Verification of a Cache Coherent system with an A53 cluster using ACE VIP with Graph Based Stimulus

Galen Blake
Perry Wobil
Altera Corporation

September 18, 2015
Austin, TX
Verification of Cache Coherent Access from an FPGA
Graph Based Multi Context Stimulus Concepts
Application to a Cache Coherent SOC-FPGA system
Verification of Cache Coherent Access from an FPGA
System Description

• An ARM quad A53 cluster

• A Cache Coherent Interconnect to coordinate coherent transactions

• Customer Defined V8 compliant ACE agent synthesized into the FPGA soft Logic
Block Diagram of Test Bench & DUT

- **ARM A53 MPU Cluster**
  - CPU0
  - CPU1
  - CPU2
  - CPU3
  - SCU
  - L2 Cache

- **FPGA Customer Design**
  - Cache

- **Cache Coherent Interconnect**

- **System Memory**

- **L3 Interconnect**

---

SNUG 2015
Verification Challenges

• How do we model a customer defined ACE agent
  – In our case we chose the Synopsys ACE VIP
  – It can be configured in many different ways to match potential customer configurations
  – A key feature needed is the local cache to support all ACE coherent transactions that require data allocation.

• How do we generate C and UVM stimulus, coordinate and check cacheable transactions
  – In our case we chose the Breker Trek-SoC tool
  – It is capable of generating C code for the A53 and UVM sequences for the ACE VIP and automatically coordinates and checks the coherent traffic from each of them
Graph Based Stimulus Concepts
Graph Construction

- Graphs are constructed from a C like language that describes a functional space or protocol
- The language is then used to construct a graph
- The graph can be used to review for accuracy and documentation
- This is an example of an ACE graph
Graph Elements

- Decision points
- Transactions
- Transaction Properties
Graph Construction

- A big picture view of the complete ACE graph
- The language used is currently proprietary
- There is an active working group in Accelerra [pswg] working to standardize this type of language
Application to a Cache Coherent Interconnect System
Block Diagram of Test Bench & DUT

DUT

- CPU0
- CPU1
- CPU2
- CPU3

SCU

L2 Cache

AXI4 Cache Coherent Interconnect

TrekSoc

Backdoor_access

AXI_master_trek_sequence

Trekest.c

Cache

Synopsys

ACE_VIP_m

ACE_L_VIP_s
(Memory)

ACE_L_VIP_m

ACE_VIP_MON

ACE_VIP_MON

ACELite

ACELite

OCRAM

AXI4

AXI4_VIP_MON

C-A53

MPU

ACE

ACE_L_VIP_s

Synopsys

ACE_VIP_m
Stimulus Generation & Synchronization

TrekSoC

CacheCoherencyApp

trek_test.c

Embedded C

ARM CPU

ACE

Trek Mail Box

UVM VIP Sequence

ACE VIP Master

ACE UVM Transactions

import "DPI-C"

Cache Coherency Interconnect
Big picture view.
Shows A53 and UVM events
Tracing activity on one cache line

Coherent transaction initiate on VIP to cause snoops on target address

Chain of Coherent TXNs that operate on target cache line

Coherent block transaction initiated on CPU which targets a memory cache line
Tracing activity on one cache line
Details about CPU based events

```c
// cache.2

trek_tbx_event( cpu0_mbox, 159 ); // [event:159 cpu:cpu0 thread:T2 instance:cache
/* tbx: trek message ("Begin cache.2"); */

/* initialize data for chain */
WMEM32 ( trek_mem_APS2MPFE+0x6ffffff40, 0x7f1e07db ); // trek_write_memory ()
/* READ MISS (exclusive) on trek_mem_APS2MPFE+0x6ffffff40 */
trek_nop_delay ( 6 ); // pre-delay
if ( MEM32 ( trek_mem_APS2MPFE+0x6ffffff40 ) != 0x7f1e07db ) { // trek_check_memory
trek_tbx_arg ( cpu0_mbox, MEM32 ( trek_mem_APS2MPFE+0x6ffffff40 ) );
trek_tbx_event( cpu0_mbox, 160 ); // [event:160 cpu:cpu0 thread:T2 instance:cache
/* tbx: trek_check_memory ( trek_mem_APS2MPFE+0x6ffffff40, 0x7f1e07db, 2 ); */
}

/* READ HIT on trek_mem_APS2MPFE+0x6ffffff40 */
if ( MEM32 ( trek_mem_APS2MPFE+0x6ffffff40 ) != 0x7f1e07db ) { // trek_check_memory
trek_tbx_arg ( cpu0_mbox, MEM32 ( trek_mem_APS2MPFE+0x6ffffff40 ) );
trek_tbx_event( cpu0_mbox, 161 ); // [event:161 cpu:cpu0 thread:T2 instance:cache
/* tbx: trek_check_memory ( trek_mem_APS2MPFE+0x6ffffff40, 0x7f1e07db, 2 ); */
}

/* EXCLUSIVE expected for trek_mem_APS2MPFE+0x6ffffff40 */
trek_tbx_event( cpu0_mbox, 162 ); // [event:162 cpu:cpu0 thread:T2 instance:cache
/* tbx: trek message ("End cache.2"); */

WMEM32 SHARED(trek cpu0 test T2 state, 0x2);
break;
case (0x2):
```
Tracing activity on one cache line
Details about a VIP (UVM) event

```c
// snoop_ext_check.1

WMH32.Shared(trek_cpu0_test_T1_state, 0x5);

if (WMH32.Shared(trek_cpu0_test_T2_state) < 0x2) break;

trek_tbx_event( cpu0_mbox, 89 ); // [event:89 cpu:cpu0 thread:T1 instance:snoop_ext_check.1]

/* vip check (trek_mem_APS2MPFE+0x6ffffff40, 4, 7f1e07db) */

/* READENCE axi0.LD_partial_cache ARLOCK=NORMAL ARCACHE=R_NOALLOC_WBACK ARDOMAIN=INNERSHAREABLE ARPRT=DATA_NON_SECURE_PRIVILEGED */

WMH32.Uncached( trek_mem_APS2MPFE+0x6e88a640, 0 );

/* txb: trek_drive_io_txn_and_wait (AA_path, ARADDR, ARBURST, ARCACHE, ARDOMAIN, ARID, ARLEN, ARREGION, ARSIZE, ARUSER, burst_size_in_bytes, coherent_xact_type, direction, port, read_strb, req_a total_length_in_bytes, valid_length_in_bytes, xact_type); */

AA_path = READENCE axi0.LD_partial_cache ARLOCK=NORMAL ARCACHE=R_NOALLOC_WBACK ARDOMAIN=INNERSHAREABLE ARBURST=INR ARPRT=DATA_NON_SECURE_PRIVILEGED

ARADDR = 2147483456
ABURST = 1
ARCACHE = 11
ARDOMAIN = 1
ARID = 8
ARLEN = 1
ARLOCK = 0
ARPRT = 3
ARQOS = 0
ARREGION = 0
ARSIZE = 2
ARUSER = 2
burst_size_in_bytes = 4
coherent_xact_type = 1
direction = LOAD
port = fpga2aps_ace_master
read_strb = _ff
req_address = 2147483456
req_length_in_bytes = 4
total_length_in_bytes = 4
valid_length_in_bytes = 4
xact_type = 3 */
```
Runtime view of a failed event
Conclusion

• It only took one week to complete the following
  – Install the Breker Trek tool
  – Integrate the tool into embedded SW (Make) flow
  – Integrate the tool with the Synopsys VIP UVM sequences
  – Develop highly complex coherent system traffic

• We also found an RTL bug that second week

• Since that time we have also deployed both the Breker Trek and Synopsys ACE and DDR VIP to develop complex traffic for our memory sub system

• Currently working to extend this verification methodology of graph based stimulus and UVM components to “non-AMBA” areas of SOC
Thank You