

Automatically synthesizing higher level of protocol abstraction for faster debug and deeper insight into modern digital designs

Alasdair Ferro, Amar Patel, Chris Jones, Yogesh Badaya

{alasdair_ferro, amar_patel, chris_jones, yogesh_badaya}@mentor.com

Mentor Graphics, a Siemens Business

Abstract- The design and verification complexity has led to the evolution of various languages and methodologies, such as SV and UVM. This evolution happened primarily because of the raised level of abstraction at which design and verification engineers have to think and capture the intent. Such abstraction simplifies the comprehension and debug activities of the system. Still, as designs go through the flow, they are transformed into lower level of abstraction (logic synthesis and instrumentation), that could be based on context in which this data is captured, such as third-party sources, or hardware accelerator-based data capture. The analysis and debug complexity again goes up due to unavailability of higher level of data abstraction.

In this paper, we propose to synthesize the higher level of abstraction from the lower level of design data and making design data available for the debug and analysis. In our experiment, we captured AXI signallevel activity from emulation system and synthesized higher level of transactions. With this view, users can debug and analyze AXI data with the packetized transactions instead of signal-level waveforms. Though the methodology is demonstrated on AXI but is applicable to all the industry standard protocols. We have also demonstrated that this higher level of data can be easily analyzed for other insights into the subsystem like instructions coverage, address ranges, and performance analysis.

Keywords—protocol, IP, VIP, AXI, NASTI, RISC-V, SoC, SIP, emulation, prototype, signal activity, bus, transactions, processor performance evaluation

I. Introduction

There are various situations where design data is available only at lower level of abstraction. For example when waveforms are captured from emulation or FPGA Prototype or any other hardware accelerator during design verification, data is available as signal-level activity. Having only lower level of abstraction makes it difficult to understand the design behavior to debug a failure, capture coverage, and map to test plan, amongst other verification tasks.

Typically, the only option available to engineers is look at the signal-level waveforms and make progress. This requires protocol expertise to be effective, and increases the time it takes to close the task at hand.

In some other cases, engineers can use tools that have been matured to give better representation of the information, mapped to original higher level of abstraction. However, such tools typically work only in few cases and that too when the implementation of that tool started with higher level of abstraction, and has all the relevant information. For example, if the tool itself is driving UVM sequences into the emulation box, it already knows the details that can map the activity back. Still, such tools offer limited features and work for only some specific cases. The problem keeps growing as higher number of standard IPs become part of the sub system. We have identified the following key issues with the existing flow:

- 1. Need of protocol expertise to work at signal level of protocols with the increase in number of protocols and the complexity of the protocols.
- 2. Extra effort is needed to make sense of the data. This increases the debug complexity and time to functional correctness.



3. Data analysis at various levels is a critical part of the verification closure. This includes test plan coverage, code coverage and other protocol specific analysis. Signal-level data makes it complex to work on these parameters effectively.

II. Proposed Solution

We propose methodology to synthesize higher level of protocol abstraction automatically from the signal level data and protocol configuration captured at run time. With this approach, we take care of the productivity issues with verification as described earlier.

We provide a flow as illustrated in Figure-1. Our solution reads signal-level activity and protocol configurationspecific information. Our tool can transform this information into higher level of abstraction. This synthesis technology makes verification faster by providing the following features to users:

- Graphical visualization to comprehend and debug the data at the transaction level.
 - o Transaction Viewer
 - o Protocol Specific Debugger
- Additional utilities to assist with data analysis based on protocol. They help in protocol-specific checks, and scenario coverage.
- Features that make it easy to visualize design activities at the higher level of abstraction so that verification engineers do not need to have deep knowledge of all the protocols.



Figure 1 Protocol Synthesis

III. Experiment and Results

The experiment was done on a RISC-V processor based sub system, Figure-2 describes the block diagram of the sub system. The design is synthesizable and can be executed on hardware based verification technologies like emulation or FPGA prototype. Waveforms can be captured for any bus or a list of signals in the design.

We captured the signal-level activity at the bus as marked in Figure-2 with a star in red box. The RISC-V design and verification flow was instrumented to capture data in our propriety waveform format (qwave.db). The chosen buses are UART and AXI-based communication channel between processor (Rocket Tile) and the memory devices. The software program loaded in the RAM and the boot program are executed on the processor and data is transferred using the chosen bus.





Figure 2 RISC-V based sub system

Figure 3 and Figure 4 display signal-level activity and corresponding synthesized transactions for AXI Write and AXI Read, respectively. The synthesized transactions contain packet-level information and is helpful in debugging and searching a specific transaction activity. During simulation, a transaction log is also generated in text version of the synthesis report.

Stupint hall Values - Cl 1350 <t< th=""><th></th><th>lit View Optio</th><th></th><th></th><th>2 -1 DOLLO</th><th>(a) -2 (a)</th><th>Entra E</th><th>ile Edit</th><th>: View Opti</th><th>ions Tools</th><th>Windo</th><th>м</th><th></th><th></th><th></th><th></th></t<>		lit View Optio			2 -1 DOLLO	(a) -2 (a)	Entra E	ile Edit	: View Opti	ions Tools	Windo	м				
Signal hase Values-c1 356 1350 13400 1050 13400 1050 13								2 6 0			Te et	1. 31 IS all	a 13581	A C2 D	▲ Diff 13581 1ns	▼ Fr
NVALIO 0 WALST MAXST Burst MAXATATORN Burst MAXATATORN Cache HAA AXIA_INGRNAL Cache HAA AXIA_INGRNAL Cache HAA AXIA_INGRNAL Gadar_user_data BAA MUTST-Lingth 3 Gadar_user_data BAA Maxia_Nords BIA Gadar_user_data BAA Gadar_user_data BAA Gadar_user_data BAA Gadar_user_data BAA Gadar_user_data <t< th=""><th></th><th></th><th>Values-C1</th><th>14130</th><th>14140</th><th>14150</th><th>1 2</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>13620</th></t<>			Values-C1	14130	14140	14150	1 2									13620
With f With f<	Tra		0							values-C1	1336	,	13390	13000	13010	13020
witzel 0 8011 0 8011 0 8011 addr 8/A 1310 witze 0 8859 0 prot 8/A 8/A 8/A 8/A 0 8859 0 bresp_stelay 10/A AVI4_NORSEC_MTA 0 8100 0 0 1 bresp_stelay 10/A AVI4_NORSEC_MTA 0 0 1 0 0 1 bresp_stelay 10/A AVI4_NORSE_CANTA 0 0 1 0 0 1 1 bresp_stelay 10/A AVI4_NORSE_CANTA 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1				001400000*/20001	0040*\b400000			Trans								
write ddlay: write n -				A00000 A00001	0040 <u>A</u> 04000000	ATTREBUGGE /							0	20121020202	0474775408647-775-86	140 435034
addr V/A s180 bresp_delay V/A Att_NOMS_SEC_DATA bresp_delay V/A Att_NOMS_SEC_DATA control RED 0 hresp_delay V/A Att_NOMS_SEC_DATA bresp_delay V/A Att_NOMS_SEC_DATA bresp_delay V/A Att_LEYTES_16 bresp_delay V/A Att_LNOM bresp_delay V/A Att_LNOM bresp_delay V/A Att_LNOM cache V/A Att_LNOM qos V/A Att_LNOM qos V/A Att_LNOM qos V/A 3 burst_longth V/A 3 ditoortio V/A Att_LNOMODITIABLE_NOMBUF qos V/A 3 ditoortio V/A Att_LNOMODITIABLE_NOMBUF qos V/A 3 ditoortio Att_LNOMODITIABLE_NOMBUF Att_LNOMODITIABLE_NOMBUF qos V/A Gos Att_LNOMODITIABLE_NOMBUF					write		_						0	3013403038-76	813423100-A CC43C2310-A	0843585
op of the spinol lay MAA XILLNORE_SEC_DATA Image: spinol lay MAA XILLNORE_SEC_DATA bresp_olay MAA 0 Image: spinol lay I				c100												
Dress_olity MA 0 Image: model of the second of the sec					ТА							<u> </u>				
region N/A 0 res 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 <	_		N/A	0								<u>е</u> <u>А</u>				
Size V/A Ax1L_DYTE_1.16 1540 burst W/A Ax1L_NORM 1540 lock W/A Ax1L_NORMAL 1772 lock W/A Ax1L_NORMAL 1772 lock W/A Ax1L_NORMAL 1772 lock W/A Ax1L_NORMAL 1772 dotable W/A Ax1L_NORMAL 1772 gos W/A X/A X/A gos W/A X/A write_strokes W/A X/A write_strokes W/A X/A gos 0 11			N/A	θ												
burst MAXI_JNCR prof. SEC.DNA ACL_MORE SEC.DNA ACL_MORE SEC.DNA SEC.DNA Gache H/A S SEC.DNA Jdd H/A 3 SEC.DNA SEC.DNA Burst_Unpth H/A 3 SEC.DNA SEC.DNA did_subsr_data H/A 3 SEC.DNA SEC.DNA did_subsr_data H/A 3 SEC.DNA SEC.DNA did_subsr_data H/A 0 SEC.DNA SEC.DNA did_subsr_data 0 SEC.DNA SEC.DNA </td <td>_</td> <td>size</td> <td>N/A</td> <td>AXI4_BYTES_16</td> <td></td> <td></td> <td></td> <td>Π</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>read</td> <td></td> <td></td>	_	size	N/A	AXI4_BYTES_16				Π						read		
lock M/A XX12_MORMAL Fresp_Galay Tresp_Galay Tresp_Galay <thtresp_gal< td=""><td>_</td><td>burst</td><td>NZA</td><td>AXI4_INCR</td><td></td><td></td><td></td><td>†1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></thtresp_gal<>	_	burst	NZA	AXI4_INCR				† 1								
C CACHO N/A AXIA_DOMODIFINIL_DOMODIFINIL_DOMODIF 0 go N/A A P region 0 0 1 d N/A 3 Size	_	lock	N/A	AXI4_NORMAL				П								
dds i//A 3 ut i//A 3 burst_length i//A 3 burst_length i//A 3 ud i//A 3 burst_length i//A 3 burst_length i//A 3 burst_length i//A 3 burst_length i//A (129 'hoooooooooooooooooooooooooooooooooooo	_	cache	NZA	AXI4_NONMODIFIAB	LE_NONBUF			H				32'h2 32'h0 3	32'h0 32'h0 32	2'h0 32'h0 32'h0 32'l	0 32'h0 32'h0 32'h0 32	2'h0 32'
L U	_	qos	NZA	Ð				P -1				0				
OWN PLAND OWN PLAND AXIA (MORAL) AXIA (MORAL) AXIA (MORAL) addr_word_lata_word_sin_/A (10*MFFF 10*MFFF 10*MFFF 10*MFFF) 0 <t< td=""><td>_</td><td>id</td><td>NZA</td><td>3</td><td></td><td></td><td></td><td>H</td><td></td><td></td><td></td><td></td><td>6</td><td></td><td></td><td></td></t<>	_	id	NZA	3				H					6			
Manual Jun	_	burst_length	N/A	3				H								
wills_strides N/A (dehther is hereff is	_	addr_user_data	NZA	θ				H	lock	_						
wdata_user_data N/A (arba 4*be 4*be 4*be) 0 1 1 resp_user_data N/A (arba 4*be 4*be) 0	_	data_words	NZA	{128'h0000000000	909E140000000	000000004 128'h00006	00008*	H	cache	BLE_NONBUF		AXI4_NONMODI	FIABLE_NONBUF			
resp N/A AVIL 06AV P brit Lingth 3 5 resp_user_data N/A 0	_	write_strobes	N/A	{16'hFFFF 16'hFF	FF 16'hFFFF 10	6'hFFFF}		• -	qos	Θ		θ				
addr_start_time N/A 44327 ns (121*M0313403240031303030078513647793 121*M021342370019130000078404 addr_start_time N/A 44327 ns (14152 ns 14151 ns) (121*M0313403240313031000078513647793 121*M0213423700191300000078404 addr_start_time N/A (14132 ns 14139 ns 14145 ns 14151 ns) - - resp. 441_0.01 NJ (241*0431340347011_0000078510813000000784047 addr_start_time N/A (14132 ns 14139 ns 14145 ns 14151 ns) - - resp. 441_0.01 NJ (41n 4 no 4 no 4 no) wr_resp.start_time N/A (14137 ns 14157 ns - - addr_start_time 13847 ns wr_resp.start_time N/A 14157 ns - - addr_start_time 1387 ns wr_resp.start_time N/A 14157 ns - - addr_start_time 1387 ns wr_resp.start_time N/A 0 - - addr_start_time 1387 ns wr_resp.start_time N/A 0 - - addr_start_time 1322 ns) 13867 ns 13617 ns 13623 ns) tart_scat_id_olay N/A 0		data_user_data	NZA	{4'h0 4'h0 4'h0	4'h0}			• -	id	1		1				
addr_start_time N/A 44327 ns (121*M0313403240031303030078513647793 121*M021342370019130000078404 addr_start_time N/A 44327 ns (14152 ns 14151 ns) (121*M0313403240313031000078513647793 121*M0213423700191300000078404 addr_start_time N/A (14132 ns 14139 ns 14145 ns 14151 ns) - - resp. 441_0.01 NJ (241*0431340347011_0000078510813000000784047 addr_start_time N/A (14132 ns 14139 ns 14145 ns 14151 ns) - - resp. 441_0.01 NJ (41n 4 no 4 no 4 no) wr_resp.start_time N/A (14137 ns 14157 ns - - addr_start_time 13847 ns wr_resp.start_time N/A 14157 ns - - addr_start_time 1387 ns wr_resp.start_time N/A 14157 ns - - addr_start_time 1387 ns wr_resp.start_time N/A 0 - - addr_start_time 1387 ns wr_resp.start_time N/A 0 - - addr_start_time 1322 ns) 13867 ns 13617 ns 13623 ns) tart_scat_id_olay N/A 0	_	resp		AXI4_OKAY				e -	burst_length	3		3				
addr_end_Line N/A Lill32 ns (All32 ns data_start_Line N/A Lill33 ns 14139 ns 14145 ns 14151 ns) (All_0AV AUL_0AV AUL_0AV AUL_0AV AUL_0AV) data_start_Line N/A Lill33 ns 14139 ns 14145 ns 14151 ns) (All_0AV AUL_0AV AUL_0AV) wr_resp_start_Line N/A Lill57 ns addr_start_Line 13581 ns wr_resp_start_Line N/A Lill57 ns addr_start_Line 13581 ns dress_vall_d_clay N/A 92 addr_start_Line 13587 ns ress_to_data_clay N/A 0 ads_end_Line 13587 ns wr_resp_start_Line N/A 0 ads_end_Line 13587 ns		resp_user_data	N/A	θ				🖷 ad	ldr_user_data	θ		0				
dat_start_time N/A (4432 ns 14130 ns 14145 ns 14151 ns) resp. user_dist i/h0 4/m0 (1412 ns 14150 ns 14150 ns 14151 ns) wf_resp_start_time N/A (4132 ns 14130 ns 14145 ns 14151 ns) resp. user_dist i/h0 4/m0 (1410 ns 14150	a	ddr_start_time						H	data_words	3FCD43823}		{128'h030134	33038130830007	7851300478793 128'h02	813423FD01011300008067	0401011
data_end_time N/A (14133 ns 14139 ns 14145 ns 14151 ns) Top_post_and = (no + no) Top_post_and = (no + no) Top_post_and = (no + no) wr_resp_start_time N/A 1197 ns 3381 ns 3387 ns 13881 ns wr_resp_start_time N/A 1197 ns 3407_start_time 1387 ns 1387 ns ddress_valid_start_time 1387 ns 322 data_start_time 13523 ns) 1381 ns 1361 ns 13627 ns) ress_to_data_delay N/A 0 0 data_start_time 13523 ns) 13581 ns 1361 ns 13627 ns 13627 ns)								H	resp	AXI4_OKAY}		{AXI4_OKAY A	XI4_OKAY AXI4	_OKAY AXI4_OKAY}		
wr_resp_start_time N/A ills7 ns adf_red[1:s] N/A ills7 ns wr_resp_end_time N/A 1457 ns adf_red[1:s] ns 1387 ns ddress_valid_celay N/A 02 ddt_start_time 1387 ns 1387 ns ress_to_data_elay N/A 0 odd oddress_valid_delay 234 ress_colay N/A 0 oddress_valid_delay 234 244	d							- re	sp_user_data	4'h0 4'h0}		{4'h0 4'h0 4	'h0 4'h0}			
wr_resp_end_Lise N/A 14157 ns addr_end_Lise 1397 ns ddress_valid_delay N/A 932 ditas_tart_tas 1387 ns 1387 ns dress_valid_delay N/A 932 ditas_tart_tas 1382 ns) 1386 ns 1381 ns 13817 ns 1362 ns) dress_to_data_delay N/A 0 ditas_tart_tas 1382 ns) 1386 ns 1381 ns 13817 ns 1362 ns)					ns 14145 ns 14	4151 ns}		add	lr_start_time	13581 ns		13581 ns				
doress_valid_dolay N/A 932 data_start_time 1362 ns) (1366 ns 1361 ns 1367 ns 1362 ns) tress_to_data_dolay N/A 0 data_start_time 1362 ns) (1366 ns 1361 ns 1361 ns 1367 ns)								- a	ddr_end_time	13587 ns		13587 ns				
ddress_valid_delay IV/A 092 ress_t_odiato_s_delay IV/A 0 a_tc_address_delay IV/A 0								- dat	a_start_time	13623 ns}		{13605 ns 13	611 ns 13617 n	ns 13623 ns}		
Tres_to_data_olay N/A 0 addres_vala_delay 234				932												
a_to_address_delay N/A 0				θ												
				0				-								

Figure 3 AXI Write Transaction

Figure 4 AXI Read Transaction



The communication bus is between the processor (Rocket Tile) and the RAM, so the transaction data contains information about the nature of instructions executed on RISC-V processor. This enables a wide range of checks and analysis, which can be done on RISC-V-based designs. For our experiment, the data was analyzed to extract instruction coverage (shown in Figure 5) and address distribution (shown in Figure 6).



Figure 5 RISC-V Instructions Coverage





IV. Conclusion

Synthesis of transactions from signal-level activity provides a platform to perform design checks and design data analysis. The data collection is done during design run and synthesis is done as post process. The feature is useful for simulation, emulation and FPGA based-verification and validation tools. RISC-V, as an open ISA, is used to provide standard and independent processors. This capability can further be used for cache coherency, security, processor performance analysis and compliance checking in RISC-V based subsystems.

Though we have demonstrated our application on AXI and UART protocol, the methodology scales to all the industry standard protocols. This reduces effort and time to debug and achieve verification targets, hence verification closure of the designs.

V. References

- [1] RISC-V. https://riscv.org.
- [2] LowRISC https://www.lowrisc.org.
- [3] ARM AMBA AXI Protocol Specifications.