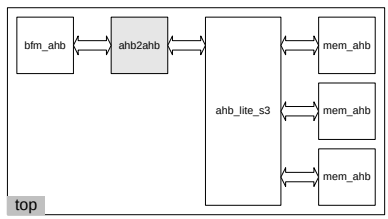
Design and Verification of AHB2AHB

2013 – 2017

Ando Ki
(adki@future-ds.com)

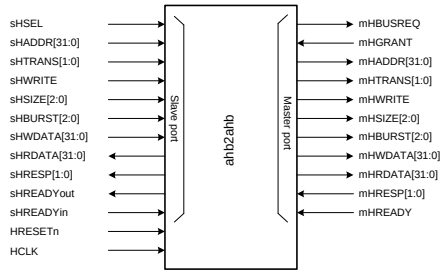
AMBA AHB-to-AHB design & verification

- Test-bench includes 'BFM', 'AMBA AHB', and 'MEMORY'.
- BFM generates test-pattern and test-vector.

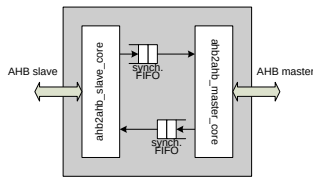


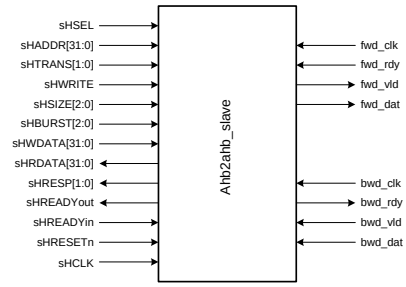
AHB-to-AHB bus bridge

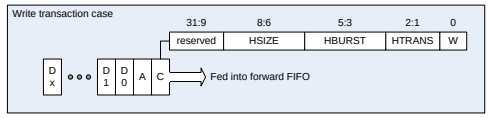
- Note that there is only one HCLK.
  - This means it is synchronous bridge.

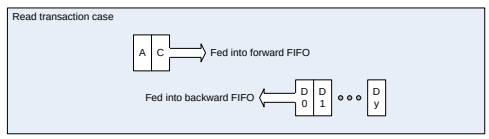


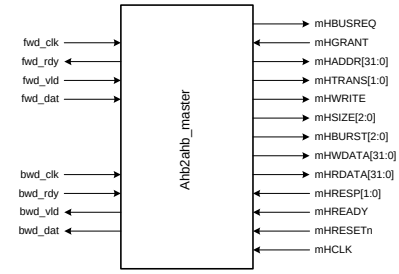
AHB-to-AHB internal











AHB2AHB: module

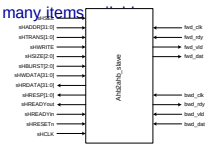
```
`timescale 1ns/1ns
`include "ahb2ahb_slave_core.v"
`include "ahb2ahb_master_core.v"
`include "ahb2ahb_fifo.v"

module ahb2ahb (
      input  wire        HRESETn
    , input  wire        HCLK
    , input  wire        sHSEL
    , input  wire [31:0] sHADDR
    , input  wire [1:0]  sHTRANS
    , input  wire        sHWRITE
    , input  wire [2:0]  sHSIZE
    , input  wire [2:0]  sHBURST
    , input  wire [31:0] sHWDATA
    , output wire [31:0] sHRDATA
    , output wire [1:0]  sHRESP
    , input  wire        sHREADYin
    , output wire        sHREADYout
    , output wire        mHBUSREQ
    , input  wire        mHGRANT
    , output wire [31:0] mHADDR
    , output wire [1:0]  mHTRANS
    , output wire        mHWRITE
    , output wire [2:0]  mHSIZE
    , output wire [2:0]  mHBURST
    , output wire [31:0] mHWDATA
    , input  wire [31:0] mHRDATA
    , input  wire [1:0]  mHRESP
    , input  wire        mHREADY
);
```

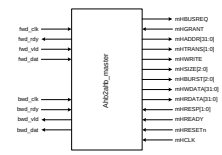
```
//-----------------------------------------------------
  localparam FIFO_AW=5;
  //-----------------------------------------------------
  wire            fwd_wr_rdy;
  wire            fwd_wr_vld;
  wire [31:0]     fwd_wr_dat;
  wire            fwd_rd_rdy;
  wire            fwd_rd_vld;
  wire [31:0]     fwd_rd_dat;
  wire            fwd_full;
  wire            fwd_empty;
  wire [FIFO_AW:0] fwd_cnt_rd;
  wire [FIFO_AW:0] fwd_cnt_wr;
  wire            bwd_rd_rdy;
  wire            bwd_rd_vld;
  wire [31:0]     bwd_rd_dat;
  wire            bwd_wr_rdy;
  wire            bwd_wr_vld;
  wire [31:0]     bwd_wr_dat;
  wire            bwd_full;
  wire            bwd_empty;
  wire [FIFO_AW:0] bwd_cnt_rd;
  wire [FIFO_AW:0] bwd_cnt_wr;
```


AHB2AHB: module

```
//-----------------------------------------------------
  // It handles AHB transaction as a slave.
  // It pushes request information to the forward FIFO.
  // It pops response information from the backwoard FIFO.
  ahb2ahb_slave_core Uslave (
        .HRESETn   (HRESETn   )
      , .HCLK      (HCLK      )
      , .HSEL      (sHSEL     )
      , .HADDR     (sHADDR    )
      , .HTRANS    (sHTRANS   )
      , .HWRITE    (sHWRITE   )
      , .HSIZE     (sHSIZE    )
      , .HBURST    (sHBURST   )
      , .HWDATA    (sHWDATA   )
      , .HRDATA    (sHRDATA   )
      , .HRESP     (sHRESP    )
      , .HREADYin  (sHREADYin )
      , .HREADYout (sHREADYout)
      , .fwr_clk   (           ) // output: should be HCLK
      , .fwr_rdy   (fwd_wr_rdy)
      , .fwr_vld   (fwd_wr_vld)
      , .fwr_dat   (fwd_wr_dat)
      , .fwr_full  (fwd_full  )
      , .fwr_cnt   (fwd_cnt_wr) // how many rooms available
      , .brd_clk   (           ) // output: should be HCLK
      , .brd_rdy   (bwd_rd_rdy)
      , .brd_vld   (bwd_rd_vld)
      , .brd_dat   (bwd_rd_dat)
      , .brd_empty (bwd_empty )
      , .brd_cnt   (bwd_cnt_rd) // how many items
  );
```




```
//-----------------------------------------------------
  ahb2ahb_master_core Umaster (
        .HRESETn   (HRESETn )
      , .HCLK      (HCLK    )
      , .HBUSREQ   (mHBUSREQ)
      , .HGRANT    (mHGRANT )
      , .HADDR     (mHADDR  )
      , .HTRANS    (mHTRANS )
      , .HWRITE    (mHWRITE )
      , .HSIZE     (mHSIZE  )
      , .HBURST    (mHBURST )
      , .HWDATA    (mHWDATA )
      , .HRDATA    (mHRDATA )
      , .HRESP     (mHRESP  )
      , .HREADY    (mHREADY )
      , .frd_clk   (        ) // output: should be HCLK
      , .frd_rdy   (fwd_rd_rdy)
      , .frd_vld   (fwd_rd_vld)
      , .frd_dat   (fwd_rd_dat)
      , .frd_empty (fwd_empty )
      , .frd_cnt   (fwd_cnt_rd) // num ele. to read
      , .bwr_clk   (        ) // output: should be HCLK
      , .bwr_rdy   (bwd_wr_rdy)
      , .bwr_vld   (bwd_wr_vld)
      , .bwr_dat   (bwd_wr_dat)
      , .bwr_full  (bwd_full  )
      , .bwr_cnt   (bwd_cnt_wr) // num rooms to write
  );
```

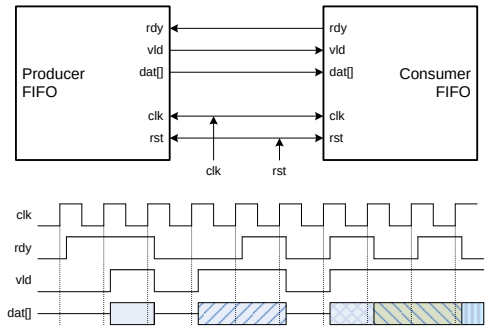


AHB2AHB: module

```
//-----------------------------------------------------
  // from slave-to-master
  // all address related information
  ahb2ahb_fifo #(32, FIFO_AW) Ufwd_fifo (
       .rst     (~HRESETn   )
     , .clr     (1'b0       )
     , .clk     (HCLK       ) // it should be HCLK
     , .wr_rdy  (fwd_wr_rdy )
     , .wr_vld  (fwd_wr_vld )
     , .wr_din  (fwd_wr_dat )
     , .rd_rdy  (fwd_rd_rdy )
     , .rd_vld  (fwd_rd_vld )
     , .rd_dout (fwd_rd_dat )
     , .empty   (fwd_empty  )
     , .full    (fwd_full   )
     , .fullN   ()
     , .emptyN  ()
     , .rd_cnt  (fwd_cnt_rd ) // how many items
     , .wr_cnt  (fwd_cnt_wr ) // how many rooms
   );
```

```
//-----------------------------------------------------
  // from master-to-slave
  // all data related information
  ahb2ahb_fifo #(32, FIFO_AW) Ubwd_fifo (
       .rst     (~HRESETn  )
     , .clr     (1'b0      )
     , .clk     (HCLK      ) // it should be HCLK
     , .wr_rdy  (bwd_wr_rdy)
     , .wr_vld  (bwd_wr_vld)
     , .wr_din  (bwd_wr_dat)
     , .rd_rdy  (bwd_rd_rdy)
     , .rd_vld  (bwd_rd_vld)
     , .rd_dout (bwd_rd_dat)
     , .empty   (bwd_empty )
     , .full    (bwd_full  )
     , .fullN   ()
     , .emptyN  ()
     , .rd_cnt  (bwd_cnt_rd)
     , .wr_cnt  (bwd_cnt_wr)
   );
```


Ready-valid handshake based FIFO



- As a short of dual-ready handshake protocol, ready-valid handshake protocol uses two signals in order to control stream style data movement between two blocks, where one is producer and the other is consumer in terms of data.
- Handshake signals:
  - Ready (rdy): the consumer is ready to accept data
  - Valid (vld):the data is now valid
- The data moves from producer to consumer whenever both ‘vld’ and ‘rdy’ are high at the rising edge of clk.

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
+incdir+../../../../bfm_ahb_task/example/design/verilog
        ./sim_define.v
        ../../design/verilog/ahb2ahb.v
        ../../../../bfm_ahb_task/example/design/verilog/bfm_ahb.v
        ../../../../bfm_ahb_task/example/design/verilog/mem_ahb.v
+incdir+../../../../ahb_lite/example/design/verilog
        ../../../../ahb_lite/example/design/verilog/ahb_lite_s3.v
//----------------------------------------------------------------
// Below are test-bench
//----------------------------------------------------------------
+incdir+../../bench/verilog
        ../../bench/verilog/top.v
//----------------------------------------------------------------
```

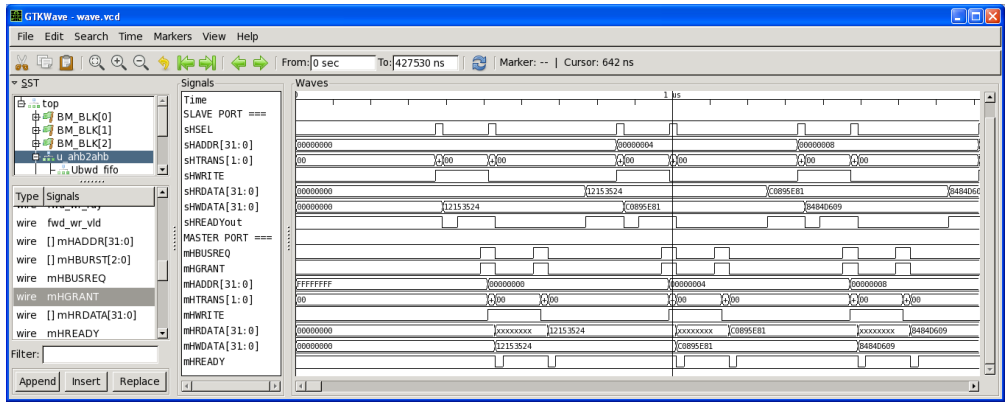
sim_define.v

```
`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
`undef  DEBUG
`define RIGOR
`define LOW_POWER
//----------------------------------------------------------------
`define CLK_FREQ   50000000
`define MEM_DELAY  1
//----------------------------------------------------------------
`define SINGLE_TEST
`define BURST_TEST
//----------------------------------------------------------------
`endif
```

Simulation with ModelSim (3/4)

This design has problem with burst access.

Simulation with ModelSim (4/4)



Example: AHB to AHB case

- This example shows how to use BFM with tasks
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/ahb_to_ahb
  - Step 2: see the codes
    - [user@host] cd $(PROJECT)/codes/ahb_to_ahb/desing/verilog
  - Step 3: compile and run
    - [user@host] cd $(PROJECT)/codes/ahb_to_ahb/sim/modelsim
    - [user@host] make
  - Step 4: waveform view
    - [user@host] gtkwave wave.vcd &

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


Issues and quiz

- Deadlock case with dual-AHB2AHB

Dead-lock example