Tracing in TLM based SystemC models



Rocco Jonack, Eyck Jentzsch MINRES Technologies GmbH



Presentation Copyright Permission

 A non-exclusive, irrevocable, royalty-free copyright permission is granted by MINRES GmbH to use this material in developing all future revisions and editions of the resulting draft and approved Accellera Systems Initiative SystemC standard, and in derivative works based on the standard.



Company

- MINRES is an enabling company, dedicated to providing the expertise and solutions required for improving development productivity:
 - Methodology services + productivity IP
 - Design consulting services & value added application engineering
 - ISO26262 compliant RISC-V RTL IP
 - Cloud based hybrid simulation solution (RTL/VP)
- MINRES Technologies GmbH is a privately-held, remotefirst startup based in Germany



INTRODUCTION



Purpose of traces

- Traces are commonly well understood
 - Accepted mechanism for debugging, performance analysis and documentation
- Traces are very widely used in HW design and verification
 - All HDL simulators support tracing implicitly
- SystemC contains default implementation for VCD tracing



Usage of traces

- Tracing signal type elements is straight forward
 - Bool, integer, bit constraint types and structs of those base types
 - VCD is the default choice, but many compressed formats exist
 - Open-source availability is important in many cases
- Tracing more complex elements like transactions is cumbersome in VCD
 - Transactions have many elements and time points
 - Phases as basic elements of transactions can overlap
- Why transaction tracing
 - Well formed format for transactions
 - Forms definition of data record with different timing points on a stream
 - Information on a stream can vary over time
 - Ties into TLM2 modeling concepts
 - Extension mechanisms can be transformed into trace structures
 - Can augment VCD



TRACING IN SCC - USE MODEL



Module sysc: waveform tracing

- SCC contains tracer module to automatically trace signals, ports, and variables
- Improved waveform tracing implementation(s)
 - Push waveform tracing implementation
 - Supports VCD
 - Supports FST
 - Compact format coming with gtkwave
 - Can be visualized using gtkwave, Impulse, or SCViewer



Module sysc: VCD push versus poll

- Default implementation polls each traced signal for changes
 - This is usually called pull approach
- SCC comes with an implementation which registers a method to each signal or port to be notified when changed
 - This we call push approach



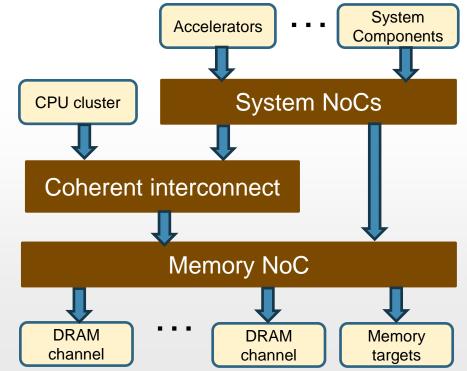
Module sysc: transaction recording

- Two implementations
 - Based on SCV
 - Based on Lightweight transaction recording (LWTR)
- Both support various backend implementations
 - Text format based on SCV reference implementation
 - Fast Transaction recording (FTR) a compact binary representation with compression
- Can be visualized using SCViewer or Impulse
- Various reference implementations read FTR format for further analysis



Transaction recorder

- Unit with 1 input and 1 output port
 - TLM2 no extensions; AXI, ACE and CHI with extensions
- Optional transaction tracing
 - Tracing is enabled if file handle was opened before construction
 - Low runtime overhead
- Can be integrated in existing systems
 - Integrator to include components
 - Top level must support file handle opening





Tracing setup

- Tracer automatically traverses object hierarchy
- Tracing is controlled by CCI parameters
- Opening database replaces default implementation
 - Registering values equivalent to default implementation

#include <scc/configurable tracer.h> #include <scc/configurer.h> int sc_main(int argc, char* argv[]) { // simple configuration scc::configurer cfg("system.yaml"); scc::configurable tracer trace("tgc tb rtl", scc::tracer::FTR, true, true); . . . sc core::sc start();



Tracing control

- Configurable tracer can control tracing by hierarchy based on CCI parameters
- Optional argument to database constructor can control time window of tracing

<pre>#include <scc.h></scc.h></pre>
<pre>int sc_main(int argc, char* argv[]) {</pre>
sc core::sc trace file* trc = scc::scc create vcd trace file("my vcd",
[]() -> bool {
// start tracing after 2us
return sc_core::sc_time_stamp() >= 2_us; }
);
<pre>sc_core::sc_start();</pre>
}



TRACING - VISUALIZATION



Trace visualization I

- SCViewer available on github as opensource
- Reads transaction and signal traces
- Contains waveform, table, details and hierarchy view

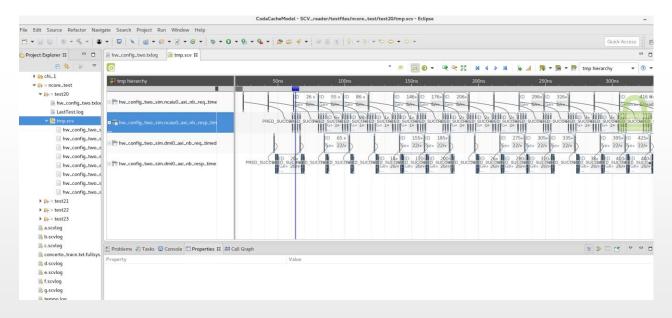
			SC Viewer				
ile Edit Navigate Window	Help						
a 😪 占 i 🗙 🗖 🗛 🔻 i 💽 🤅	2						
tgc tb rtl.fst ×							
					Next in stream 👻	• • # • 🔀	1
Enter text to filter hierarchy	Name	Value	43000.0ns	44000.0ns	Enter text to	filter attributes	
🛶 pc 🕞 dBusTat	tb.router.targ	et_0_b read			Name	Value	Туре
iBusAdapt	tb.router.targ	et_0_b read			 Properties 		
iBusTgt	tb.router.targ	et 1 b read			Name	tb.router.target_1_	
🕨 🚞 router	tb.router.targ	et 1 b read			Туре	read	String
🔻 🔚 uart0	tb.adapter.com		11			e 44400.0ns e 44400.0ns	Time Time
🖾 regs				(.	✓ Attributes	e 44400.0hs	Time
蝳 socket_bl	tb.uart0.socke	-		~		ay 100000 [0x186a0]	
socket bl timed	tb.uart0.socke	et_bl_ti read			start de	-	UNSIGNE
Enter text to filter wavefor		< ⊂			trans		ONSTONE
™irq_o	Property to m	atch: 👻	Proper	ty to show:	 Incoming 	el	
≖rst_i	Tx ID	Start time Pro	nerty Value		▼ Outgoing	el	
socket_bl	tx#25	1700.0ns 598	. ,		PARENT	CI tx#1006[44400.0	read
socket_bl_timed	tx#29		3435456 [0x10000000]		PRED/S	JC tx#1004[44400.0	read
socket_nb	tx#41	2300.0ns 599					
socket_nb_timed	tx#45		3435460 [0x10000004]				
™tx_o	tx#75		3451832 [0x10003ff8]				
çe 🦕	tx#79	3900.0ns 268	3451828 [0x10003ff4]				
Elements		C	: 44400.0ns M		0ns C-M:	44300.0ns	of 64M

https://github.com/Minres/SCViewer/releases



Trace visualization II

- Impulse as open source tool for visualization
- Plugin to eclipse
- Variety of input formats
- Commercial usage and integration based on licensing





Traces visualization - gtkwave

- gtkwave is the default choice open-source tool for signal trace viewing
- Reads fast and efficiently signal traces
 - Specifically fst format
 - Provides utilities to convert from and to VCD



Traces visualization - Verdi

- Verdi is a common tool in the marketplace
- Proprietary transaction format
 - Converter from open-source transaction format to proprietary format through FSDB API
- Alignment with RTL analysis

e View Stream Transaction Tools Wi	index Melo						ne/rocco/L								0 7 8 Menu
															O a D Menu
> 🖉 👋 🕄 🗆 🕄 🖕 🙇 📍	1 篇 4 图 4 0	x lns	A 🔣 🔍 🍕		 भूभू ।) 📕 🖛 📕	💁 🚯 E							
t Tree Quick Filter Protocols				1, 350	L.L.L.	1,400 . 1		L,450	1.1.1	L,597, I .	$1 \cdot 1 \cdot 1_{1}$	0	1, 1,600	T Detalls	
Hierarchy	Count	📕	bel" bel" b	1 Slabel	aviabel'el"	hel'be	1 bel".	bel'bel	bel h	el"bel".	hel" bel	.hel"i.bel	belbel Sla		
	1,000	t_reg_timed				4			<u>u</u> u					Attribute	Value
E C top1 501	1,000	resp_timed				1 1 1 I			10-10				· [1" [hel"	\$label	'nyLabel"
	1,000		. bel" bel		1. beri.t	al". bel"		i bel'	hel'ihel'		1º. bel". b	al'i bal'i.	bel*Slabel: 'nyLei	SbeginTime	1350
	5,500	_reg_timed						× *	<u> </u>			N 14	<u> </u>	ŞendTime	1356
 axi mst nb req. 33, 	200			1.1 1.	1. 1. 1.	1. 1. 1.	1.1. 1.1	·	. r j. r	1.1.	r 1. r 1.	r . r	.1* Slabel:"mgl	\$type	"Transaction"
axi_nst_nb_res 133	3,200	resp_timed												"trans.addres	71680
axi_slv_nb_bw 500														"trans.connan	"tim::TLM READ CONMAND""
- e axi_slv_nb_fw 500 - e axi_slv_nb_req 100														"trans.data 1	64
axi_siv_nb_req. 100 axi_siv_nb_res. 400														"trans.respon	"tlm::TLM INCOMPLETE RESPO
														"trans.dmi_al.	0
														"trans.byte_s.	
														"trans.strean	
														"trans.gp_opt	"tim::TLM MIN FAYLOAD""
														8	
	1.1														



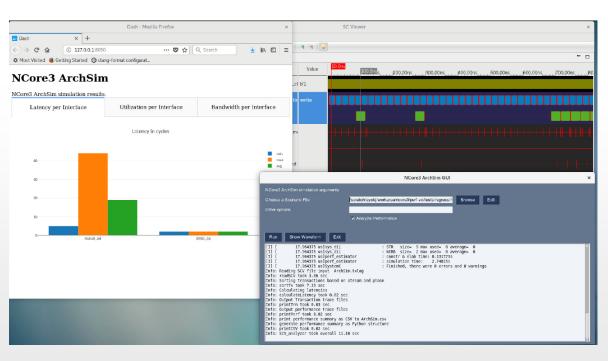
Transaction trace analysis

- Parsing and postprocessing of transaction traces
 - Well defined interfaces allow reliable post processing
 - Split transaction recording from analysis step
 - Post processing without simulation impact
 - Partial transaction analysis
 - Compression handling
 - Flexible output format
- Python script implementation
 - Example handles AXI, ACE and CHI
 - Outputs can be transaction journal, performance summary and STL per socket



Trace analysis visualization

- Trace analysis output can be used by opensource visualization tools like dash
- Python libraries allow simple analysis and even simulation control interfaces





TRACING - SUMMARY



Results of signal trace implementations

- Comparing in columns signal traces default, without signal duplication, push interface and FST implementation
- comparing in rows impact on simulation time, size of generated file and Iz4 compression (where applicable)

	SystemC VCD sc_trace	SCC VCD no duplication		SCC VC	CD push	FST		
simulation time (s)	23,01	14,18	61,63 %	11,86	51,54 %	13,57	58,97 %	
file size (Mbytes)	561,15	206,51	36,80 %	206,92	36,87 %	5,77	1,03 %	
compressed file size (Mbytes)	164,33	36,20	22,03 %	38,36	23,34 %			



Results of transaction trace implementations

- Comparing in columns transaction traces as text (txlog), binary enconded (txftr) and compressed binary encoded (cxftr)
- comparing in rows impact on simulation time, size of generated file and runtime of postprocessing

	txlog	tx	ftr	ctxftr		
simulation time (s)	163,82	121,16	73,96 %	125,00	76,31 %	
file size (Mbytes)	496,89	60,06	12,09 %	8,02	1,61 %	
SCV TX read time (s)	42,70	12,71	29,77 %	13,51	31,64 %	
SCV TX overall (s)	49,93	18,89	37,83 %	20,35	40,76 %	



Wishlist

- Alignment around transaction recording format
- Better VCD implementation as part of SystemC standard
- Alignment with formats used by RTL simulators



BACKUP

