

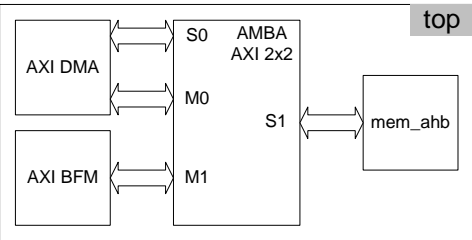
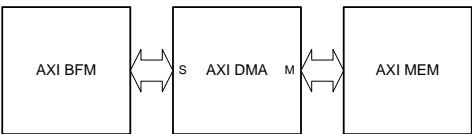
# Design and Verification of AXI DMA

2015 – 2020

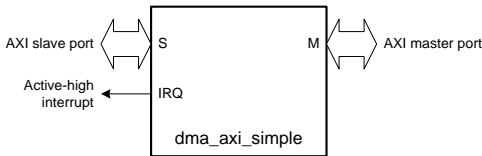
Ando Ki, Ph.D.  
(adki@future-ds.com)

## AXI DMA design & verification

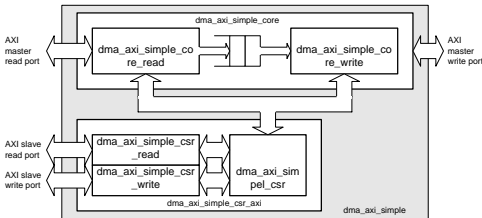
- Test-bench includes
- ◆ DMA
  - ◆ BFM: Controlling DMA
  - ◆ Memory



# AXI DMA



- There are some highlights as follows.
  - AMBA AXI 3 and AXI 4 are supported
  - Interrupt signal when data movement completes
  - Up to  $2^{16}-1$  bytes can be moved
- There are some limitations as follows.
  - Starting addresses of source and destination should be the same in terms of modulus (4 x burst length)



## DMA CSR (1/3)

Name	Address offset	description	
		Bit#	
NAME0	+000h	RO	DMA
NAME1	+004h	RO	AXI
NAME2	+008h	RO	
NAME3	+00Ch	RO	
COMP0	+010h	RO	DYNA
COMP1	+014h	RO	LITH
COMP2	+018h	RO	
COMP3	+01Ch	RO	
VERSION	+020h	RO	Version (0x2015_0712)
RESERVED	+024h		Reserved
	+028h		Reserved
	+02Ch		Reserved

## DMA CSR (2/3)

Name	Address offset	Bit#	description
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
	+038h		Reserved
		31:0	
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)
		28:24	Reserved
		23:16	CHUNK: num of bytes for a chunk - It should be a multiple of data-bus width. - Data-bus width is used when it is 0.
		15:0	BYTES : num of bytes to move

Copyright © 2015-2017-2018 by Ando Ki

AXI DMA ( 5 )

## DMA CSR (3/3)

Name	Address offset	Bit#	description
SOURCE	+044h		SRC register (default: 0x0000_0000)
		31:0	source address
DESTINATION	+048h		DST register (default: 0x0000_0000)
		31:0	destination address

Copyright © 2015-2017-2018 by Ando Ki

AXI DMA ( 6 )

## dma\_axi\_simple.v

```

`include "dma_axi_simple_defines.v"
`include "dma_axi_simple_csr_axi.v"
`include "dma_axi_simple_core.v"
`timescale 1ns/1ns

module dma_axi_simple
#(parameter AXI_MST_ID =1
  ,AXI_WIDTH_CID=4
  ,AXI_WIDTH_ID=4
  ,AXI_WIDTH_AD=32
  ,AXI_WIDTH_DA=32
  ,AXI_WIDTH_DS=(AXI_WIDTH_DA/8)
  ,AXI_WIDTH_DSB=clogb2(AXI_WIDTH_DS)
  ,AXI_WIDTH_SID=AXI_WIDTH_CID+AXI_WIDTH_ID
)
(
  input wire      ARESETn
  ,input wire      ACLK
  // DMA AXI Master Port
  ,AMBA_AXI_MASTER_PORT(wire)
  // AXI Slave Port for CSR
  ,AMBA_AXI_SLAVE_PORT(wire)
  //
  ,input wire      CSYSREQ
  ,output wire     CSYSACK
  ,output wire     CACTIVE
  //
  ,output wire     IRQ
);

//-----
assign CSYSACK = CSYSREQ;
assign CACTIVE = 1'b1;
//-----
wire      DMA_EN ;
wire      DMA_GO ;
wire      DMA_BUSY ;
wire      DMA_DONE ;
wire [31:0] DMA_SRC ;
wire [31:0] DMA_DST ;
wire [15:0] DMA_BNUM ;// num of bytes to move
wire [ 7:0] DMA_CHUNK ;// AxLEN (+1 beats)
//-----
dma_axi_simple_csr_axi
#(AXI_WIDTH_CID(AXI_WIDTH_CID)
  ,AXI_WIDTH_ID(AXI_WIDTH_ID)
  ,AXI_WIDTH_SID(AXI_WIDTH_SID)
  ,AXI_WIDTH_AD(AXI_WIDTH_AD)
  ,AXI_WIDTH_DA(AXI_WIDTH_DA))
u_csr (
  .ARESETn (ARESETn )
  ,.ACLK    (ACLK )
  ,.AMBA_AXI_SLAVE_PORT_CONNECTION
  ,.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_CHUNK(DMA_CHUNK )
);

```

Copyright © 2015-2017-2018 by Ando Ki

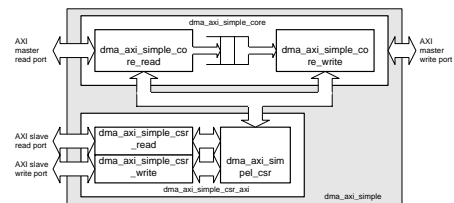
AXI DMA ( 7 )

## dma\_axi\_simple.v

```

//-----
dma_axi_simple_core #(AXI_MST_ID (AXI_MST_ID )
  ,AXI_WIDTH_CID(AXI_WIDTH_CID)
  ,AXI_WIDTH_ID(AXI_WIDTH_ID)
  ,AXI_WIDTH_AD(AXI_WIDTH_AD)
  ,AXI_WIDTH_DA(AXI_WIDTH_DA))
u_core (
  .ARESETn (ARESETn )
  ,.ACLK    (ACLK )
  ,.AMBA_AXI_MASTER_PORT_CONNECTION
  ,.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_CHUNK(DMA_CHUNK )
);
//-----
function integer clogb2;
input [31:0] value;
reg [31:0] tmp;
begin
  tmp = value - 1;
  for (clogb2 = 0; tmp > 0; clogb2 = clogb2 + 1) tmp = tmp >> 1;
end
endfunction
//-----
endmodule

```



Copyright © 2015-2017-2018 by Ando Ki

AXI DMA ( 8 )

## Macros to handle complex ports

<pre>// see dma_axi_simple_defines.v `define AMBA_AXI_MASTER_PORT_AW(Otype)\     output Otype [AXI_WIDTH_ID-1:0] M_AWID\     output Otype [AXI_WIDTH_AD-1:0] M_AWADDR\     `ifdef AMBA_AXI4\         output Otype [7:0] M_AWLEN\         output Otype M_AWLOCK\     `else\         .....         output Otype M_AWVALID\         input wire M_AWREADY\     `endif AMBA_AXI4\     output Otype [3:0] M_AWQOS\     output Otype [3:0] M_AWREGION\     `endif `define AMBA_AXI_MASTER_PORT_W(Otype)\     ..... `define AMBA_AXI_MASTER_PORT_B(Otype)\     ..... `define AMBA_AXI_MASTER_PORT_AR(Otype)\     ..... `define AMBA_AXI_MASTER_PORT_R(Otype)\     ..... `define AMBA_AXI_MASTER_PORT(Otype)\     output Otype [AXI_WIDTH_CID-1:0] M_MID\     `AMBA_AXI_MASTER_PORT_AW(Otype)\     `AMBA_AXI_MASTER_PORT_W(Otype)\     `AMBA_AXI_MASTER_PORT_B(Otype)\     `AMBA_AXI_MASTER_PORT_AR(Otype)\     `AMBA_AXI_MASTER_PORT_R(Otype)</pre>	<pre>`define AMBA_AXI_SLAVE_PORT_AW(Otype)\     input wire [AXI_WIDTH_SID-1:0] S_AWID\     input wire [AXI_WIDTH_AD-1:0] S_AWADDR\     `ifdef AMBA_AXI4\         input wire [7:0] S_AWLEN\         input wire S_AWLOCK\     `else\         .....         input wire S_AWVALID\         output Otype S_AWREADY\     `endif AMBA_AXI4\     input wire [3:0] S_AWQOS\     input wire [3:0] S_AWREGION\     `endif `define AMBA_AXI_SLAVE_PORT_W(Otype)\     ..... `define AMBA_AXI_SLAVE_PORT_B(Otype)\     ..... `define AMBA_AXI_SLAVE_PORT_AR(Otype)\     ..... `define AMBA_AXI_SLAVE_PORT_R(Otype)\     ..... `define AMBA_AXI_SLAVE_PORT(Otype)\     `AMBA_AXI_SLAVE_PORT_AW(Otype)\     `AMBA_AXI_SLAVE_PORT_W(Otype)\     `AMBA_AXI_SLAVE_PORT_B(Otype)\     `AMBA_AXI_SLAVE_PORT_AR(Otype)\     `AMBA_AXI_SLAVE_PORT_R(Otype)</pre>
---	---

Copyright © 2015-2017-2018 by Ando Ki

AXI DMA ( 9 )

## Simulation with ModelSim (1/4)

<pre># 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)</pre>	<p>Modelsim commands</p> <p>Specify where to store compile results</p> <p>Compilation</p> <p>Simulation</p> <pre>@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%</pre>
--	---

Copyright © 2015-2017-2018 by Ando Ki

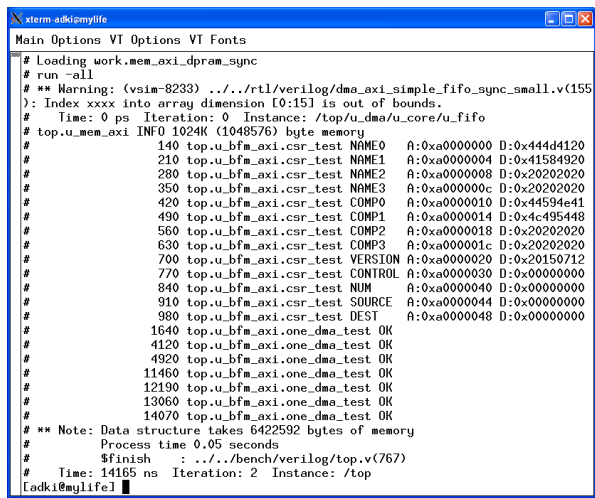
AXI DMA ( 10 )

## Simulation with ModelSim (2/4)

```
//-----
+define+VCD
+define+SIM
//-----
/sim_define.v
//-----
+incdir+../rtl/verilog
../rtl/verilog/dma_axi_simple.v
//-----
+incdir+../bench/verilog
../bench/verilog/top.v
../bench/verilog/bfm_axi.v
../bench/verilog/mem_axi.v
//-----
+incdir+../amba_axi/design/verilog
../amba_axi/design/verilog/axi_switch_m2s2.v

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

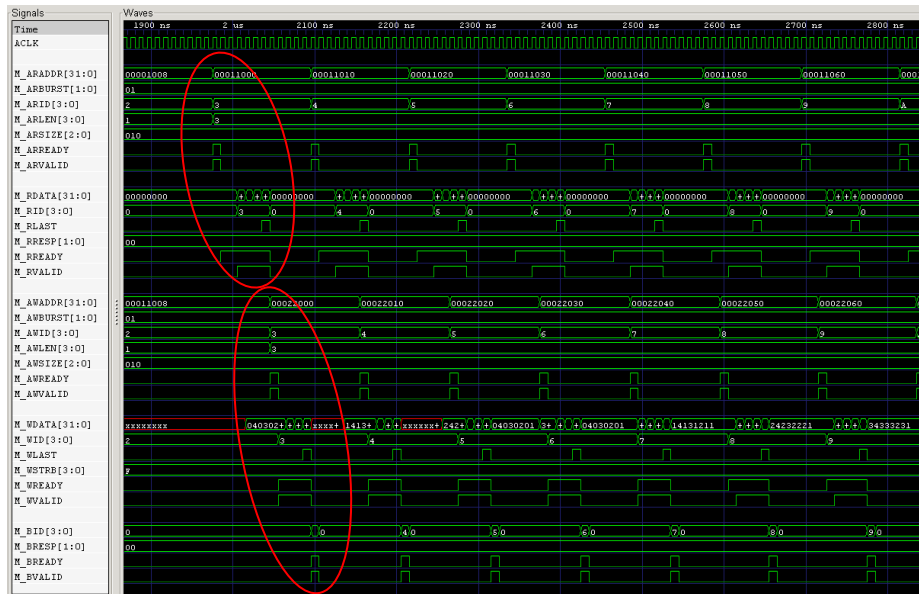
## Simulation with ModelSim (3/4)



The screenshot shows a terminal window titled "xterm: andki@mylife" with a menu bar "Main Options VI Options VI Fonts". The terminal output displays the following information:

```
# Loading work.mem_axi_dpram_sync
# run -all
# ** Warning: (vsim-8233) ../rtl/verilog/dma_axi_simple_fifo_sync_small.v(155
): Index xxx into array dimension [0:15] is out of bounds.
# Time: 0 ps Iteration: 0 Instance: /top/u_dma/u_core/u_fifo
# top.u_mem_axi INFO 1024K (1048576) byte memory
#
# 140 top.u_bfm_axi.csr_test NAME0 A:0xa0000000 D:0x444d4120
# 210 top.u_bfm_axi.csr_test NAME1 A:0xa0000004 D:0x41584920
# 280 top.u_bfm_axi.csr_test NAME2 A:0xa0000008 D:0x20202020
# 350 top.u_bfm_axi.csr_test NAME3 A:0xa000000c D:0x20202020
# 420 top.u_bfm_axi.csr_test COMP0 A:0xa0000010 D:0x44594e41
# 490 top.u_bfm_axi.csr_test COMP1 A:0xa0000014 D:0x4c495448
# 560 top.u_bfm_axi.csr_test COMP2 A:0xa0000018 D:0x20202020
# 630 top.u_bfm_axi.csr_test COMP3 A:0xa000001c D:0x20202020
# 700 top.u_bfm_axi.csr_test VERSION A:0xa0000020 D:0x20150712
# 770 top.u_bfm_axi.csr_test CONTROL A:0xa0000030 D:0x00000000
# 840 top.u_bfm_axi.csr_test NUM A:0xa0000040 D:0x00000000
# 910 top.u_bfm_axi.csr_test SOURCE A:0xa0000044 D:0x00000000
# 980 top.u_bfm_axi.csr_test DEST A:0xa0000048 D:0x00000000
#
# 1640 top.u_bfm_axi.one_dma_test OK
# 4120 top.u_bfm_axi.one_dma_test OK
# 4920 top.u_bfm_axi.one_dma_test OK
# 11460 top.u_bfm_axi.one_dma_test OK
# 12190 top.u_bfm_axi.one_dma_test OK
# 13060 top.u_bfm_axi.one_dma_test OK
# 14070 top.u_bfm_axi.one_dma_test OK
# ** Note: Data structure takes 6422592 bytes of memory
# Process time 0.05 seconds
# $finish : ../bench/verilog/top.v(767)
# Time: 14165 ns Iteration: 2 Instance: /top
[adki@mylife]
```

## Waveform



Copyright © 2015-2017-2018 by Ando Ki

AXI DMA ( 13 )

## Example: AXI DMA

❏ This example shows how to use BFM with tasks

- ◆ Step 1: go to your project directory
  - ❏ [user@host] cd \$(PROJECT)/codes/dma\_axi\_simple
- ◆ Step 2: see the codes
  - ❏ [user@host] cd \$(PROJECT)/codes/dma\_axi\_simple/desing/verilog
- ◆ Step 3: compile and run
  - ❏ [user@host] cd \$(PROJECT)/codes/dma\_axi\_simple/sim/modelsim
  - ❏ [user@host] make
- ◆ Step 4: waveform view
  - ❏ [user@host] gtkwave wave.vcd &

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

Copyright © 2015-2017-2018 by Ando Ki

AXI DMA ( 14 )