PS and PL communication using AXI DMA

For the course 1DT109 Accelerating System with Programmable Logic Components, 2022. Department IT, Uppsala University

Contact: yuan.yao@it.uu.se; shiming.li@it.uu.se; shiming.li@it.uu.se; shiming.li@it.uu.se; shiming.li@it.uu.se; shiming.li@it.uu.se; shiming.li@it.uu.se; shiming.li@it.uu.se

This document will walk you through

- 1. Understand what DMA is and why it is important for FPGA
- 2. How to transfer data between PS and PL using DMA with the correct components from the FPGA
- 3. How to exploit the above DMA feature in software (using the C programming language)

Overview: What is DMA and why we should use it for FPGA

In this design, we'll use the DMA to transfer data from memory to an IP block and back to the memory. In principle, the IP block could be any kind of data producer/consumer such as the HDR-NN hardware implement or just a matrix multiplier, but in this tutorial, we will use a simple FIFO to create a loopback. After, you'll be able to break the loop and insert whatever custom IP you like.

//TODO: think about what you want to implement in the PL side. You can implement the whole HDR-NN using Verilog and invoke it from the C. Or you can implement your HDR-NN in software and ship part of the functionalities to PL.



The block diagram above illustrates the design that we'll create. The processor and DDR memory controller are contained within the Zynq PS. The AXI DMA and AXI Data FIFO are implemented in the Zynq PL. The **AXI-lite** bus allows the processor to communicate with the AXI DMA to setup, initiate and monitor data transfers. The **AXI_MM2S** and **AXI_S2MM** are memory mapped AXI4 buses and provide the DMA access to the DDR memory. The **AXIS_MM2S** and **AXIS_S2MM** are AXI4-streaming buses, which source and sink a continuous stream of data, without addresses.

Notes:

- MM2S stands for Memory-Mapped to Streaming, whereas S2MM stands for Streaming to Memory-Mapped.
- When Scatter-Gather is used, there is an extra AXI bus between the DMA and the memory controller. It was left out of the diagram for simplicity.
- We'll start this tutorial with the base system project for the MiniZed that you've created in project lecture 2.

PART 1 – Setting up the DMA communication hardware platform

1, Add the AXI DMA

- 1. Open the base project in Vivado.
- 2. In the Flow Nagigator, click "Open Block Design".



- 3. The block diagram should open, and you should only have the Zynq PS in the design.
- 4. Make sure that the Zynq PS has master GPO enabled.

ZYNQ7 Processing System Occumentation Presets	n (5.5)		
Page Navigator	- PS-PL Configuration		
Zynq Block Design	← Q ≭ ≑		
PS-PL Configuration	Search: Q-		
Peripheral I/O Pins	Name > General	Select	Description
MIO Configuration	 AXI Non Secure Enablement 	0 🗸	Enable AXI Non Secure Transaction
Clock Configuration	 GP Master AXI Interface 		
clock configuration	> M AXI GP0 interface	\checkmark	Enables General purpose AXI master interface 0
DDR Configuration	> M AXI GP1 interface		Enables General purpose AXI master interface 1
SMC Timing Colculation	> GP Slave AXI Interface		
Sivic Tilling Calculation	> HP Slave AXI Interface		
Interrupts	> ACP Slave AXI Interface		
	> DMA Controller		

5. Connect the M_AXI_GP0_ACLK to the FCLK_CLK0 pin, which will be the driving clock of the master GP0 interface.



6. Click the "Add IP" icon and double click "AXI Direct Memory Access" from the catalog.

Search:	Q* axi direct	8	(4 matches)
👎 AXI	Central Direct Memory Access		
👎 AXI	Direct Memory Access		
👎 AXI	Multi Channel Direct Memory	Access	
👎 AXI	Video Direct Memory Access		
NTER to	o select, ESC to cancel, Ctrl+Q f	or IP details	

 The DMA block should appear and designer assistance should be available. Click the "Run Connection Automation" link and select /axi_dma_0/S_AXI_LITE from the drop-down menu.

omatically make connections in your design by ions on the right.	checking the boxes of the interfaces to connect. Select	an interface on the left to display its configurat	ation
Q ¥ ♦	Description		MAXLGPOACLK ZYNQ SDIO.0 +
 All Automation (1 out of 1 selected) 	Connect Slave interface (/axi_dma_0/S_AXI_LITE) to a selected Master address space. Options		M.AKI_GP0 + FCLX_CLX0 FCLX_CLX1 FCLX_REST0,N
	Master Bridge IP Clock source for driving Interconnect IP Clock source for Master interface Clock source for Slave interface	/processing_system7_0/M_AXQ_GP0 V New AXI Interconnect Auto /processing_system7_0/FCLK_CLK0 (50 MHz) Auto	ZYNQ7 Processing System axi.dma.0 + S.AXI.III M.AXI.SS + + S.AXI.SS.MM M.AXI.SS + + S.AXI.SS.MM M.AXI.SS + - M.AXI.SS
\rightarrow		ОК	Cancel AXI Direct Memory Access

8. Click "OK" in the window that appears. Vivado will connect the AXI-lite bus of the DMA to the General Purpose AXI Interconnect of the PS. Your block diagram should now look like the following.



2, Connect memory controller to the DMA

1. Now we need to connect AXI buses **M_AXI_SG**, **M_AXI_MM2S** and **M_AXI_S2MM** of the DMA to a high performance AXI slave interface on the PS. Our PS doesn't seem to have a high-performance AXI slave interface, so we need to change the Zynq configuration to enable one. Double click on the Zynq block.



2. Select "PS-PL Configuration", open the "HP Slave AXI Interface" branch and tick the "S AXI HPO interface" to enable it. Then click OK.

Ocumentation 🔅 Presets	IP Location			
age Navigator	PS-PL Configuration			Summary Re
nq Block Design	← Q <u>×</u> ≑			
-PL Configuration	Search: Q.			
	Name	Select	Description	
ripneral I/O Pins	> General			
O Configuration	> AXI Non Secure Enablement	0 ~	Enable AXI Non Secure Transaction	
	> GP Slave AXI Interface			
ock configuration	 HP Slave AXI Interface 			
OR Configuration	> S AXI HP0 interface	V	Enables AXI high performance slave interface 0	
M Timing Calculation	> S AXI HP1 interface		erformance slave interface 1	
ic riming Calculation	> S AXI HP2 interface		CHADICS PATTING Performance slave interface 2	
errupts	> S AXI HP3 interface		Enables AXI high performance slave interface 3	
	> ACP Slave AXI Interface			
	> DMA Controller			
	> PS-PL Cross Trigger interface		Enables PL cross trigger signals to PS and vice-versa	

3. The high-performance AXI slave ports should now be visible in the block diagram, and designer assistance should be available. Click the "Run Connection Automation" link and select /processing_system7_0/S_AXI_HP0 from the drop-down menu. Click "OK".

omatically make conn figuration options on	ections in your design by the right.	checking the boxes of the interfaces to connect. Se	elect an interface on the left to displ	lay its 🗼 🛛 🕻	छ 💉 C थ छ		
omatically make conn figuration options on Q ↓ Ξ ↓ Φ ✓ Ø All Automation ✓ Ø ₱ processin Ø ₱ § SAX Non mentation emented Design D DEBUG listream ware Manager	ections in your design by the right. (1 out of 1 selected) g_system7_0 L_HPO	checking the boxes of the interfaces to connect. Since the interface (processing_system space. Options Master Bridge IP Clock source for driving Interconnect IP Clock source for Slave interface Clock source for Slave interface Clock source for Slave interface	elect an interface on the left to displ m7_0/S_AXI_HP0) to a selected Mas //axi dma_0/M_AXI_MM2S ~ Auto Auto Auto Auto Auto Auto Cities ent_rest. Processing_system7, processing_system7,	ay its	g ★, C SI 45 ps7.0_axi_periph + 930.40 AdSITN AdSITN M00_ACL M00_ACL M00_ACL Additive Additive Additive Additive Additive Additive	AXI dma.0 MAM.MS H SANS.15 MAM.MM H SANS.15 MAXS.15 MA	
man munugu	Block Interface Pr	Perfits ? □<	+ SANJAPO FROCTRE + SANJAPO MARE CRO ACLE SANJAPO SANJAPO ACLE	GPI0_0 + DOR + IC_0 + UART_0 + UART_0 + USDIO_0 + USDIO_0 + USDIO_0 + M_R0_GPO + FCLK_CLKI FCLK_REFIN_N		AXI Direct Memory Access	-D DDF

4. Designer assistance should again be available, click the "Run Connection Automation" link and select **/axi_dma_0/M_AXI_SG** from the drop-down menu. Click "OK".



5. Designer assistance should still be available, click the "Run Connection Automation" link and select **/axi_dma_0/M_AXI_S2MM** from the drop-down menu. Click "OK".



6. Now all the memory mapped AXI buses are connected to the DMA. Now we only must connect the AXI streaming buses to our loopback FIFO and connect the DMA interrupts.

3, Add the FIFO

1. Click the "Add IP" icon and double click "AXI4-Stream Data FIFO" from the catalog.

	natches)
AXI4-Stream Accelerator Adapter	
AXI4-Stream Broadcaster	
AXI4-Stream Clock Converter	
AXI4-Stream Combiner	
🌻 AXI4-Stream Data FIFO	
AXI4-Stream Data Width Converter	
AXI4-Stream Interconnect	
AXI4-Stream Protocol Checker	
AXI4-Stream Register Slice	
AXI4-Stream Subset Converter	
AXI4-Stream Switch	
AXI4-Stream to Video Out	
AXI4-Stream Verification IP	
Video In to AXI4-Stream	
NTER to select, ESC to cancel, Ctrl+Q for IF	P details

 The FIFO should be visible in the block diagram. Now we must connect the AXI-streaming buses to those of the DMA. Click the S_AXIS port on the FIFO and connect it to the M_AXIS_MM2S port of the DMA.



3. Then connect the **M_AXIS** port on the FIFO and connect it to the **S_AXIS_S2MM** port of the DMA.



4. Now we must connect the FIFO clock and reset. Click the **s_axis_aresetn** port of the FIFO and connect it to the **axi_resetn** port of the DMA.



5. Click the **s_axis_aclk** port of the FIFO and connect it to the **s_axi_lite_aclk** port of the DMA.



6. In our design, we won't need the AXI-Streaming status and control ports which are used to transmit extra information alongside the data stream. You might use them if you were connecting to the AXI Ethernet core or a custom IP that made use of them. In the block diagram, double click the AXI DMA block. Un-tick the "Enable Control / Status Stream" option and click OK.

(I Direct Memory Access (7.1)	
Show disabled ports	Component Name axi.dma.0 Imable Adynchronous Clocks (Auto) Imable Adynchronous Clocks (Auto) Imable Statter Gather Engine Imable Multi Channel Support Imable Multi Channel Support Imable Multi Channel Support Imable Read Channel Imable Read Channel Imable Multi Size Imable Allow Unaligned Transfers Imable Single AXM Data Multi Atta Size Imable Single AXM Data Interface

4, Enable interrupt from the DMA

Our software application will test the DMA in polling mode, but to be able to use it in interrupt mode, we need to connect the interrupts **mm2s_introut** and **s2mm_introut** to the Zynq PS.

1. First, we must enable interrupts from the PL. Double click the Zynq block and select the Interrupts tab.

VNQ7 Processing System	(5.5) IP Location 🌣 Import XPS Settings			4
Page Navigator —	Interrupts			Summary Report
Zynq Block Design	+ Q ≚ ♦			
PS-PL Configuration	Search: Q-			
	Interrupt Port	ID	Description	
Peripheral I/O Pins	V 🗹 Fabric Interrupts		Enable PL Interrupts to PS and vice versa	
MIO Configuration	 PL-PS Interrupt Ports 			
	✓ IRQ_F2P[15:0]	[91:84], [68:61]	Enables 16-bit shared interrupt port from the PL. MSB is	
Clock Configuration	Core0_nFIQ	28	Enables fast private interrupt signal for CPU0 from the PL	
DDP Configuration	Core0_nIRQ	31	Enables private interrupt signal for CPU0 from the PL	
DDR conliguration	Core1_nFIQ	28	Enables fast private interrupt signal for CPU1 from the PL	
SMC Timing Calculation	Core1_nIRQ	31	Enables private interrupt signal for CPU1 from the PL	-
Interrupts	> PS-PL Interrupt Ports			

2. Tick "Fabric Interrupts" and IRQ_F2P[15:0] to enable them, and click OK.

3. Click the "Add IP" icon and double-click "Concat" from the catalog.

Search:	Q- concat	😣 (1 match
👎 Cond	at	

4. Connect the **dout** port of the Concat to the **IRQ_F2P** port of the Zynq PS.



5. Connect the **mm2s_introut** port of the DMA to the **In0** port of the Concat.



- xlconcat_0 In0[0:0] dout[1:0] n1[0:0] Connect from 's2mm_introut' port Cc to 'In1' port axi dma 0 axi_smo + S00_AXI + S_AXI_HP + S_AXI_HP S01_AXI a[31:0] M_AXI_SG + + S02_AXI M00_AXI M_AXI_MM2S + M_AXI_S2MM + M_AXI_GP0_ S_AXI_HP0_4 :p[3:0] X adk aresetr M_AXIS_MM2S + IRQ_F2P[0:0 mm2s_prmry_reset_out_n o AXI SmartConnect s2mm_prmry_reset_out_n • mm2s_introut • s2mm_introut ect Memory Access
- 6. Connect the **s2mm_introut** port of the DMA to the **In1** port of the Concat.

6, Validate and build the design

1. From the menu select Tools->Validate Design.



2. You should get this message saying that validation was successful.



3. Our block diagram now looks like this.



4. In the Flow Navigator, click "Generate Bitstream". It may take a few minutes to generate the bit stream. Below is the LUT utilization overview for the above design. Lightblue boxes indicate busy LUTs while dark ones the free LUTs, which you can use to implement your own design.



PART 2 – Programming the DMA system using C

1, Export the hardware design to SDK

Once the bitstream has been generated, we can export our design to SDK where we can develop the software application that will setup a DMA transfer, wait for completion and then verify the loopback.

1. In Vivado, from the File menu, select "Export->Export hardware".

I <u>m</u> port	•	
Expor <u>t</u>		Export <u>H</u> ardware
Launch SDK		Export <u>C</u> onstraints
Print	Ctrl+P	Export Pblocks
E <u>x</u> it		Export IBIS Model
		Export I/O Ports
		Export Bitstream File
RTL ANALYSIS		Export Simulation
Open Elaborated Dec	ian 📃 🖵	

2. In the window that appears, tick "Include bitstream" and click "OK".

À Export Hardw	are	×		
Export hardware platform for software development tools.				
✓ Include bits	stream			
<u>E</u> xport to:	<local project="" to=""></local>	~		
?	OK	Cancel		

- 3. Again, from the File menu, select "Launch SDK".
- 4. In the window that appears, use the following settings, and click "OK".

🔥 Launch SDK	×
Launch software development tool.	4
Exported location: Second to) Project> 🗸
Workspace: Second to Project	t> V
ЭОК	Cancel
2'	
Text E <u>d</u> itor	•
I <u>m</u> port	→
ြာ Project Explorer 🛛	□ 🕏 🔻 ▽ 🗖 🖬
~ / design_1_wrapper_hw_platform_0	
design_1_wrapper.bit	
g ps7_init_gpl.c	
🗎 ps7_init_gpl.h	
g ps7_init.c	
li ps7_init.h	
ps7_init.html	
ps7_init.tci	
ar system.nur	

- 5. At this point, the SDK loads and a hardware platform specification will be created for your design. You should be able to see the hardware specification in the Project Explorer of SDK as shown in the image below.
- 6. We are now ready to create the software application.

2, Create a software application

1. At this point, your SDK window should look somewhat like this:

Edit Navigate Search Project Run Xilinx	Window Help					
oject Explorer 🛛 🕒 🕏	▼ ▼ □ □ 🕞 system.hdf 🛛					-
design_1_wrapper_hw_platform_0	design_1_wrapper_hv	v_platform_(0 Hardware	Platform	Specification	
g ps7_init_qpl.c	Design Information					
s7_init_gpl.h						
Init.c	Target FPGA Device: 7z007	S D7eeleDDE 1				
▶ ps7_init.h	Part: XC/200 Created With: Vivad	2018 3				
ps7_init.html	Created On: Fri No	v 26 13:45:01 202	1			
ps7_init.tcl		1 20 10110101 202				
all system.nut	Address Map for processor	ps7_cortexa9_0				
	6-11	Deve Antes	LICEN A JUL	Claure 1.46	Marry (Dara	
	Cell	Base Addr	High Addr	Slave I/T	Mem/Keg	
	ps/_intc_dist_0	0000-000	0x18101111		REGISTER	
	ps7_gpi0_0	0xe000a000	0xe000a111		PEGISTER	
	ps7_sectime_0	0xf8000000	0xf8000fff		REGISTER	
	ps7_scuwdt 0	0xf8f00620	0xf8f006ff		REGISTER	
	ps7 I2cachec 0	0xf8f02000	0xf8f02fff		REGISTER	
	ps7_scuc_0	0xf8f00000	0xf8f000fc		REGISTER	
	ps7_qspi_linear_0	0xfc000000	Oxfcffffff		FLASH	
	ps7_pmu_0	0xf8893000	0xf8893fff		REGISTER	
	ps7_afi_1	0xf8009000	0xf8009fff		REGISTER	
	ps7_afi_0	0xf8008000	0xf8008fff		REGISTER	
	ps7_qspi_0	0xe000d000	0xe000dfff		REGISTER	
	ps7_usb_0	0xe0002000	0xe0002fff		REGISTER	
	ps7_afi_3	0xf800b000	0xf800bfff		REGISTER	
	ps7_afi_2	0xf800a000	0xf800afff		REGISTER	
	ps7_globaltimer_0	0xf8f00200	0xf8f002ff		REGISTER	
	ps7_dma_s	0xf8003000	0xf8003fff		REGISTER	
	ps7_iop_bus_config_0	0xe0200000	0xe0200fff		REGISTER	
	ps/_xadc_0	0x1800/100	0x1800/120		REGISTER	
	ps7_ddr_0	0x00100000	0x111111		RECISTER	
	ps7_ddrc_0	0xf800c000	0xf800cfff		REGISTER	
	ps7_bclinc_b	0xf8f02000	0xf8f02fff		REGISTER	
	axi dma 0	0x40400000	0x4040ffff	S AXLLITE	REGISTER	
	ps7 uart 1	0xe0001000	0xe0001fff		REGISTER	
	ps7_coresight_comp_0	0xf8800000	0xf88fffff		REGISTER	
	ps7_uart_0	0xe0000000	0xe0000fff		REGISTER	
	ps7_i2c_0	0xe0004000	0xe0004fff		REGISTER	
	ps7_scugic_0	0xf8f00100	0xf8f001ff		REGISTER	
	ps7_dev_cfg_0	0xf8007000	0xf80070ff		REGISTER	
	ps7_dma_ns	0xf8004000	0xf8004fff		REGISTER	
	ps7_sd_1	0xe0101000	0xe0101fff		REGISTER	
	ps7_sd_0	0xe0100000	0xe0100fff		REGISTER	
	ps7_gpv_0	0xf8900000	0xf89fffff		REGISTER	
	ps7_ram_1	0xtttt0000	0xtttttdtt		MEMORY	
	ps/_ram_0	0x00000000	0x0002ffff		MEMORY	
	IP blocks present in the des	ign				
	ps7_0_axi_periph axi_	interconnect	2.1			
	ps7_intc_dist_0 ps7_	intc_dist	1.00.a			
	axi_dma_0 axi_	dma	7.1 Regis	ters		
	ps7_gpio_0 ps7_	gpio	1.00.a			
	axis_data_fifo_0 axis	_data_fifo	2.0			
	ps7_scutimer_0 ps7_	_scutimer	1.00.a			
	ps/_sicr_U ps/_	SICF	1.00.a			
	psr_scuwar_v psr_	scawar	1.00.d			

2. To make things easy for us, we'll use the template for the hello world application and then modify it to test the AXI DMA. From the File menu, select New->Application Project.

ile E	Edit Navigate	- ·							
		Search	Project	Run	Xilinx	Window	Help		
N	lew				Alt	t+Shift+N >		Application Project	
O	pen File						~	SPM Project	
📮 Oj	pen Projects fro	m File Sys	tem				UN.	Board Support Package	
CI	lose					Ctrl+W		Project	rm_0 H

3. In the first page of the New Project wizard, choose a name for the application. I've chosen hello_world. Click "Next".

ect\ASPLOC\ps_pl_	com\ps_pl_com.sdk\hello_wc	rld	Browse			
tem: default 🗸						
.e			~			
sign_1_wrapper_h	w_platform_0		~ N <u>e</u> w			
rocessor: ps7_cortexa9_0						
○ C ○C++						
32-bit	~					
N/A	~					
: OCreate New	hello_world_bsp					
	ect\ASPLOC\ps_pl_ tem: default ~ ee esign_1_wrapper_hr 7_cortexa9_0 C C O C++ 32-bit N/A	ect\ASPLOC\ps_pl_com\ps_pl_com.sdk\hello_wo tem: default v e esign_1_wrapper_hw_platform_0 7_cortexa9_0 C O C++ 32-bit v N/A v C Create New bello world bop	ect\ASPLOC\ps_pl_com\ps_pl_com.sdk\hello_world tem: default v e e esign_1_wrapper_hw_platform_0 7_cortexa9_0 C O C++ 32-bit N/A v e Create New Pallo world hsp			

4. On the templates page, select "Hello World" template and click "Finish".



5. The SDK will generate a new application which you should find in the Project Explorer as in the image below.



6. The hello_world folder contains the Hello World software application, which we will modify to test our AXI DMA.

3, Modify the software application

We need to modify the hello world software application to test our DMA. The application source code we are using in this tutorial is one of the many valuable examples provided by Xilinx in the installation files. If you didn't know about those examples, I suggest you check it out every time you start playing with a new IP core.

- 1. From the Project Explorer, open the hello_world/src folder. Open the "helloworld.c" source file.
- 2. Replace all the code in this file with the code that you will find in the Vivado example project for AXI DMA.



3. Copy the contents of "xaxidma_example_sg_polll.c" into "helloworld.c".



- 4. Compile the project and launch it on the board. If everything is successful, the output message "Successfully ran AXI DMA SG Polling Example" will appear in the com port.
- 5. Note that by default the "hello world" project template initiate stdin/stdout to ps7_uart_0.

system.hdf system system.hdf system.hdf system.hdf system.hdf system.hdf	em.mss 🛛 🖻 helloworld.c Soard Support Packa	ige				🗖 🗖 🗄 Outline 🛛 🔁
Modify this BSP's Settin	ngs Re-generate BSP Source	es				An outline is not
Target Information		e ur				×
This Board Support Pac	Board Support Package	Settings				<u>^</u>
Hardware Specificatior Target Processo	Board Support Packag Control various settings of	e Settings your Board Support Package.				
Operating System	Quantinu					
Depart Current Department	~ Overview	Configuration for OC.	defense.			
воаго зорроп Раскаде	standalone	Configuration for US: Stan	dalone			
Name: stand	standalone ~ drivers	Name	Value	Default	Туре	Description
Name: stand Version: 6.8	standalone ~ drivers ps7_cortexa9_0	Name hypervisor_guest	Value false	Default false	Type boolean	Description Enable hypervisor guest su
Name: stand Version: 6.8 Description: Stanc	standalone ~ drivers ps7_cortexa9_0	Name hypervisor_guest lockstep_mode_debug	Value false false	Default false false	Type boolean boolean	Description Enable hypervisor guest st Enable debug logic in nor
Name: stand Version: 6.8 Description: Stanc as we	standalone ~ drivers ps7_cortexa9_0	Name hypervisor_guest lockstep_mode_debug sleep_timer	Value false false none	Default false false none	Type boolean boolean peripheral	Description Enable hypervisor guest su Enable debug logic in nor This parameter is used to
Name: stand Version: 6.8 Description: Stand as we Documentation: <u>stand</u>	standalone ~ drivers ps7_cortexa9_0	Name hypervisor_guest lockstep_mode_debug sleep_timer stdin	Value false false none ps7_uart_0	Default false false none none	Type boolean boolean peripheral peripheral	Description Enable hypervisor guest si Enable debug logic in nor This parameter is used to stdin peripheral
Name: stand Version: 6.8 Description: Stanc as we Documentation: <u>stand</u>	standalone ~ drivers ps7_cortexa9_0	Name hypervisor_guest lockstep_mode_debug sleep_timer stdin stdout	Value false false none ps7_uart_0 ps7_uart_0	Default false false none none none	Type boolean boolean peripheral peripheral peripheral	Description Enable hypervisor guest su Enable debug logic in nor This parameter is used to stdin peripheral
Name: stand Version: 6.8 Description: Stanc as we Documentation: stand	standalone ~ drivers ps7_cortexa9_0	Name hypervisor_guest lockstep_mode_debug sleep_timer stdin stdout ttc_select_cntr	Value false false none ps7_uart_0 ps7_uart_0 2	Default false false none none 2	Type boolean boolean peripheral peripheral peripheral enum	Description Enable hypervisor guest st Enable debug logic in nor This parameter is used to stdin peripheral stdout peripheral Selects the counter to be
Name: stand Version: 6.8 Description: Stanc as we Documentation: stand Peripheral Drivers Drivers present in the B	standalone ~ drivers ps7_cortexa9_0	Name hypervisor_guest lockstep_mode_debug sleep_timer stdin stdout ttc_select_cntr zynqmp_fsbl_bsp	Value false false none ps7_uart_0 ps7_uart_0 2 false	Default false false none none porre 2 false	Type boolean boolean peripheral peripheral enum boolean	Description Enable hypervisor guest st Enable debug logic in nor This parameter is used to stdin peripheral stdout peripheral Selects the counter to be Disable or Enable Optimi;
Name: stand Version: 6.8 Description: Stand as we Documentation: stand Peripheral Drivers Drivers present in the B axi_dma_0	standalone ~ drivers ps7_cortexa9_0	Name hypervisor_guest lockstep_mode_debug <u>sleep_timer</u> stdin <u>stdout</u> ttc_select_cntr zynqmp_fsbl_bsp > microblaze_exceptions	Value false false ps7_uart_0 ps7_uart_0 2 false false	Default false false none none 2 false false false	Type boolean peripheral peripheral peripheral enum boolean boolean	Description Enable hypervisor guest si Enable debug logic in nor This parameter is used to stdin peripheral stdout peripheral Selects the counter to be i Disable or Enable Optimii Enable MicroBlaze Except

 However, if we go back to Vivado and open up the Zynq7000 configuration view for peripheral I/O, we can see that in our system, by default, UART 0 is used for EMIO but UART 1 is used for communication port UART 1.

		_															
UAKT U	JARTO		UARTO		UARTO		UARTO		UARTO		UART0		UART0		UART0		EMIO
VART 1		HADTA		UADTA		HADTA		UADTA		HADTA		UADTA		UADTA		UADTA	ENIO
0		UARTI		UARTI		UARTI		UARTI						UARTI		UARTI	EMIO

7. Modify the bsp file so that stdin/stdout are pointing to UART 1 instead of UART 0.

Board Support Package	Settings				×		
Board Support Package	e Settings						
Control various settings of	your Board Support Package.						
~ Overview							
standalone	Configuration for OS: stan	dalone					
 drivers 	Name	Value	Default	Туре	Description		
ps7_cortexa9_0 hyp loci slee	hypervisor_guest	false	false	boolean	Enable hypervisor guest si		
	lockstep_mode_debug	false	false	boolean	Enable debug logic in nor		
	sleep_timer	none	none	peripheral	This parameter is used to		
	stdin	ps7_uart_1	none	peripheral	stdin peripheral		
	stdout	ps7_uart_1	none	peripheral	stdout peripheral		
	ttc_select_cntr	2	2	enum	Selects the counter to be		
	zynqmp_fsbl_bsp	false	false	boolean	Disable or Enable Optimiz		
	> microblaze_exceptions	false	false	boolean	Enable MicroBlaze Except		
	> enable_sw_intrusive_profili	false	false	boolean	Enable S/W Intrusive Profi		

8. Ater the modification you should see the output correctly in the COM port.

4	Libraries	
	No libraries in the Board Support Package	
C	Overview Source	
	🖞 Problems 🧟 Tasks 🖳 Console 🔲 Properties 🖳 SDK Terminal 🕴 🛛 🕂 🐥 🖓 🕫	3
C	Connected to: Serial (COM4, 115200, 0, 8)	
	Entering main() Successfully ran AXI DMA SG Polling Example Exiting main()	•

9. If you see this message, then everything should be set correctly in the whole HW/SW platform.