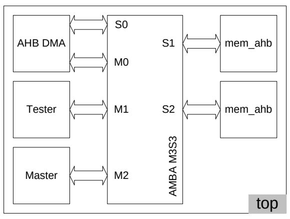
Design and Verification of AHB DMA

2015 – 2017

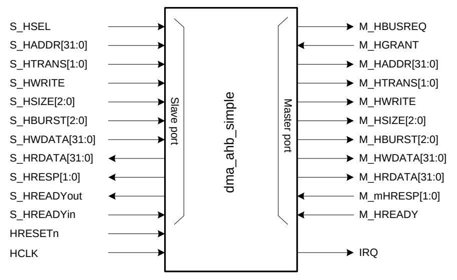
Ando Ki, Ph.D.
(adki@future-ds.com)

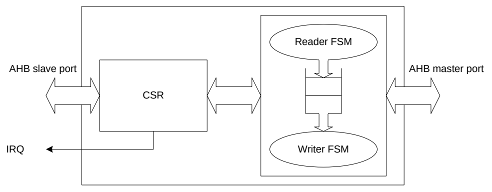
AHB DMA design & verification



- Test-bench includes
  - DMA
  - Tester: Controlling DMA
  - Master: memory testing
    - It tests memory by writing & reading while DMA operates.
  - Memory

AHB DMA





- There are some highlights as follows.
  - AMBA AHB 2.0 is supported
  - Interrupt signal when data movement completes
  - Single or burst (4, 8, 16) are supported
  - Misaligned accesses are supported, i.e., starting address can be any, for example 0, 1, 2, or 3 of modulus 4
  - Up to 216-1 bytes can be moved
- There are some limitations as follows.
  - AHB split transaction is not supported
  - AHB early termination is not supported
  - Starting addresses of source and destination should be the same in terms of modulus (4 x burst length)

DMA CSR (1/2)

Name	Address offset			description
			Bit#	
NAME0	+000h		RO	DMA
NAME1	+004h		RO	AHB
NAME2	+008h		RO	
NAME3	+00Ch		RO	
COMP0	+010h		RO	DYNA
COMP1	+014h		RO	LITH
COMP2	+018h		RO	
COMP3	+01Ch		RO	
VERSION	+020h		RO	Version (0x2014_0429)
RESERVED	+024h			Reserved
	+028h			Reserved
	+02Ch			Reserved
CONTROL	+030h		RW	CONTROL register (default: 0x0000_0000)
			31	EN: enable
			30:2	Reserved
			1	IP: 1 when interrupt is pending
			0	IE: interrupt is enabled when 1

DMA CSR (2/2)

Name	Address offset			description
			Bit#	
NUM	+040h			NUM register (default: 0x0001_0000)
			31	GO: start DMA when 1 and return 0 when completed
			30	BUSY (read-only)
			29	DONE (read-only)
			28:21	Reserved
			20:16	BURST: burst length (1, 4, 8, 16)
			15:0	BYTES : num of bytes to move
SOURCE	+044h			SRC register (default: 0x0000_0000)
			31:0	source address
DESTINATION	+048h			DST register (default: 0x0000_0000)
			31:0	destination address

dma_ahb_simple.v

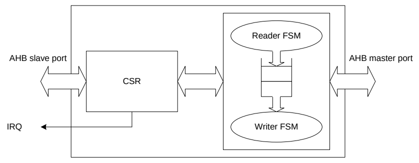
```
`include "dma_ahb_csr_ahb.v"
`include "dma_ahb_master.v"
`timescale 1ns/1ns

module dma_ahb_simple
(
      input   wire         HRESETn
    , input   wire         HCLK
    //--------------------------------------------------
    , input   wire         S_HSEL
    , input   wire  [31:0]  S_HADDR
    , input   wire  [ 1:0]  S_HTRANS
    , input   wire         S_HWRITE
    , input   wire  [ 2:0]  S_HSIZE
    , input   wire  [ 2:0]  S_HBURST
    , input   wire  [31:0]  S_HWDATA
    , output  wire  [31:0]  S_HRDATA
    , output  wire  [ 1:0]  S_HRESP
    , input   wire         S_HREADYin
    , output  wire         S_HREADYout
    //--------------------------------------------------
    , output  wire         IRQ
    //--------------------------------------------------
    // bus interface
    , output  wire         M_HBUSREQ
    , input   wire         M_HGRANT
    , output  wire  [31:0]  M_HADDR
    , output  wire  [ 1:0]  M_HTRANS
    , output  wire         M_HWRITE
    , output  wire  [ 2:0]  M_HSIZE
    , output  wire  [ 2:0]  M_HBURST
    , output  wire  [ 3:0]  M_HPROT
    , output  wire  [31:0]  M_HWDATA
    , input   wire  [31:0]  M_HRDATA
    , input   wire  [ 1:0]  M_HRESP
    , input   wire         M_HREADY
    //--------------------------------------------------
);
```

```
//--------------------------------------------------
 wire           DMA_EN   ; // synchronous reset
 wire           DMA_GO   ;
 wire           DMA_BUSY ;
 wire           DMA_DONE ;
 wire  [31:0]    DMA_SRC  ; // source address
 wire  [31:0]    DMA_DST  ; // destination address
 wire  [15:0]    DMA_BNUM ; // num of bytes to move
 wire  [ 4:0]   DMA_BURST; // burst length 1, 4, 8 , 16
 //--------------------------------------------------
 dma_ahb_csr_ahb
 u_csr (
     .HRESETn   (HRESETn    )
   , .HCLK      (HCLK       )
   , .HSEL      (S_HSEL     )
   , .HADDR     (S_HADDR    )
   , .HTRANS    (S_HTRANS   )
   , .HWRITE    (S_HWRITE   )
   , .HSIZE     (S_HSIZE    )
   , .HBURST    (S_HBURST   )
   , .HWDATA    (S_HWDATA   )
   , .HRDATA    (S_HRDATA   )
   , .HRESP     (S_HRESP    )
   , .HREADYin  (S_HREADYin )
   , .HREADYout (S_HREADYout)
   , .IRQ       (IRQ        )
   , .DMA_EN    (DMA_EN     )
   , .DMA_GO    (DMA_GO     )
   , .DMA_BUSY  (DMA_BUSY   )
   , .DMA_DONE  (DMA_DONE   )
   , .DMA_SRC   (DMA_SRC    )
   , .DMA_DST   (DMA_DST    )
   , .DMA_BNUM  (DMA_BNUM   )
   , .DMA_BURST (DMA_BURST  )
 );
```

dma_ahb_simple.v

```
  //------------------------------------------------------
   dma_ahb_master
   u_master (
       .HRESETn    (HRESETn     )
     , .HCLK       (HCLK        )
     , .HBUSREQ    (M_HBUSREQ   )
     , .HGRANT     (M_HGRANT    )
     , .HADDR      (M_HADDR     )
     , .HTRANS     (M_HTRANS    )
     , .HWRITE     (M_HWRITE    )
     , .HSIZE      (M_HSIZE     )
     , .HBURST     (M_HBURST    )
     , .HPROT      (M_HPROT     )
     , .HWDATA     (M_HWDATA    )
     , .HRDATA     (M_HRDATA    )
     , .HRESP      (M_HRESP     )
     , .HREADY     (M_HREADY    )
     , .DMA_EN     (DMA_EN      )
     , .DMA_GO     (DMA_GO      )
     , .DMA_BUSY   (DMA_BUSY    )
     , .DMA_DONE   (DMA_DONE    )
     , .DMA_SRC    (DMA_SRC     )
     , .DMA_DST    (DMA_DST     )
     , .DMA_BNUM   (DMA_BNUM    )
     , .DMA_BURST  (DMA_BURST   )
   );
   //------------------------------------------------------
endmodule
```



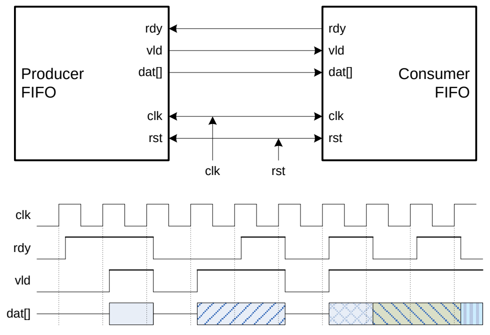
dma_ahb_master.v

```
`include "dma_ahb_fifo_sync_small.v"
`timescale 1ns/1ns

module dma_ahb_master
(
       input  wire          HRESETn
     , input  wire          HCLK
     , output reg           HBUSREQ=1'b0
     , input  wire          HGRANT
     , output reg  [31:0]   HADDR=~32'h0
     , output reg  [ 1:0]   HTRANS=2'b0
     , output reg           HWRITE=1'b0
     , output reg  [ 2:0]   HSIZE=3'b0
     , output reg  [ 2:0]   HBURST=3'b0
     , output wire [ 3:0]   HPROT
     , output reg  [31:0]   HWDATA=~32'h0
     , input  wire [31:0]   HRDATA
     , input  wire [ 1:0]   HRESP
     , input  wire          HREADY
     //------------------------------------------------------
     , input  wire          DMA_EN
     , input  wire          DMA_GO
     , output reg           DMA_BUSY=1'b0
     , output reg           DMA_DONE=1'b0
     , input  wire [31:0]   DMA_SRC
     , input  wire [31:0]   DMA_DST
     , input  wire [15:0]   DMA_BNUM // num of bytes to move
     , input  wire [ 4:0]   DMA_BURST // burst length 1, 4, 8 , 16
);
 //------------------------------------------------------
   assign HPROT  = 4'b0011;
```

```
//------------------------------------------------------
 reg [31:0] dma_rd_addr  = ~32'h0;
 reg [15:0] dma_rd_bnum  =  16'h0;
 reg [ 4:0] dma_rd_burst =   5'h0;
 reg [31:0] dma_rd_data  = ~32'h0;
 reg [31:0] dma_wr_addr  = ~32'h0;
 reg [15:0] dma_wr_bnum  =  16'h0;
 reg [ 4:0] dma_wr_burst =   5'h0;
 reg [31:0] dma_wr_data  = ~32'h0;
 reg        dma_wr_data_vld=1'b0;
 reg [ 6:0] bnum  =  6'h0; // 1~60
 //------------------------------------------------------
 reg        fifo_wr_vld=1'b0;
 wire       fifo_wr_rdy;
 reg  [31:0] fifo_wr_dat=32'h0;
 wire       fifo_rd_vld;
 reg        fifo_rd_rdy=1'b0;
 wire [31:0] fifo_rd_dat;
 wire       fifo_empty ;
 wire       fifo_full  ;
 //------------------------------------------------------
 localparam ST_READY    = 'h0
          , ST_START    = 'h1
          , ST_RD_ARB_S = 'h2
          , ST_RD_S0    = 'h3
          , ST_RD_S1    = 'h4
          , ST_WR_ARB_S = 'h5
          , ST_WR_S0    = 'h6
          , ST_WR_S1    = 'h7
          , ST_RD_ARB_B = 'h8
          , ST_RD_B0    = 'h9
          , ST_RD_B1    = 'hA
          , ST_RD_B2    = 'hB
          , ST_WR_ARB_B = 'hC
          , ST_WR_B0    = 'hD
          , ST_CHECK    = 'hE
          , ST_DONE     = 'hF;
 reg [3:0] state = ST_READY;
 //------------------------------------------------------
```

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)

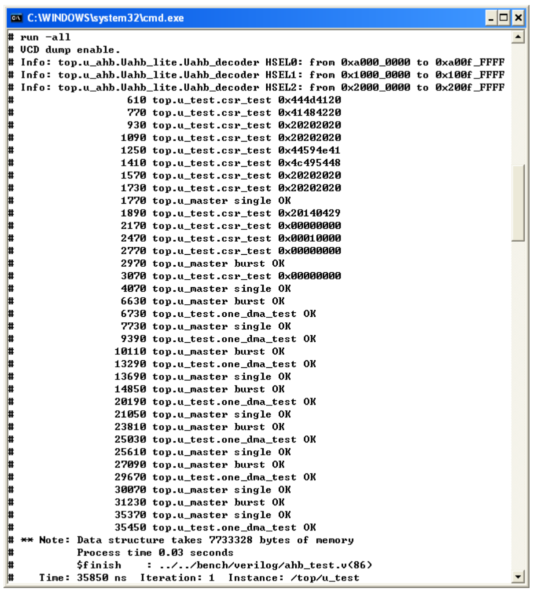
modelsim.args

```
//-----------------------------------------------
+define+VCD
+define+SIM
//-----------------------------------------------
./sim_define.v
//-----------------------------------------------
+incdir+../../rtl/verilog
       ../../rtl/verilog/dma_ahb_simple.v
//-----------------------------------------------
+incdir+../../bench/verilog
       ../../bench/verilog/top.v
       ../../bench/verilog/mem_ahb.v
       ../../bench/verilog/ahb_test.v
       ../../bench/verilog/amba_ahb_m3s3.v
       ../../bench/verilog/ahb_lite_s3.v
       ../../bench/verilog/ahb_master.v
//-----------------------------------------------
```

sim_define.v

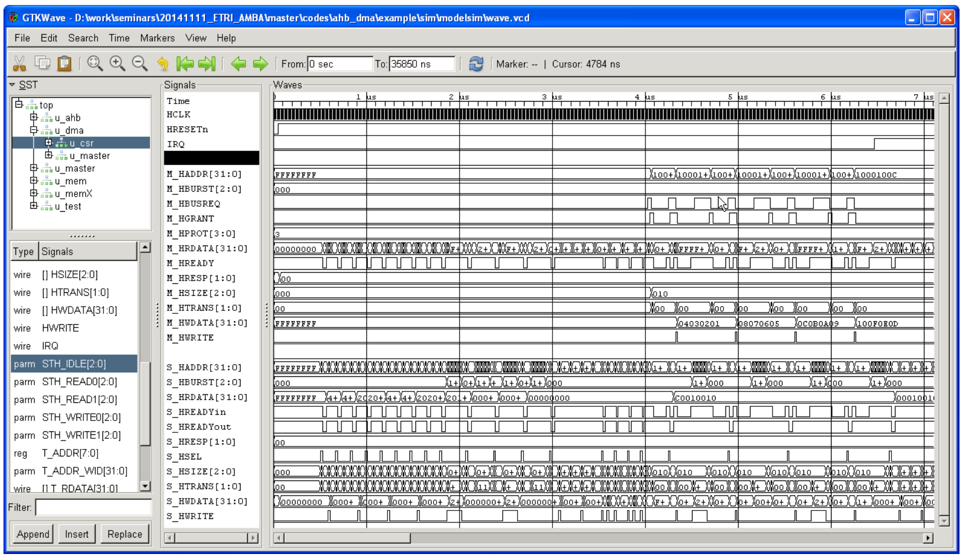
```
`ifndef _SIM_DEFINE_V_
`define _SIM_DEFNE_V_
//-----------------------------------------------------
// Copyright (c) 2013 by Dynalith Systems Co., Ltd.
// All rights reserved.
//-----------------------------------------------------
`define SIM
`undef  SYN
//-----------------------------------------------------
`ifdef SIM
`define RIGOR
`define VCD
`endif
//-----------------------------------------------------
`endif
```

Simulation with ModelSim (3/4)



```
# run -all
# VCD dump enable.
# Info: top.u_ahb.Uahb_lite.Uahb_decoder HSEL0: from 0xa000_0000 to 0xa00f_FFFF
# Info: top.u_ahb.Uahb_lite.Uahb_decoder HSEL1: from 0x1000_0000 to 0x100f_FFFF
# Info: top.u_ahb.Uahb_lite.Uahb_decoder HSEL2: from 0x2000_0000 to 0x200f_FFFF
#                  610 top.u_test.csr_test 0x444d4120
#                  770 top.u_test.csr_test 0x41484220
#                  930 top.u_test.csr_test 0x20202020
#                 1090 top.u_test.csr_test 0x20202020
#                 1250 top.u_test.csr_test 0x44594e41
#                 1410 top.u_test.csr_test 0x4c495448
#                 1570 top.u_test.csr_test 0x20202020
#                 1730 top.u_test.csr_test 0x20202020
#                 1770 top.u_master single OK
#                 1890 top.u_test.csr_test 0x20140429
#                 2170 top.u_test.csr_test 0x00000000
#                 2470 top.u_test.csr_test 0x00010000
#                 2770 top.u_test.csr_test 0x00000000
#                 2970 top.u_master burst OK
#                 3070 top.u_test.csr_test 0x00000000
#                 4070 top.u_master single OK
#                 6630 top.u_master burst OK
#                 6730 top.u_test.one_dma_test OK
#                 7730 top.u_master single OK
#                 9390 top.u_test.one_dma_test OK
#                10110 top.u_master burst OK
#                13290 top.u_test.one_dma_test OK
#                13690 top.u_master single OK
#                14850 top.u_master burst OK
#                20190 top.u_test.one_dma_test OK
#                21050 top.u_master single OK
#                23810 top.u_master burst OK
#                25030 top.u_test.one_dma_test OK
#                25610 top.u_master single OK
#                27090 top.u_master burst OK
#                29670 top.u_test.one_dma_test OK
#                30070 top.u_master single OK
#                31230 top.u_master burst OK
#                35370 top.u_master single OK
#                35450 top.u_test.one_dma_test OK
# ** Note: Data structure takes 7733328 bytes of memory
#          Process time 0.03 seconds
#          $finish    : ../../bench/verilog/ahb_test.v(86)
#    Time: 35850 ns  Iteration: 1  Instance: /top/u_test
```

Simulation with ModelSim (4/4)



Example: AHB DMA

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

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