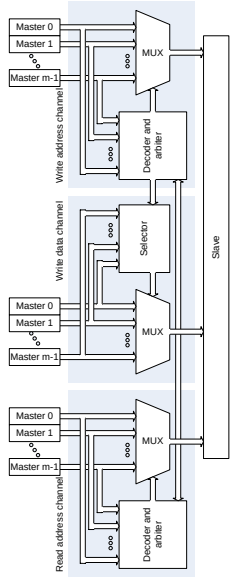
Read channels



Write channels

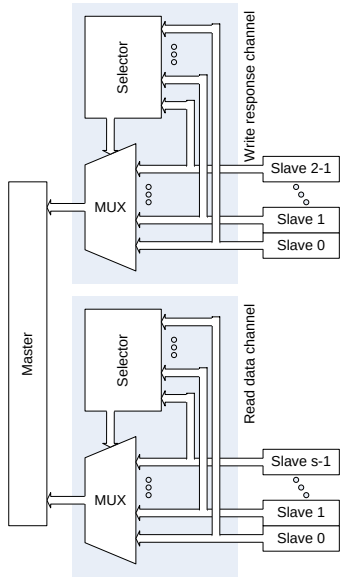


Forward channels



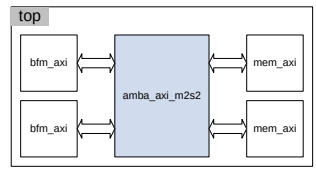
- Forward channels are dedicated to a specific slave and handles transactions driven by multiple masters. Write-address and read-address channel use address (AWADDR, ARADDR) to select a specific slave, while write-data channel uses information given by write-address channel.

Backward channels



- Backward channels are dedicated to a specific master and handles transactions driven by multiple slaves. Write-response and read-data channels use tags (BID and RID) to select the master that is waiting for the transaction.

AMBA AXI two-master and two-slave case



Simulation with ModelSim (1/4)

```
# Makefile
SHELL                   = /bin/sh
MAKEFILE        = Makefile
#----------------------------------------------------------
VLIB            = $(shell which vlib)
VLOG            = $(shell which vlog)
VSIM            = $(shell which vsim)
WORK            = work
#----------------------------------------------------------
TOP             = top
#----------------------------------------------------------
all: vlib compile simulate

vlib:
                if [ -d $(WORK) ]; then /bin/rm -rf $(WORK); fi
                $(VLIB) $(WORK)

compile:
                $(VLOG) -lint -work $(WORK)  -f modelsim.args

simulate: compile
                $(VSIM) -novopt -c -do "run -all; quit" $(WORK).$(TOP)
```

Modelsim commands

Specify where to store compile results

Compilation

Simulation

```
@ECHO OFF
REM RunMe.bat
SET MODELSIMWORK=work
SET MODELSIMVLIB=vlib
SET MODELSIMVSIM=vsim
SET MODELSIMVCOM=vcom
SET MODELSIMVLOG=vlog

SET DESIGNTOP=top

IF EXIST %MODELSIMWORK% RMDIR /S/Q %MODELSIMWORK%

%MODELSIMVLIB% %MODELSIMWORK%
%MODELSIMVLOG% -work %MODELSIMWORK% -lint^
        -f modelsim.args
%MODELSIMVSIM% -novopt -c -do "run -all; quit"^
        %MODELSIMWORK%.%DESIGNTOP%
```

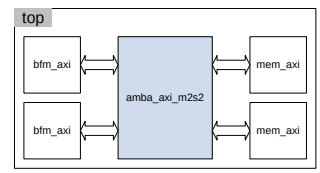

Simulation with ModelSim (2/4)

modelsim.args

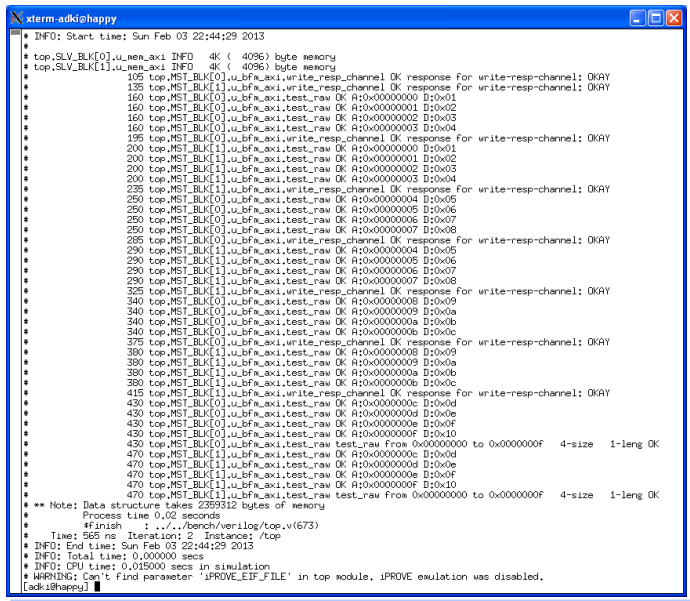
```
//------------------------------------------------------------------------
+incdir+../../design/verilog
        ./sim_define.v
        ../../design/verilog/axi_switch_m2s2.v
//------------------------------------------------------------------------
+incdir+../../../../bfm_axi_task/example/design/verilog
        ../../../../bfm_axi_task/example/design/verilog/bfm_axi.v
        ../../../../bfm_axi_task/example/design/verilog/mem_axi.v
//------------------------------------------------------------------------
// Below are test-bench
//------------------------------------------------------------------------
+incdir+../../bench/verilog
        ../../bench/verilog/top.v
//------------------------------------------------------------------------
```

sim_define.v

```
`ifndef _SIM_DEFINE_V_
`define _SIM_DEFNE_V_
`define SIM      // define this for simulation case if you are not sure
`define VCD      // define this for VCD waveform dump
`define DEBUG
`define RIGOR
`define LOW_POWER
//------------------------------------------------------------------------
`endif
```



Simulation with ModelSim (3/4)



Example: AMBA AXI case

- This example shows how to use BFM with tasks
  - Note that this design uses codes in "$(PROJECT)/codes/bfm_axi_task/design/verilog"
    - bfm_axi.v and mem_axi.v
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/amba_axi
  - Step 2: see the codes
    - [user@host] cd $(PROJECT)/codes/amba_axi/desing/verilog
    - You should fill necessary code in "$(PROJECT)/codes/amba_axi/bench/verilog/top.v"
  - Step 3: compile and run
    - [user@host] cd $(PROJECT)/codes/amba_axi/sim/modelsim
    - [user@host] make
  - Step 4: waveform view
    - [user@host] gtkwave wave.vcd &

[user@host] cd $(PROJECT)/codes/amba_axi/sim/modelsim
[user@host] make
[user@host] gtkwave wave.vcd &

Example: AMBA AXI case