fidus
innovate•design•deliver

A Hardware Designer's Informal Guide to Xilinx® Zynq® UltraScale+™

By Fidus Systems

For Anyone Interested in Learning More

Version: 1.0
2020-04-06

XILINX® | ALLIANCE PROGRAM PREMIER MEMBER

Revision History

Revision	Author	Release Date	Description of Change
0.1	ST	2020-03-27	Initial Draft (incomplete)
0.2	ST	2020-04-01	Initial Draft (complete, ready for review)
1.0	ST	2020-04-06	Updated following review. Released.

Table of Contents

Disclaimer

This article is provided for the User's enjoyment. All information contained herein is believed to be correct. It is possible that some information is incorrect, misleading, or out-dated. We recommend that the information contained herein be used as a guide only, and that the User reviews all claims, concerns, or design decisions, with Xilinx, Xilinx's official information, and/or Xilinx's representatives (i.e. Avnet). Fidus cannot be held responsible for losses or damages associated with the use or misuse of the information within article. Use of this article and the information contained is at User's own risk.

The views and opinions expressed in this article are those of the author(s) and do not necessarily reflect the official policy or position of Fidus Systems.

Fidus Systems assumes no responsibility or liability for any errors or omissions in the content of this article. The information contained in this article is provided on an "as is" basis with no guarantees of completeness, accuracy, usefulness or timeliness.

Fidus Systems is a Xilinx Premier Design Services Member. Fidus does not represent or pretend to represent Xilinx.

1 Introduction

After delivering more than twenty (20) Zynq® UltraScale+™ (Zynq US+) designs last year, Fidus can truly say that they are expert implementers of the latest Multi-Processor System On-a-Chip (MPSoC; pronounced em-pee-sok) technology from Xilinx®. These designs spanned multiple applications and markets. This whitepaper is targeted at people who are generally familiar with the Zynq US+ and are either, a) Considering Zynq US+ for their next design, b) Want to gain insight into some common Zynq US+ applications, and/or c) Frankly, potential Customers who are considering Fidus for their next custom Zynq US+ design. This paper can also be used as a guide to the most 'design critical' Xilinx Zynq US+ reference documents.

I'm Scott, a hardware designer by trade, so this document focuses mostly on the actual hardware aspects related to Zynq US+. And let's face it, a paper on Zynq US+ FPGA and Software development would be several thousand pages long as the chip can do pretty much anything.

Hope you enjoy and find this article of value. Feel free to reach out via the contact information provided at the end.

2 The Tips

This section describes some of the most valuable knowledge and interesting resources I've come across when designing with the Zynq US+. It is my hope that these little tidbits will help jumpstart your design, or (better yet), give you the full confidence that Fidus is the right partner to get a great Zynq US+ design to market fast!

2.1 Selecting your Device

Following on the success of the 7-series Zynq device, Zynq US+ is the latest MPSoC from Xilinx. The Zynq US+ is a heterogenous device consisting of two main elements: A Processing System (PS) and a Programmable Logic (PL) system. The PS contains "hard" elements (meaning elements that cannot be reconfigured like they can be in the PL section) such as ARM Processors and related support architecture, Memory Controllers, a whole slew of hardware controllers (e.g. SecureDigital, I2C, SPI, USB, PCIe, etc.), and a variety of other fixed functions. The PL system is the traditional programmable fabric (i.e. FPGA with high-speed I/O and gigabit-rate transceivers) that Xilinx made their name delivering. Provisioning your function correctly by assigning the sub-blocks into either PS or PL is a critical architectural element of designing with Zynq US+.

Warning!

With a chip as flexible and powerful as the Zynq US+ it's tempting to haphazardly dive in and just say we'll figure out the provisioning later, but don't do this, sure you may get it to work in the end, but spending the time upfront will pay off in terms of development effort, overall elegance, and hence time-to-market.

So, how do I pick my target device?

The Zynq US+ comes in three different family members: CG, EV, and EG. The following table from Xilinx clearly identifies the functional differences. You will note that the table is entitled "Processing System Features". You can extrapolate from this that the differences between the CG, EV, and EG, are in the PS (or the hard blocks) and not necessarily within the PL.

Processing System Features

	CG Devices	EG Devices	EV Devices
Application Processing Unit	Dual-core ARM® Cortex™-A53	Quad-core ARM Cortex-A53	Quad-core ARM Cortex-A53
Real-Time Processing Unit	Dual-core ARM Cortex-R5	Dual-core ARM Cortex-R5	Dual-core ARM Cortex-R5
Graphics Processing Unit	-	ARM Mali™-400 MP2	ARM Mali-400 MP2
Video Codec Unit	-	-	Supports H.264/H.265
Dynamic Memory Interface	DDR4, LPDDR4, DDR3, DDR3L, LPDDR3		
High-Speed Peripherals	PCIe® Gen2, USB3.0, SATA 3.1, DisplayPort, Gigabit Ethernet		

Ref: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html#productTable

Although not strictly in line with Xilinx marketing lingo, we boil down the differences as follows:

CG. Targeted at applications that are cost sensitive and do not require crazy amounts of processing power or graphics.

EV. Targeted at cost sensitive video or broadcast applications that are going to leverage the hard VCU (Video Codec Unit) to deliver H.264/H.265 compression/decompression functionality.

EG. Targeted at performance applications that have a lot of high-speed I/O requirements (like 100GE, storage applications, other streaming data, processor off-load), and are then required to move large amounts of data through the FPGA.

Xilinx also summarizes some target applications in the following:
https://www.xilinx.com/support/documentation/selection-guides/zynq-ultrascale-plus-product-selection-guide.pdf#page=2

As common place in the FPGA families, each Zynq US+ family (CG, EV, EG) also contains multiple members. The members of a family all have identical PS's but are differentiated by their traditional "FPGA characteristics" (e.g. size of their fabric size, I/O count, transceiver count and speed, package type, and speed grade).

So, you now probably have an idea of whether you want to use CG, EV, or EG, however, how about identifying the other "FPGA Characteristics" that will drive your part number selection? Fabric-based IO and transceiver count is relatively straightforward – tabulate all of your IO, make rough bank assignments, and transceivers requirements – and make note of how many FPGA banks you will require and which IO should leverage MIO (see subsequent sections). Speed grade is generally driven by IO speed requirement and/or by internal operating speed; whichever is the driving factor. This is a good time to have a detailed discussion with the FPGA architect, and while you're talking to her, also ask her to estimate the logic resource count and functions (e.g. PLLs, BRAMs, etc.) required to support the functionality, as this will enable you both to narrow down fabric size. Package type will be driven first by IO count and then by speed requirements (i.e. only certain package types may support certain IO speeds), followed by any limitations in circuit-board technology (e.g. are you layer constrained, are blind and buried vias acceptable, etc). This is a good time to have a discussion with your PCB layout designer. Having a great understanding of all these elements will allow you to identify the family member to target.

Warning!

Cost drives all. I intentionally left cost out of this discussion because it can't be described or quantified in any more insightful way other than: Larger (fabric, pin count), faster devices cost more; there, done! My assumption is that your bill of materials target cost can support the device you have architected for. If not, you will have to consider re-architecting or reducing functionality.

The following references from Xilinx (informative, but too cumbersome to snapshot below) highlight the family members and upon careful inspection, bolsters the generic statements we made above regarding the target applications for CG, EV, and EG.

CG: https://www.xilinx.com/support/documentation/selection-guides/zynq-ultrascale-plus-product-selection-guide.pdf#CG.

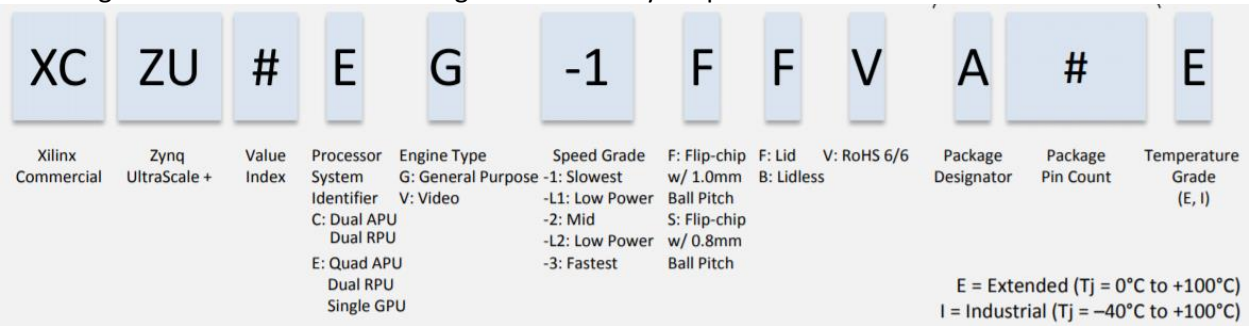
EV: https://www.xilinx.com/support/documentation/selection-guides/zynq-ultrascale-plus-product-selection-guide.pdf#EV.

EG: https://www.xilinx.com/support/documentation/selection-guides/zynq-ultrascale-plus-product-selection-guide.pdf#EG

Warning!

Xilinx only produces certain combinations of features per device (aka valid part numbers), including Temperature and Speed Grades, so use page 9-11 of the Xilinx selection guide to select carefully and ensure that the device you identified containing your desired combination of features actually exists!

Use the guidance within the selection guide and build your part number:



Now that you have built a part number, reach out to Xilinx or Avnet to confirm your part number is valid (and I'm sure you'll be interested in costing information as well). Having done all of the above research you'll be able to get the most out of your conversation with Xilinx/Avnet. The people there are good and can make recommendations, but you're the expert in your application, so you'll have to convey the application and needs of the design to them so they can best guide you.

2.2 MIO Structure and Function Assignment

First off, what is MIO? MIO stands for Multiplexed IO and refers to the IO pins the are connected to the hard Processing System (PS) of the Zynq US+. The 'Multiplexed' adjective refers to the fact that the Zynq US+ PS is provisioned with a multitude of hard controllers, that, through the tools-based configuration of internal multiplexers, can be connected to various PS-based IO pins. This is a powerful capability but understanding how it works is critical.

As we explored in the previous section, there are plenty of valid part numbers of Zynq US+ devices. The good news is: All of the MIO structures are the same! While it's true, that they have different pin numbers based on their package type, in a stroke of pure genius Xilinx made the arrangement and functionality the same across all part numbers. Let's explore this.

Regardless of part number, the MIOs are always arranged into three (3) banks: Bank 500, 501, and 502. We won't discuss Bank 503 in detail here, although technically it is part of the PS system (PS Config), but unlike MIO it has fixed functionality, however, note that like MIO, it too is the same across all part numbers. Each of Bank 500, 501, and 502, contain twenty-six (26) MIO pins. They are assigned as follows:

Bank 500: PS_MIO[25:0]
Bank 501: PS_MIO[51:26]
Bank 502: PS_MIO[77:52]

So, you've got 78 MIO pins, however, to make the best use of them, you have to understand the assignment restrictions. This is the complicated part – not because it's hard or daunting, but because you need to take the time to understand how to read Xilinx's somewhat intuitive, yet large MIO assignment chart.

Warning!

Do not make the mistake of thinking that any MIO pin can be assigned any function; this is not true.

Here’s where Xilinx’s 1200 page document, UG1085 Zynq US+ Technical Reference Manual becomes crucial, specifically the MIO chart in Chapter 28.

Here’s a small clip of this table that actually spans two-pages:

MIO Table at a Glance

For pin planning, see Table 28-1. MIO signals are also listed in each controller chapter along with their function, direction, and presence in EMIO.

Table 28-1: **MIO Interfaces**

Interface	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
gem0																										
gem1																										
gem2																										
gem3																										
gem_tsu																										
qspi[2]	4	1	2	3	0	5	12	6	8	9	10	11	7													
nand										2	13	14	4	1	3	0	5	6	7	8	9	10	16	11	12	15
pcie																										
usb0																										
usb1																										

Interface	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
gem0	0	1	2	3	4	5	6	7	8	9	10	11														
gem1													0	1	2	3	4	5	6	7	8	9	10	11		
gem2																										
gem3																										
gem_tsu	0																								0	0
qspi[2]																										
nand	2	13	14				4																			
pcie				0	0	0		0	0	0	0	0														
usb0																										
usb1																										

Interface	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77
gem0																										
gem1																										
gem2	0	1	2	3	4	5	6	7	8	9	10	11														
gem3													0	1	2	3	4	5	6	7	8	9	10	11		
gem_tsu																										
qspi[2]																										
nand																										
pcie																										
usb0	0	1	6	3	4	5	2	7	8	9	10	11														
usb1													0	1	6	3	4	5	2	7	8	9	10	11		

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=789

This chart is interpreted as follows:

a. Across the top are the MIO pin assignments. These are not the pin numbers relating to the package itself, these are the MIO pin identifiers. Note: When you create the schematic symbol, Xilinx names the pins MIO0, MIO1, MIO2, etc., so the pin names align with the table ordinals.
b. Down the side are the hard controller interface options. For example, “gem0” refers to the first of four (4) available Gigabit Ethernet MACs, “qspi” refers to the QSPI flash controller, “USB0” refers to the first to two (2) USB ULPI/PIPE3 Controllers, and so on.

How do we use this information to assign MIO pins to our design? We have to go back to our Requirements, understand which of the PS hard controllers we are going to leverage, and then assign them appropriately to MIO pins, without overlap! When I say, “without overlap” here’s what I mean: See how in the chart gem2 and usb0 can each only be assigned to PS MIO[63:52]? This means that I have to pick one function or the other to use those pins – obviously they both can’t be assigned to those pins at the same time. Then, to meet the rest of your requirements, you go through and assign hard controllers without overlap. If you don’t fill up your MIO with hard controller needs, unused pins can generally be leveraged as general purpose IO, but keep the function simple, because remember at this point software will have to interact directly with that GPIO pin. Once my MIO assignment is complete, I always, always, always enter the assignment into the Xilinx tools to ensure I don’t have any conflicts (remember, the ‘M’ in ‘MIO’ refers to a series of multiplexers that connect the hard controllers to the IO pin itself, so there is a chance that your assignment may leave the tools without a way to connect the controller through the layers of multiplexers to the IO pin).

Warning
Always verify your MIO pinout using the Xilinx tools. Not all combinations will work!

The next level of complexity, and perhaps the trickiest to ensure you get right, is understanding what the numbers within the function mean and why more configurable interfaces sometimes have these pins out of order.

First, we will look at the qspi controller assignment, as it is an out of order example, then we will look at the gem0 controller as it is more straightforward.

Starting with qspi, here's a close-in clip from the main MIO assignment table:

Table 28-1: **MIO Interfaces**

Interface	0	1	2	3	4	5	6	7	8	9	10	11	12	1
gem0														
gem1														
gem2														
gem3														
gem_tsu														
qspi(2)	4	1	2	3	0	5	12	6	8	9	10	11	7	

As per the table, Xilinx is instructing you to connect MIO[0] to "4", MIO[1] to "1", MIO[2]to "2", MIO[3] to "3", MIO[4] to "0", and so on. What do these seemingly random assignments mean? To understand you have to go back to that huge Technical Reference Manual and flip to the Chapter that describes the QSPI Controller (Chapter 24), and you will find the table below that holds the key to decoding the main MIO first table. Let's decode it.

Quad-SPI Flash Memory Interface			MIO Pin				Controller Default Input Value
Data Mode			Quad-SPI0 (lower)	Quad-SPI1 (upper)	I/O	Name	
1-Bit Data	2-Bit Data	4-Bit Data					
Chip select			5	7	O	SS_b	~
Serial clock			0	12	O	SCLK	~
Optional feedback clock			6		O	LPBK_CLK	~
MOSI	I/O 0	I/O 0	4	8	I/O	IO[0]	0
MISO	I/O 1	I/O 1	1	9	I/O	IO[1]	0
Write protect		I/O 2	2	10	I/O	IO[2]	0
Hold		I/O 3	3	11	I/O	IO[3]	0

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=698

QSPI is typically implemented in 4-bit data mode, so we'll discuss that implementation, which means we can ignore Quad-SPI1 (upper) and focus on Quad-SPI0 (lower). QSPI is relatively simple and consists of: Chip Select, Clock, and data I/O[3:0].

From the table, we uncover the following mapping of ordinals:
Chip Select = 5, Clock = 0, I/O 0 = 4, I/O 1 = 1, I/O 2 = 2, I/O 3 = 3

Now remember back to the master MIO table (the big one), and we now see that the defined ordinals from the QSPI Controller Chapter map to the ordinals in the MIO table.

Table 28-1: **MIO Interfaces**

Interface	0	1	2	3	4	5	6	7	8	9	10	11	12	1
gem0														
gem1														
gem2														
gem3														
gem_tsu														
qspi(2)	4	1	2	3	0	5	12	6	8	9	10	11	7	

Which allows us to complete the physical assignment mapping:

Chip Select = 5 maps to MIO[5]
Clock = 0 maps to MIO[4]
I/O 0 = 4 maps to MIO[0]
I/O 1 = 1 maps to MIO[1]
I/O 2 = 2 maps to MIO[2]
I/O 3 = 3 maps to MIO[3]

Let's do this two-stage mapping again, but this time with the more straightforward gem0 controller:

Start with the main MIO table:

Table 28-1: **MIO Interfaces**

Interface	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
gem0																											0	1	2	3	4	5	6	7	8	9	10	11	
gem1																																							0

Then look for the mapping on the IO table within Chapter 34, GEM Ethernet:

Table 34-13: **Ethernet RGMII Interface Signals via MIO Pins**

Controller Signal		MIO Pins					
Signal Description	Default Controller Input Value	GEM 0	GEM 1	GEM 2	GEM 3	Name	I/O
Tx clock to PHY	~	26	38	52	64	RGMII_TX_CLK	O
Tx control to PHY	~	31	43	57	69	RGMII_TX_CTL	O
Tx data 0 to PHY	~	27	39	53	65	RGMII_TXD[0]	O
Tx data 1 to PHY	~	28	40	54	66	RGMII_TXD[1]	O
Tx data 2 to PHY	~	29	41	55	67	RGMII_TXD[2]	O
Tx data 3 to PHY	~	30	42	56	68	RGMII_TXD[3]	O
Rx clock from PHY	0	32	44	58	70	RGMII_RX_CLK	I
Rx control from PHY	0	37	49	63	75	RGMII_RX_CTL	I
Rx data 0 from PHY	0	33	45	59	71	RGMII_RXD[0]	I
Rx data 1 from PHY	0	34	46	60	72	RGMII_RXD[1]	I
Rx data 2 from PHY	0	35	47	61	73	RGMII_RXD[2]	I
Rx data 3 from PHY	0	36	48	62	74	RGMII_RXD[3]	I
GEM TSU clock options	~	26,50,51	26,50,51	26,50,51	26,50,51	GEM_TSU_CLK	I

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=1032

Note that unlike the qspi0 table, the gem0 table does not just show an ordinal that needs mapping, it shows the actual MIO pin identifier. So, we can more easily come to our conclusion:

TX CLK = 26 maps to MIO[26]
TX CTL = 31 maps to MIO[31]
TX DATA0 = 27 maps to MIO[27]
TX DATA1 = 28 maps to MIO[28]
TX DATA2 = 29 maps to MIO[29]
TX DATA3 = 30 maps to MIO[30]
RX CLK = 32 maps to MIO[32]
RX CTL = 37 maps to MIO[37]
RX DATA0 = 33 maps to MIO[33]
RX DATA1 = 34 maps to MIO[34]
RX DATA2 = 35 maps to MIO[35]
RX DATA3 = 36 maps to MIO[36]

It is always important to look at the MIO table, and then understand the actual MIO mapping that is well defined within the Chapter related to that desired controller. As we have seen, sometimes we need to map ordinals back to the original table and sometimes the MIO pins map directly. Once you have done this a couple times, it is simple enough – just don't skimp on the details – understand what the detail within the Controller Chapter is telling you.

Warning
Always refer to both the MIO table and the actual Chapter that defines the use and IO of that hard controller!

2.3 PS MIO Bank Voltage

Just like their FPGAs, Xilinx has made the MIO bank voltage configurable. As discussed, there are three (3) MIO banks: Bank 500, 501, and 502. The voltage levels at which these banks operate are defined by the voltage applied to the VCCO_PSIO[0:2] pins. More succinctly put:

The MIO within Bank 500 will operate at the voltage level provided to VCCO_PSIO0
The MIO within Bank 500 will operate at the voltage level provided to VCCO_PSIO1
The MIO within Bank 500 will operate at the voltage level provided to VCCO_PSIO2

Now, unlike the FPGAs, there are only three (3) voltage options supported: 1.8V, 2.5V, and 3.3V.
Ref: https://www.xilinx.com/support/documentation/data_sheets/ds925-zynq-ultrascale-plus.pdf#page=18

As usual, the voltage you select for each bank is determined by the devices that are interfacing to that bank. Note that unlike FPGAs, the MIO within that bank can only operate with the exact voltage applied to that bank. So, now you not only have to be aware of the MIO hard controller assignment, but also that all functions assigned to that bank must operate at the same voltage.

2.4 Extended Multiplexed IO (EMIO)

Xilinx makes FPGAs, FPGAs can do almost anything, so why not add crazy amounts of flexibility to the MPSoC as well, right? That's exactly what Xilinx did with their EMIO capability. As I mentioned before, the MIO consists of a total of 78 pins, and there are a plethora of hard controllers with the PS. In fact, there are more hard controllers than there are MIO pins to break them out. EMIO is a solution to this problem. EMIO technology enables you to bridge PS-based hard controllers to the PL (i.e. typically the FPGA pins). If you are going to do this, it is critical that you run your planned connectivity through the Xilinx tools, as there are a lot of considerations and limitations (operating speed, protocol types, etc.) associated with EMIO. The following Xilinx table summarizes some of these limitations.

Interface	MIO Access	EMIO Access	Notes
GEM{0:3}	RGMII	GMII	MIO: 4-bit RGMII v2.0, external PHY, 250 MHz data rate. EMIO: 8-bit GMII, RGMII v2.0 (HSTL), RGMII v1.3, MII, SGMII, 1000BASE-SX, and 1000BASE-LX in PL, 125 MHz data rate.
SDIO{0, 1}	Yes	Yes	The SDIO interface performance is reduced when using the EMIO interface.
USB{0, 1}	USB 2.0 to external ULPI PHY.	No	The USB 3.0 interface is routed to a GTR channel
I2C{0, 1}	Yes	Yes	
SPI{0, 1}	Yes	Yes	The SPI interface performance is reduced when using the EMIO interface.
UART{0, 1}	Yes (RX, TX)	Yes (RX, TX, modem signals).	
CAN{0, 1}	Yes	Yes	External PHY.
GPIO Banks {0:2}	Yes (up to 78)	No	
GPIO Banks {3:5}	No	Yes (up to 96)	Input, output, and 3-state control.
Quad-SPI	Yes	No	
NAND	Yes	No	
LPD_SWDT, FPD_SWDT	Yes	Yes	Reset and output pulse.
CSU_SWDT	No	No	
TPIU Trace	Up to 16 bits	Up to 32 bits	

Take special note of the "No" in the EMIO Access column.

Warning

Always verify your EMIO pinout using the Xilinx tools. Not all combinations will work! And there may be limitations that your application cannot tolerate.

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=48

2.5 Bank 503

Bank 503 houses the command and control pins of for the Zynq US+. Like the MIO, Bank 503 architecture and functionality is identical (excluding pin numbers of course) on every Zynq device. Bank 503 is home to the following pins.

VCCO_PSIO3: The bank voltage that will determine the signaling level of the Bank 503 pins. Like the MIO VCCO_PSIO pins, this voltage can be 1.8, 2.5, or 3.3V.

PS_DONE: Indicates the PL configuration is complete.

PS_ERROR_STATUS: Asserted for accidental loss of power, a hardware error, or an exception in the PMU.
PS_POR_B: Power On Reset. This is a hard reset that re-starts both the Low Power Domain (LPD) and the Full Power Domain (FPD).

PS_REF_CLK: PS System's Reference Clock

PS_SRST_B: System Reset. This is a little less hard that PS_POR_B as there are several registers that are unaffected (see reference below). This is usually connected to the debugging tool (e.g. JTAG connector).

PS_JTAG_TCK/TDI/TDO/TMS: JTAG Connection.

PS_MODE[3:0]: The "Mode Pins" are sampled on the rising edge of PS_POR_B and determine the Primary Boot Method (e.g. QSPI, SD, NAND, etc.).

PS_ERROR_OUT: Asserted for accidental loss of power, a hardware error, or an exception in the PMU.

PS_INIT_B: Indicates that the PL is initialized after a power-on reset.

PS_PROG_B: PL Configuration Reset. Similar to the usual FPGA configuration PROG_B pin.

2.6 Other Dedicated Pins

The following other dedicated pins are worth a quick discussion.

B0_PUDC_B: Enables weak pull-ups on all SelectIO pins during configuration. Referenced to VCCAUX. These can be used if the Zynq US+ is interfacing to other 'less-smart' components that benefit from a defined state. I don't typically use this function as I prefer to externally control things, so I don't have to worry about pull-up strength and timing considerations.

DXP/N: Internal temperature sensing diode pins. P is the Anode; N is the Cathode. Although SYSMON gives you similar capabilities, SYSMON may not always be available, so you may connect this to your thermal monitoring solution. For example, perhaps your thermal monitoring solution observes the temperature of the Zynq US+ to either adjust fan speed or to completely shut down the system if the Zynq US+ temperature exceeds a specified threshold. In this case, you would typically shutdown the Zynq US+ (SYSMON no longer available), and then monitor the DXP/DXN until the temperature returns to acceptable level before re-enabling the system. It's comforting having a completely hardware-based thermal strategy.

VCC_PSADC/GND_PSADC: PS System Monitor supply voltage and ground reference.

POR_OVERRIDE: Sets the PL Power-On Delay Time. Logic 0 is the recommended, default setting. Xilinx explains:

"Reduces TPOR time (from power up to INIT_B rise) as specified in data sheet. Connect directly to VCCINT for a shorter TPOR time if required and if supported by the power-up timing of the configuration data source. Connect to GND for standard longer POR delay.

CAUTION! *Do not allow this pin to float before and during configuration. This pin must be tied to VCCINT or GND."*

Ref: https://www.xilinx.com/support/documentation/user_guides/ug570-ultrascale-configuration.pdf#page=30

PS_PADI/PADO: Crystal connection for internal Real Time Clock.

VCCADC, GNDADC, VP/N: VCCADC is the voltage used by the internal analog PL System Monitor function. It is referenced to GNDADC. VP/N are differential inputs, used like inputs to an ADC, connected to the PL System Monitor function.

VCC_PSBATT: PS RTC and Battery Backed-up RAM (BBRAM) Battery Voltage. Typical range: 1.2 to 1.5V. Note that if you require Secure Configuration, the key is stored in BBRAM.

Ref: https://www.xilinx.com/support/documentation/application_notes/xapp1323-zynq-usp-tamper-resistant-designs.pdf

VREFP/N: Externally applied PL SYSMON Reference voltage. VREFP receives a 1.25V (typical) supply and VREFN is the ground reference (this should be filtered from GNDADC). As these are high-impedance inputs, the 1.25V is typically generated from a simple voltage reference. Alternatively, if accuracy is less important than cost or real-estate, the internal reference can be utilized, in which case both VREFP and VREFN are connected to GNDADC.

Ref: https://www.xilinx.com/support/documentation/user_guides/ug580-ultrascale-sysmon.pdf#page=95

Warning!
Be diligent to ensure that you are referencing the SYSMON analog voltage supplies, references, and signals, to the correct ground reference, using the appropriate filtering.

2.7 PS and PL System Monitor

The Zynq US+ device contains a PS and PL System Monitor (referred to as PS and PL SYSMON). The SYSMON can monitor and report certain internal voltage levels and temperatures, in addition, the internal ADCs can be pinned out to measure external analog signals. There are differences between PS and PL SYSMON. Refer to the attached Xilinx table:

Comparison of PS SYSMON and PL SYSMON

The notable differences between the PS SYSMON and the PL SYSMON are the programming bus interfaces, sampling rates, and analog input signal sources. The differences are listed in Table 9-1.

Table 9-1: **PS SYSMON and PL SYSMON Comparison**

Function	PS SYSMON	PL SYSMON
Sampling frequency	1 M samples per second.	200K samples per second.
Voltage reference	Internal.	Internal or external (VREFP, VREFN).
Programming interfaces	APB on AXI interconnect. Includes DAP controller via JTAG.	APB/AXI interconnect. DRP (PL configuration required). I2C/PMBus. PL JTAG controller.
Power domain	LPD.	PLPD.
Temperature sensors with OT	Temp_LPD near the RPU MPCore. Temp_FPD near the APU MPCore.	Temp_PL near the PL SYSMON unit.
On-chip supply sensors	Three PS internal voltage nodes. Three I/O voltage nodes.	Three PS internal voltage nodes. Three PL internal voltage nodes. Four PL internal VUSER nodes.
PL external sensor channels	None.	16 signal pairs; VAUXP, and VAUXN(2). One set of dedicated pins, VP and VN(2).
PL user inputs	None.	Four, full featured.
Event driven trigger	AMS.PS_SYSMON_CONTROL_STATUS	CONVST start signal input.(1)
EOS, EOC	AMS.ISR_1 [eos], [eoc] interrupts.	EOS, EOC signals to PL fabric.(1)
Reset (see Reset Sources)	POR, write to VP_VN register, AMS.PS_SYSMON_CONTROL_STATUS.	RESET pin, write to VP_VN register.

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=187

The PS and PL SYSMON can be very useful in situations where you are assessing the robustness of your design or by avoiding the need to add a simple external ADC.

2.8 PS Transceivers (GTRs)

The PS supports a total of four (4) gigabit rate transceivers, called GTR[3:0]. The GTRs operate at rates up to 6 Gbits/s. As the name suggests, they support four (4) transmitters and four (4) receivers. Typical applications for these GTRs include: PCIe Gen2, SATA3.1, DisplayPort, USB3.0, and serialized Gigabit Ethernet (SGMII, 1000BaseSX/LX). There are also the associated MGT GTR Reference Clocks, which will be discussed later.

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=792

When assigning the GTRs be very careful, as complex transceivers have restrictions. Once again, I'll recommend verifying your assignment in the Xilinx tools. Also read the Xilinx documentation carefully. Here's an often overlooked, critical table from the Xilinx documentation:

Interconnect Matrix

The interconnect matrix (ICM) implements connectivity between the media access controllers (MACs) and the physical coding sublayer (PCS). The ICM is automatically programmed by the *Processing System Configuration Wizard* (PCW). Table 29-1 shows the connectivity implemented by the ICM between PS-GTR transceivers and the available MACs.

Table 29-1: **Interconnect Matrix**

Controller	PHY Lane 0	PHY Lane 1	PHY Lane 2	PHY Lane 3
PCIe v2.0	PCIe.0	PCIe.1	PCIe.2	PCIe.3
SATA	SATA.0	SATA.1	SATA.0	SATA.1
USB0 3.0	USB0	USB0	USB0	
USB1 3.0				USB1
DisplayPort	DP.1	DP.0	DP.1	DP.0
GEM0(1)	GEM0			
GEM1(1)		GEM1		
GEM2(1)			GEM2	
GEM3(1)				GEM3

Notes:
1. The GEM Ethernet interface to the GTRs includes SGMII, 1000BASE-SX, and 1000BASE-LX protocols.

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=796

This table tells you that GTR assignment is not as flexible as one might initially believe. My key interpretations/observations of this table are:

1. You can only assign one function per PHY Lane (obvious)
2. PHY Lane[0:3] corresponds to pins MGT[0:3]
3. If you are implementing a single lane PCIe interface, you must assign it to PHY Lane 0/PCIe.0
4. If you are implementing a single lane SATA interface, you must assign it to either PHY Lane 0 or PHY Lane 2
5. If you are implementing a DisplayPort interface, note that DP Lane 0 is on PHY Lane 1 or 3 and DP Lane 1 is one PHY Lane 0 or 2. Note: I've never tried implementing DP Lane 0 on PHY Lane 1 and DP Lane 1 on PHY Lane 2, I've always used either PHY Lane[1:0] or PHY Lane[3:2].

As with FPGA transceivers, the MGT GTR Reference Clocks are also critical. Xilinx has generously provided four (4) MGT GTR Reference Clock inputs. The protocol you are implementing determines which reference clock frequency you need to provide.

The following table defines the Requirements:

Table 29-2: **Reference Clock per Protocol**

Protocol	Reference Clock Frequency (MHz)
PCIe v2.0 (multi-lane) Only the common clock architecture is supported.	100.0 MHz
SATA (multi-core)	125.0 MHz, 150.0 MHz
USB 3.0	26.0 MHz, 52.0 MHz, 100.0 MHz
DisplayPort (harmonic of 27.0 MHz)	27.0 MHz, 108.0 MHz, 135.0 MHz
GEM SGMII, 1000BASE-SX, or 1000BASE-LX	125.0 MHz

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=799

I don't think it's strictly required as there are interconnects that allow multiple transceivers to leverage a single reference clock, but I generally assign the reference clock to the same ordinal as the protocol it is being provided for. Because, why not? For example: If USB were on PHY Lane 1, I would provide either a 26, 52, or 100MHz clock

to MGT REF CLK 1. In designs with both USB and PCIe (Host application), I can provide a single 100MHz reference clock (above table confirms that both support 100MHz) and allow the transceivers to share it.

Warning!
A PCIe End Point application would require the 100MHz be driven by the system common PCIe clock, so I would then provide a standalone, dedicated 100MHz reference in support of the USB.

2.9 PCIe

The Zynq US+ can support PCIe Host or End Point functionality either via a PS hard block or soft-IP on the PL. The PS hard block is limited to PCIe Gen2 x1 or x4 operation (there are only four (4) PS GTRs after all), whereas the PL supports soft-IP up to PCIe Gen3 x1, x4, x8, or x16. PCIe Gen4 is not fully supported due to protocol limitations (this shortcoming was addressed in the Virtex® US+ devices that contain HBM and obviously in Xilinx products moving forward, see links below).

Answer Record regarding PCIe Gen4 support in UltraScale+ devices
https://forums.xilinx.com/t5/PCIe-and-CPM/PCIe-Gen4-x8-support-for-Virtex-UltraScale-devices/td-p/927922

UltraScale+ Devices Integrated Block for PCI Express v1.3 Product Guide
https://www.xilinx.com/support/documentation/ip_documentation/pcie4_uscale_plus/v1_3/pg213-pcie4-ultrascale-plus.pdf

If implementing PCIe on the PS, there are some caveats to be aware of:

1. 100MHz clock. If you are a Host, your board will likely be generating the PCIe reference clocks for the system. One copy of this clock will need to be applied to the MGT GTR Reference Clock input. If you are an End Point, the 100MHz system clock from the Host will need to be applied to the MGT GTR Reference Clock input. Note: Only common clock mode is supported. This is PCIe lingo meaning that all 100MHz clocks are locked and are generated by a single clock generator (don't use independent oscillators). This limitation is due to bit errors that can occur if two independent clock sources are used but their frequencies differ by too many PPM, as is very possible with low stability oscillators or spread spectrum clocking.

2. MIO signal assignment. As per Xilinx's table below, if you are an End Point, the selection flexibility of the MIO is limited as PCIe_RESET_N must be ingested by the hard IP properly, whereas, if you are a Root Port, the generation of PCIe_RESET_N occurs outside of the hard IP (e.g. within software), and thus can be assigned to any MIO (make sure the MIO you select is located within a bank whose voltage meets the PCIe signaling requirements).

MIO Signals

The PCIe Root Port mode and Endpoint mode reset signals are routed to specific MIO pins as listed in Table 30-10.

Table 30-10: **PCIe Reset Signals on MIO**

PCIe Reset	MIO Pins	I/O	Default Input Value to Controller
Rootport reset output (use GPIO controller)	0 ... 77	O	~
Endpoint reset input	29,30,31,33,34,35,36,37	I	0

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=841

Alternatively, and to achieve higher theoretical throughputs, PCIe soft-IP can be instantiated to enable up to Gen3 x16 operation. The same PS clocking requirements typically apply (refer to your IP vendor's requirements), however, the reference clock is applied to the appropriate PL MGT Reference Clock input, rather than the PS MGT GTR Reference Clock input. Below are a couple common PCIe IP vendors.

Warning!
If utilizing the PL to implement PCIe, Tandem configuration mode may be required. If so, be very careful. Tandem mode specifies a two-stage configuration to allow the first stage configuration to be fast enough to meet the PCIe enumeration requirements. There are a multitude of Tandem configurations that can use different resources for the storage of the first- and second-stage loads. The method selected impacts the use/implementation of the PERSTN line, as well as placing limitations on PROM support details, and limitations on IP core and GT quad selections. For more information study PG213, especially chapter 4.
Ref: https://www.xilinx.com/support/documentation/ip_documentation/pcie4_uscale_plus/v1_3/pg213-pcie4-ultrascale-plus.pdf#page=88

Xilinx
https://www.xilinx.com/products/technology/pci-express.html#usplus_pcie

Northwest Logic
https://nwlogic.com/products/pci-express-solution/

2.10 Powering the Zynq US+

Most complex chips require multiple power rails and sequencing, the Zynq US+ is no exception. Besides architecting a good overall solution, designing a cost-effective, physically practical, and appropriately sized for the power required, is perhaps the greatest hardware challenge when designing with Zynq US+. The good news: Xilinx provides simple to follow instructions related to powering the rails. Here's how I handle this design:

1. List all of the rails you will need across the entire design (not just the ones you need for the Zynq US+). Make note of special low-noise requirements of certain rails (e.g. transceiver voltages, etc.). My last Zynq US+ design required a total of 18 unique rails. When it comes to the Zynq US+, make sure you know what each rail is and how it is used; spending time on this now, will make everything better later (don't just treat a rail as a pin name and voltage).

Warning!
Understand what each voltage rail does and where it is used.

Xilinx provides excellent references for this:

Voltage Rails identified: https://www.xilinx.com/support/documentation/data_sheets/ds925-zynq-ultrascale-plus.pdf#page=4

Identifying which rails can be shared and which must be independent:
https://www.xilinx.com/support/documentation/user_guides/ug583-ultrascale-pcb-design.pdf#page=32

Love this drawing:

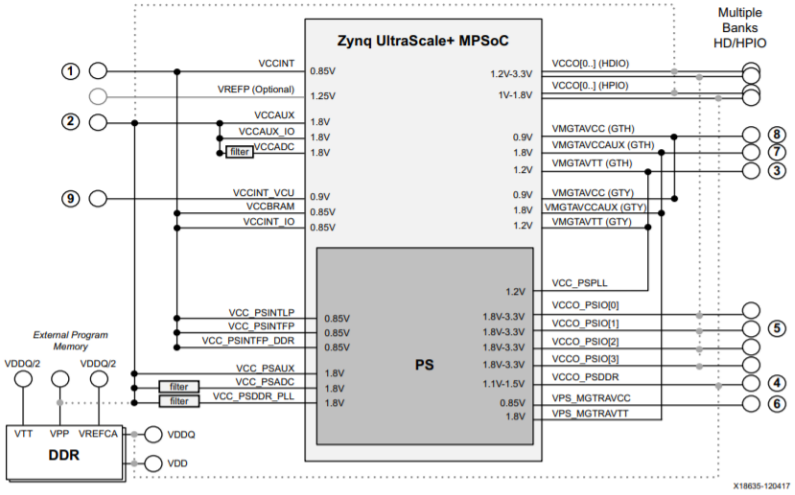


Figure 1-4: **Always On: Cost-Optimized Power Rail Consolidation**

Note: In Figure 1-4, the dashed lines are dependent on the user configuration.

Ref: https://www.xilinx.com/support/documentation/user_guides/ug583-ultrascale-pcb-design.pdf#page=35

2. Review all of the sequencing requirements for all of your multi-rail components and the components they interact with. Considering the connectivity between components is critical as you don't want to unintentionally have a powered component driving IOs of an unpowered component, as the protection diodes on the pins can leak the voltage onto the unpowered rail (we call this back-powering). If you drive an unpowered Zynq US+ IO, this will happen, so make sure that any peripheral devices that are powered prior to the Zynq US+ PS and PL IO are tri-stated until the Zynq US+ is fully powered.

PS and PL sequencing instructions
https://www.xilinx.com/support/documentation/data_sheets/ds925-zynq-ultrascale-plus.pdf#page=15

3. Group the rails according to their sequencing requirements. My last Zynq US+ design required three (3) stages of sequencing. My sequencing is always done one of two ways: One, by controlling the enable pin on the subsequent stage using the power good output from the current stage (this is my preferred method), or two, adding a MOSFET that is controlled by the previous stage's power good or voltage supervisor. I should mention that I have used complex, sometimes programmable power sequencers in the past, however, I have found that simple sequencing for the Zynq US+ works well. Note: I don't like adding additional factory programming requirements and maintaining software revisions for power sequencing, if it can be avoided.

4. Complete a power budget spreadsheet by assigning every rail for each component a typical and maximum current value. These current values for most components can easily be extracted from the component's datasheet, however, for Zynq US+, you must use the Xilinx Power Estimator (XPE) spreadsheet. The XPE is big, and frankly, complex, but by carefully reviewing the different line items and tabs, reviewing Xilinx documentation, but most importantly fully understanding your application (you will likely need to discuss with your FPGA and Software designers too), you can get a good idea of how much current each of your Zynq US+ rails will consume.

Warning!
Spend time understanding and completing the Xilinx Power Estimator. A good job here will make everything else easier.

Warning!
When running XPE power estimator, be sure to fill in the PS section at the end.

Warning!
Remember to set up environment variables on the first page.

Xilinx Power Estimator spreadsheet
https://www.xilinx.com/products/technology/power/xpe.html

5. Given all of the rails and their sequencing requirements, determine which rails can be shared (remember to refer to Xilinx's power rail sharing diagram above), and architect your power sub-system.

6. Select your components. Some rails will require switchers, some will require linear regulators, and some a little bit of isolation using beads or inductors. This also includes thinking about the decoupling capacitors you will need for the Zynq US+. Luckily, Xilinx put together detailed notes on decoupling.

Zynq US+ Decoupling Requirements
https://www.xilinx.com/support/documentation/user_guides/ug583-ultrascale-pcb-design.pdf#page=26

> For Zynq US+ power supply components, I really like Analog Devices' (formerly Linear Technology's) uModules. They are compact and various options can provide high-core currents, multiple outputs, and/or low-noise outputs. In addition, their integrated inductor is shielded which helps with your emissions. That said, be aware, these are easy to use parts, but since the inductor is integrated and hence of fixed value, you must be aware of efficiency (due to the higher switching frequencies employed to achieve minimum size) and pick the right part number for the job. Also be very aware of the minimum on-time requirements as you will often be stepping down a much higher input voltage to say a 0.8V to 0.9V core current. You will have multiple switchers in your design, I like to optionally synchronize mine to help avoid trouble during Regulatory Approvals.

Here's the link to ADI's uModules
https://www.analog.com/en/products/power-management/switching-regulators/umodule-regulators.html

2.11 Thermals

Thermal discussion is really beyond the scope of this document, I will leave the complex topic with just two thoughts:

1. Pay attention to thermals! The Zynq US+ is a complex, powerful chip. In my applications, I've seen the chip itself dissipate between 10W (small ZU device) and 100W+ (ZU19EG device). Proper thermal planning and solutions are critical. You will use the Xilinx Power Estimator and your system power budget to feed your thermal designer. Don't skimp on this or wait until the last minute for a thermal strategy – you may need to add heatsink mounting holes or other common features to your PCB.

2. It is important to consider how your Zynq US+ will be cooled both in production and during the development of your product. For example, Fidus's Sidewinder product ships with a custom heat spreader which is a sufficient passive cooling solution, as Sidewinder is intended to be installed in chassis with air-flow, but during the development of the product a fan-sink solution was used because the heat spreader blocked access to some of the test points on the PCB, and because Sidewinder was being tested bench-top in a natural convection environment.

Warning!
Pay attention to thermals and carefully design a thermal solution.

2.12 Device Configuration

The Zynq UltraScale+ is a flexible device, and thus, it makes sense that it can boot from many different types of non-volatile storage, the most common include: JTAG, QSPI, NAND, SecureDigital, and eMMC. Which one you select is driven by your end application. Certainly, we always provision JTAG – this will help us in the lab and on the manufacturing floor. QSPI is a cost-effective, fast, and physically small method to hold your information. NAND takes lots of pins and offers large capacity. SecureDigital is nice for in-field software upgrades, and super convenient if you are implementing a full file system. eMMC has similar benefits to SecureDigital, but with potentially lower cost and no physical user accessibility. Whichever boot method you employ you will typically house both your PS and PL information within this memory.

You tell the Zynq US+ which boot method to utilize by setting the PS_MODE[3:0] pins appropriately, as per the Xilinx table below:

Table 11-1: **Boot Modes**

Boot Mode	Mode Pins [3:0]	Pin Location	CSU Mode	Description
PS JTAG	0000	JTAG	Slave	PSJTAG interface, PS dedicated pins.
Quad-SPI (24b)	0001	MIO[12:0]	Master	24-bit addressing (QSPI24).
Quad-SPI (32b)	0010	MIO[12:0]	Master	32-bit addressing (QSPI32).
SD0 (2.0)	0011	MIO[25:21, 16:13]	Master	SD 2.0.
NAND	0100	MIO[25:09]	Master	Requires 8-bit data bus width.
SD1 (2.0)	0101	MIO[51:43]	Master	SD 2.0.
eMMC (1.8V)	0110	MIO[22:13]	Master	eMMC version 4.5 at 1.8V.
USB0 (2.0)	0111	MIO[52:63]	Slave	USB 2.0 only.
PJTAG (MIO #0)	1000	MIO[29:26]	Slave	PJTAG connection 0 option.
PJTAG (MIO #1)	1001	MIO[15:12]	Slave	PJTAG connection 1 option.
SD1 LS (3.0)	1110	MIO[51:39]	Master	SD 3.0 with a required SD 3.0 compliant voltage level shifter.

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=235

Note that the Mode Pins (i.e. PS_MODE[3:0]) are sampled on the de-assertion of PS_POR_B. My highlights from this table:

1. SD0 and SD1. This refers to the ability of the Zynq US+ to boot from either SecureDigital Controller. Note the Pin Location column in this table, and that booting from SD0 on MIO pins [58:30) is not supported, but other combinations are.

Warning

Be careful with SD card design – only SD1 can accommodate high speed 1V8. Also, highest speeds can only be accommodated at 1V8, but SD initialization is done at 3V3. Level shifters will be required. See UG1085 sections 11, 26 and 28.

2. USB0 (2.0). Slave only. Being Slave only means that you cannot boot from a USB stick attached to your USB Host Port. This boot method is not common, and Xilinx has more detailed documentation discussing it, and the requirements of the Host System that your System would be connected to.

3. eMMC is not found in the Master MIO table. This is because to boot from eMMC you must utilize a subset of pins from within the SD0 MIO [25:13] allocation. Although an eMMC is supported by the SD1 Controller, I am unsure if you can boot eMMC from the SD1 pins (I suspect not otherwise it would have been listed in the table above). Note: Xilinx says "The SD1/eMMC can only operate in 4-bit mode when it is mapped to MIO bank 3." For completeness, at the bottom of the Master MIO table, Xilinx mentions:

1. SD0/1 peripheral pins can also be configured as eMMC 0/1, respectively. The difference between SD and eMMC configuration is as follows.
   - The **Card Detect** and **Write Protect** signals are only available in SD mode.
   - The BUS_POW pin in SD mode is treated as a reset pin in eMMC mode.
   - In SD mode, data transfers in 1-bit and 4-bit modes. In eMMC mode, data transfers in 1-bit, 4-bit, and 8-bit modes.
   - If the SD interface is configured for SD 3.0, the signals SEL, DIR_CMD, DIR_0, and DIR_1_3 are mapped to sdio{0,1}_data_out [4], [5], [6], and [7], respectively.

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=790

Warning

DIP switches or jumpers are useful if you're not sure which boot mode you want to use! And they save you from having to bust out the soldering iron to Stuff/DNS resistors!

2.13 PL HP and HD

Zynq US+ PL (FPGA fabric) IO banks are designated as either High Performance (HP) or High Density (HD). HP, as its name infers, is where your highest speed IO will be connected. HP is limited to a maximum VCCO (i.e. your bank voltage) of 1.8V. HD is for general purpose use and can support a maximum VCCO of 3.3V.

You will always use HP banks to interface to DDR. Whereas, I typically use HD banks for low-speed interfaces or DC-like control signals that require 3.3V (chip enables, resets, interrupts, etc.).

Always be aware of your need for IO-types, internal terminations, etc, and ensure that the VCCO you select can support this. Once again, a smart hardware designer will have the FPGA designer verify the intended pinout in the Xilinx tools – you don't need a complete design to do this, and our FPGA designers typically only take a few hours to give our pinout the thumbs up.

Warning!

Always run your intended pinout through the Xilinx tools.

2.14 PS and PL DDR

Most Zynq US+ systems leverage both PS and PL DDR; typically, 64-bit wide (can reach 72-bits with ECC), sometimes multiple PL DDR interfaces. As one would expect, the PS DDR is used by the Operating System (e.g.

Linux), and the PL DDR is used for buffering data on the PL side. The PS and PL DDR are often required to interact via DMA.

Based on the memory structure, the following clips from Xilinx tables describe limitations on PS and PL DDR performance. Note: I have only clipped the DDR4 portion of the tables, the full tables also describe additional memory types.

Table 30: **PS DDR Performance**

Memory Standard	Package	DRAM Type	Speed Grade				Units
			-3E -2E/-2LE -1E		-2I/-2LI -1I/-1M/-1Q -1LI		
			Min	Max	Min	Max	
DDR4[4]	All FFV and FFR packages, FBVB900, SFVC784, and SFRC784	Single rank component	664	2400	1000	2400	Mb/s
		1 rank DIMM[1, 2]	664	2133	1000	2133	Mb/s
		2 rank DIMM[1, 3]	664	1866	1000	1866	Mb/s
	SFVA625[7]	Single rank component	664	2133	1000	2133	Mb/s
		1 rank DIMM[1, 2]	664	1866	1000	1866	Mb/s
		2 rank DIMM[1, 3]	664	1600	1000	1600	Mb/s
	SBVA484[7]	Single rank component	664	1066	1000	1066	Mb/s
		1 rank DIMM[1, 2]	664	1066	1000	1066	Mb/s
		2 rank DIMM[1, 3]	664	1066	1000	1066	Mb/s
LPDDR4[5]	All FFV and FFR packages, FBVB900, SFVC784,	Single die package[6, 7]	664	2400	1000	2400	Mb/s

Ref: https://www.xilinx.com/support/documentation/data_sheets/ds925-zynq-ultrascale-plus.pdf#page=30

See how the achievable PS DDR4 speed is not just dependent on the package type and speed grade, it is also dependent on the structure of the memory (e.g. chip-on-board vs DIMM vs # of ranks).

And for PL DDR:

Table 74: **Maximum Physical Interface (PHY) Rate for Memory Interfaces**

Memory Standard	Packages[1]	DRAM Type	Speed Grade and V_{CCINT} Operating Voltages					Units
			0.90V	0.85V		0.72V		
			-3	-2	-1	-2	-1	
DDR4	All FFV, FFR, and FBVB900 packages	Single rank component	2666	2666	2400	2400	2133	Mb/s
		1 rank DIMM[2, 3, 4]	2400	2400	2133	2133	1866	Mb/s
		2 rank DIMM[2, 5]	2133	2133	1866	1866	1600	Mb/s
		4 rank DIMM[2, 6]	1600	1600	1333	1333	N/A	Mb/s
	SFVC784 and SFRC784	Single rank component	2400	2400	2133	2133	1866	Mb/s
		1 rank DIMM[2, 3]	2133	2133	1866	1866	1600	Mb/s
		2 rank DIMM[2, 5]	1866	1866	1600	1600	1600	Mb/s

Ref: https://www.xilinx.com/support/documentation/data_sheets/ds925-zynq-ultrascale-plus.pdf#page=51

See how the achievable PL DDR4 speed is not just dependent on the package type and speed grade, it is also dependent on the structure of the memory (e.g. chip-on-board vs DIMM vs # of ranks vs VCCint voltage). Also take note that the PL DDR4, generally speaking, can operate faster that the PS DDR4 – Xilinx did after all start as an FPGA company!

Assigning pins for PS and PL DDR interfaces is a little different. PS DDR assignment is not as flexible as the PL DDR assignment. This is a very long subject and Xilinx captured it well. Please refer to the following Xilinx information:

PS DDR4 Pin Assignment
https://www.xilinx.com/support/documentation/user_guides/ug1075-zynq-ultrascale-pkg-pinout.pdf#page=82

PL DDR4 Pin Assignment
https://www.xilinx.com/support/documentation/ip_documentation/ultrascale_memory_ip/v1_4/pg150-ultrascale-memory-ip.pdf#page=104

PL DDR4 PCB Design Guidelines
https://www.xilinx.com/support/documentation/user_guides/ug583-ultrascale-pcb-design.pdf#page=64

The links above are for DDR4, the same documents contain instructions for other memory types.

Designing with high-speed memory is not easy – you have to get the pinout, hardware design, PCB layout, and Signal Integrity all correct, otherwise your interface will not work reliably, if at all. Fidus is an expert in this, give us a call and we'll help you.

Warning!
Understanding both PS and PL DDR4 pin assignment is critical. Ensure you read and fully understand the following references.

2.15 PL Transceivers

Similar to other Xilinx FPGAs, almost all of the Zynq US+ family members support PL-based Multi-Gigabit Transceivers (MGTs). Assuming you have MGTs, depending on the device and speed grade of your device, you may have GTH (16.3 Gbps) and/or GTY (28Gbps (-2 speed grade), 32.75 Gbps (-3 speed grade)) transceivers. Remember, only certain -3 speed grade devices support 32.75 Gbps, so if you need that speed, pick carefully, otherwise your GTYs will be limited to 28 Gbps.

The Zynq US+ transceivers are arranged in quads (i.e. four (4) transceivers grouped together). Each quad supports two (2) MGT reference clocks. Just like with the PS-based GTRs it is important to provide the clock or clocks to these pins that your application requires. For example, if I were implementing an HDMI sink (RX), I would run the three (3) HDMI data lines into the three (3) receivers, ground the fourth receiver, and run the HDMI receive clock into one MGT reference clock, and if I needed to support lower frequency HDMI signals, I would also provide a second on-board clock into the other MGT reference clock input. Make sure you have the clocks you need – the IP provider will state their requirements.

Warning!
Carefully review the IP clocking requirements.

Review the implementation of SerDes blocks with respect to the internal IP blocks you are intending to use, as not all internal hard IP blocks can be connected to all SerDes blocks. Information pertaining to this is detailed in UG1075. Internal IP blocks can only reach SerDes blocks on the same "side" of the device, and only 1 or 2 blocks north or south. See below for an example of what IP blocks can connect to what GT quads.

XCZU11, XAZU11, and XQZU11 Bank Diagram Overview

GTY Quad 131 X0Y16-X0Y19	PCIE4 X0Y3	HP I/O Bank 71	HD I/O Bank 91	GTH Quad 231 X0Y28-X0Y31
GTY Quad 130 X0Y12-X0Y15	CMAC X0Y1	HP I/O Bank 70	HD I/O Bank 90	GTH Quad 230 X0Y24-X0Y27
GTY Quad 129 X0Y8-X0Y11 (RCAL)	ILKN X0Y0	HP I/O Bank 69	HD I/O Bank 89	GTH Quad 229 X0Y20-X0Y23
GTY Quad 128 X0Y4-X0Y7	PCIE4 X0Y2	HP I/O Bank 68	HD I/O Bank 88	GTH Quad 228 X0Y16-X0Y19
GTY Quad 127 X0Y0-X0Y3	CMAC X0Y0	HP I/O Bank 67	PCIE4 X1Y1	GTH Quad 227 X0Y12-X0Y15
PS GTR 505	PS MIO 502	HP I/O Bank 66	SYSMON Configuration	GTH Quad 226 X0Y8-X0Y11 (RCAL)
PS DDR 504	PS MIO 501	HP I/O Bank 65	Configuration	GTH Quad 225 X0Y4-X0Y7
PS CONFIG 503	PS MIO 500	HP I/O Bank 64	PCIE4 X1Y0 (tandem)	GTH Quad 224 X0Y0-X0Y3

X15132-121517

Figure 1-16: **XCZU11, XAZU11, and XQZU11 Banks**

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1075-zynq-ultrascale-pkg-pinout.pdf#page=43

Once again, it is always wise to have the FPGA designer connect the IP to your pin assignment and confirm the tools accept your instantiation.

2.16 Clocks

2.16.1 PS CLOCKS

Clocking the PS is straightforward, and typically only requiring two clocks: A 32.768kHz crystal (across the PS_PADI and PS_PADO pins) to feed the internal RTC, and an oscillator connected to the PS_REF_CLK pin. PS_REF_CLK is the PS System's Reference Clock. There is some flexibility when selecting the PS_REF_CLK frequency, this is because there are PLLs that can be provisioned within the PS to help you generate other frequencies to run items, like the PS DDR interface. That said, make sure you pick a frequency that allows you to generate the frequencies you need.

I typically use 33.33MHz for this clock. The reference link below takes you to Xilinx's PS clock requirements information.

Warning!
Ensure a PS_REF_CLOCK clock exists on power up. Do not create this from a clock chip that must be configured by the PS section. The PS section needs this clock to operate and hence to program any clock parts.

Warning!
Don't forget the balance resistor across PS_PADI and PS_PADO!

Ref: https://www.xilinx.com/support/documentation/data_sheets/ds925-zynq-ultrascale-plus.pdf#page=32

2.16.2 PL GLOBAL CLOCKS

Xilinx FPGAs utilized the concept of global clocks – the Zynq US+ PL is no different. Both HD and HP banks support global clocks (reminder: HD supports up to 3.3V and HP up to 1.8V IO). Global clock implementation is fairly straightforward, however, be aware of whether your VCCO bank voltage supports the internal termination (i.e. 100ohms differential). In a lot of my designs, I have to add an external 100ohm resistor. An example would be running an LVDS clock to the global clock input of a 3.3V HD bank. This would require an external 100ohm termination, as the internal termination is not supported. Also be aware whether you need to AC couple and potentially re-bias your global clock, based on your combination of VCCO and IO-standards involved.

2.17 XTP427 Schematic Checklist

XTP427 is a schematic checklist to help make sure you haven't missed anything major! You will need a Xilinx account to gain access to this utility.

3 Debug Peripherals Every Zynq US+ Design Should Have…

The following peripherals will assist you in the design, debug, and lab bring-up of your Zynq UltraScale+ design. I recommend adding these to your designs, and then depopulating the unnecessary ones once you're ready for production.

3.1 Reset Switches

Connect push button reset switches to the following signals to aid in debugging. Of course, recognizing that under normal operation a supply voltage supervisor, or similar, must automatically activate and deactivate these signals.

PS_POR_B: Power On Reset. This is a hard reset that re-starts both the Low Power Domain (LPD) and the Full Power Domain (FPD).

PS_SRST_B: System Reset. This is a 'little less hard' than PS_POR_B as there are several registers that are unaffected (see reference below). This is usually connected to the debugging tool (e.g. JTAG connector).

PS_PROG_B: PL Configuration Reset. Similar to the usual FPGA configuration PROG_B pin.

Ref: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf#page=1123

3.2 Status LEDs

Connect LEDs to the following signals to aid in debugging. These definitions were copied from Xilinx's TRM.

PS_INIT_B: Indicates that the PL is initialized after a power-on reset.

PS ERR STATUS: Indicates a secure lockdown state. Alternatively, it can be used by the PMU firmware to indicate system status.

PS_DONE: Indicates the PL configuration is complete.

PS_ERROR_OUT: Asserted for accidental loss of power, a hardware error, or an exception in the PMU.

3.3 Debug Headers

When real-estate permits, I like attaching debug headers to the buses associated with the PS hard controllers. This gives me the ability to observe signals, generate signals, and generally aid in debugging.

I2C: 3-pins required, SCL, SDA, GND. Make sure that your debug utility can support the onboard voltage you have selected for this interface. I like the Total Phase Aardvark (I2C/SPI Host Adapter) for mastering this bus, and the Total Phase Beagle (I2C/SPI Protocol Analyzer) for observing this bus.

Total Phase Aardvark
https://www.totalphase.com/products/aardvark-i2cspi/

Total Phase Beagle
https://www.totalphase.com/products/beagle-i2cspi/

SPI: Minimum of 5-pins required, SS, SCLK, MISO, MOSI, GND. Of course, if you have more than one device you may want to support more Slave Selects. Once again, the Total Phase Aardvark and Beagle are good for either mastering or observing this bus respectively.

PS IO: I recommend connecting a few (1 or more) of your unused PS IOs to a header and/or LED. This way you can observe status information to assist in software debug.

PL IO: Similarly, I recommend connecting a few (1 or more) of your unused PL IOs to a header and/or LED. This way you can observe status information to assist in FPGA debug.

3.4 Boot Mode Configuration

If you have multiple boot methods, which I'm sure you will, I recommend provisioning a DIP switch to enable you to change the boot method quickly. Place resistors such that in production your default boot mode is set, as you will depopulate the DIP switch.

3.5 Spare Clocks

If possible, provision a footprint for a common oscillator package (e.g. 5x7) and connect it to a spare global clock. If either you or your FPGA designer forgot about a needed frequency, this is a way out.

3.6 SecureDigital

Xilinx supports the option to boot the Zynq US+ from a Secure Digital interface. Given the amount of software that is typically written and debugged, it is extremely convenient for the software designers to apply their loads via SD. So, even if your end design will always boot from QSPI, if you have the pins and real estate available, put down a simple SD or uSD interface – your software colleagues will thank you!

4 Typical Zynq UltraScale+ Applications

Throughout this document I have discussed the power and flexibility of Zynq US+, this section contains just three (3) examples of the many implementations that Fidus has enabled with Zynq US+.

4.1 Video Streaming

Video streaming is massive these days. Functionally, to contribute content to the internet, one must accept video from an input device (e.g. web cam, video camera, HDMI source, image sensor, etc.), adapt the format to optimize transmission (i.e. aligned with bandwidth and user needs; color depth, color space, frame rate, resolution), compress as appropriate (based on desired bandwidth), and then transmit to the Network (i.e. 10/100/1000 Ethernet). Functionally, to consume content, one must do the same, but in reverse order: Receive packets, decompress, format for the intended display, and output to the display. A Zynq US+ EV family member is ideal for this. Here's why:

1. Sink and Source connectivity capability
    a. HDMI: Zynq US+ can receive and transmit HDMI utilizing the PL transceivers. Xilinx provides an HDMI subsystem for this.
        i. DisplayPort: Zynq US+ supports DisplayPort TX/RX using the PS GTRs, if you prefer DP over HDMI
    b. SDI: Zynq US+ can receive and transmit SDI utilizing the PL transceivers. Xilinx provides an SDI subsystem for this.
    c. Web Cam: Zynq US+ has two (2) internal USB3.0 capable Hosts.
    d. Image Sensor: Zynq US+ supports the connection of D-PHY based MIPI image sensors to the PL. Northwest Logic (nwlogic.com) and Xilinx provide MIPI Controllers for this.

2. Format Conversion: There are three (3) ways format conversions can be done. Which way you use depends on which type or types of conversion you are trying to accomplish. The first method is to instantiate a PL-based block like Xilinx's Video Processing Subsystem (VPSS). This block can complete a multitude of conversions and formatting. The second way, if you're not doing anything too heavy, is to leverage pre-built functions within GStreamer[1] and simply do your conversions on the APU/GPU within the PS. And the third way is by doing a combination of both. For example, we like doing computationally intensive functions like color space conversions in the PL (fabric) and then things like frame-rate conversions within GStreamer. Frame rate conversions make sense because the PL would require an external frame buffer and all of the memory support logic around it, whereas, using GStreamer does not require any additional infrastructure as the PS already has a frame buffer, and the memory accesses for handling for 'simple' frame rate conversation is almost free.

3. Compression/Decompression: As discussed, the EV version of the Zynq US+ has an internal hard core optimized and dedicated for either H.264 (AVC) or H.265 (HEVC) compression and decompression, called the VCU (Video Codec Unit). Xilinx has demonstrated extremely low latency with this engine, so it is very useful for a multitude of applications. The H.264/H.265 cores can also handle multiple streams and can compress and decompress simultaneously, however, be aware that like anything, the VCU's capability is limited, so read your Xilinx documentation carefully – understanding the core capability as well as the bandwidth demands on the memory interface.

[1] GStreamer is a multi-media framework that supports a multitude of libraries. Refer to: https://gstreamer.freedesktop.org/ and https://en.wikipedia.org/wiki/GStreamer.

H.264/H.265 Video Codec Unit
https://www.xilinx.com/products/intellectual-property/v-vcu.html

4. Network Transmission: As discussed, the Zynq US+ PS contains four (4) gigabit ethernet MACs, so getting your streaming data to the network is more or less already in place and you just need to add a PHY.

From the above you can see that Zynq US+ EV has all of the resources you need to create a streaming video solution.

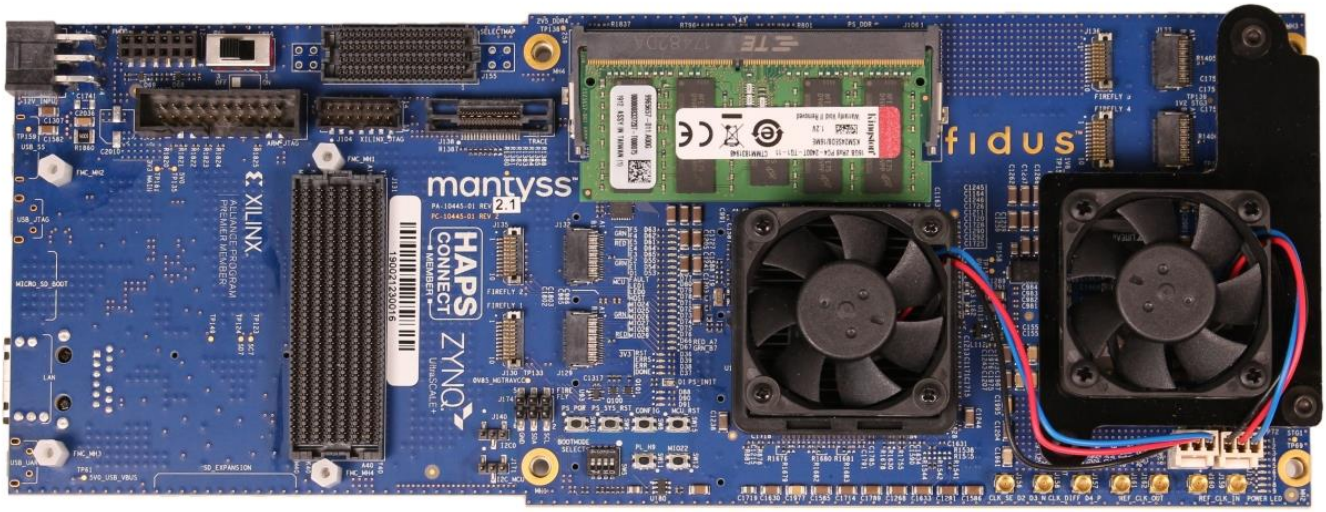
Would love to show you a picture of some of the Customer products we have designed in this space, alas, we can't share Customer data… but trust me, they're really cool.

4.2 Compute Offload and Acceleration

The Zynq US+ also fits well in compute-offload and acceleration applications. These applications typically pass the heavy data processing workload to the Zynq US+, so that the Zynq can crunch away on one or multiple data sets in parallel, and then advise the Host System that it has a result ready to return. In these implementations, to enable integration with a Host System, the Zynq US+ is typically placed on a PCIe expansion card form factor solution. This type of instantiation is typically characterized by a x4, x8, or x16 PCIe Gen3 bus, one or more banks of PL-based on-board DDR4 memory, and little or no other external connectivity.

Some compute-offload and acceleration functions benefit a lot more from an FPGA "sea of gates" rather than from an MPSoC with a PS and some PL. So, depending on your requirements, you might find that a Kintex® US+ or Virtex US+ is better suited to supporting a multitude of independent memory interfaces resulting from high levels of parallelization. That said, some functions benefit from the local monitoring and control that the PS can offer, making Zynq US+ a good choice. Either way, know your algorithm, know the capabilities of the potential devices, and pick the right one (or just give it to Fidus and we can help).

A picture of Fidus's Zynq US+ based, Mantyss™ offload system for Synopsys' HAPS System:



4.3 Storage

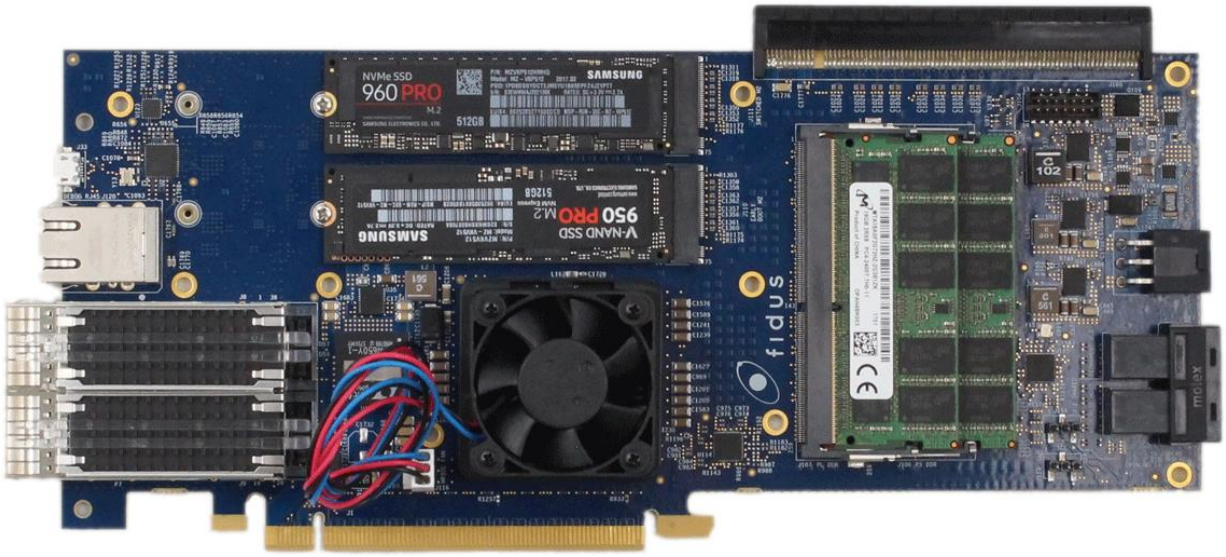
Due to crazy increases in the amount of available data, various storage topologies (local, NAS/SAN, etc.), increases in drive capacities, decreases in cost per gigabyte, and storage speeds, the Storage market is growing at an exponential rate. Given the external interfaces, memory interfaces, throughput capabilities, PL-offload capabilities, and Operating System functionality, the Zynq US+ is an ideal candidate for implementing remote or standalone storage solutions.

An example of a Fidus implementation is a recorder for streaming data. Note: the data could be video, ethernet packets, really anything. This system could be implemented leveraging a high-speed front-end, like Video, 100GE, or PCIe, a storage control system, a data processing block, a storage technology, and perhaps an additional storage access port. Here's how Zynq US+ could implement this:

1. High-speed front-end. Zynq US+ provides many options for receiving data.
    a. Video. As discussed in the Streaming Video example above Zynq US+ supports a plethora of different video interfaces.
    b. 100GE. Some Zynq US+ devices have a built in 100GE CMAC that obviates the need to buy IP and keeps the PL open for your custom design. Alternatively, 10/40GE soft IP could be purchased and instantiated.
    c. PCIe. Some applications will require the Zynq US+ device to receive and store data over PCIe. As previously outlined, a Zynq US+ device can support up to Gen2 x4 Host or End Point functionality via PS hard block or up to Gen3 x16 Host or End Point functionality via PL soft IP instantiation. Typically, this type of application likes to move data on and off the card quickly, and thus the wider the better.

2. Storage Control System: These drives just don't run themselves! Let's consider the case of storing to an NVMe SSD, you will need a PCIe Host Controller likely paired with an NVMe Accelerator IP. The NVMe Accelerator IP enables the PL to control the drive with little or no intervention from the Zynq US+ APUs. This piece is important because having to pass all data through the processor would bottleneck your storage system.

A picture of Fidus's Zynq US+ based, Sidewinder Storage Accelerator product:



Xilinx Accelerator IP
https://www.xilinx.com/products/intellectual-property/ef-di-nvmeha.html#overview

Design Gateway IP
https://dgway.com/APS-IP_X.html

NVMe Accelerator IP
https://www.intelliprop.com/ipc-nv164-hi/

3. Data Processing Block. This is really whatever you need it to be and sometimes nothing at all. For example, perhaps your mission is to search the data within the drives for a certain key pattern, then you could implement your function (or buy another piece of IP) either in the PL or the PS. But if all you're doing is receiving, storing, and retrieving data, you likely don't need this block.

4. Storage Technology. There are a lot of common storage technologies, but I'll focus on the one that's all the rage right now for high-performance systems: NVMe SSD. This is due to its ubiquitous PCIe interface, low latency, the huge IOPs (throughput) capability, and the simpler control mechanisms. NVMe SSDs are becoming the preferred storage mechanism - albeit capacity and cost can sometimes be an issue, but things are getting better every day!

5. Storage Access Port. Not all applications require it, but sometimes the data arrives on one port and a user needs to extract the stored data from another port. This means adding an additional high-speed port to retrieve your stored data. There are a million ways of doing this, so I won't get into it here, but suffice it to say, due to its architectural flexibility, the Zynq US+ can accommodate this extra port, and read-out can occur through the PL or via the PS.

   To enable fast, simpler, low-latency access to remote storage, NVMeOF (NMVe over Fabrics) and RDMA technology are becoming more and more popular and fit well into a Zynq US+ storage-based system. Refer to the links below:

Mellanox (purchased by Nvidia)
https://community.mellanox.com/s/article/what-is-nvme-over-fabrics-x

Xilinx
https://www.xilinx.com/publications/product-briefs/nvme-2017-web.pdf

5 About Fidus Systems

Since 2001, Fidus has been delivering complex customer designers for our customers. Fidus built its reputation on delivering value. To both Us and our Customers, value means delivering expertise and top-notch solutions, all at a fair price. In 2011, Xilinx named Fidus its inaugural North American Premier Design Services Member. Fidus was only selected after a Xilinx audit that inspected Fidus's technical capabilities, Fidus's business practices, and Fidus's processes – I'm sure the fact that we had recently completed our 100th Xilinx design didn't hurt either!

Our large design team are experts in FPGA, Software, Hardware, Signal Integrity, and Mechanical/Thermal design. Our tools and capabilities enable us to deliver turn-key system designs, but also single discipline consulting and assistance. We are not a Contract Manufacturer (CM), however, we have CM partners to manufacture Prototypes and can deliver Production quantities.

To best serve our customers, Fidus has three (3) Design Centers: Our Headquarters in beautiful Ottawa, Canada, our Kitchener-Waterloo location, just outside of bustling Toronto, Canada, and our Bay Area office in well-located Fremont, California. If we don't have a center near you, don't worry, we routinely do work with customers all over the World.

Our Mission To be the leader in transforming innovative ideas into great products.

Our Vision As a dynamic technology partner, we deliver differentiated solutions through seamless execution. We consistently exceed client and employee expectations leading to long-term relationships.

If you have a custom design need or simply a question, whether it involves Zynq US+ or not, **contact us today**!

www.fidus.com

info@fidus.com

About the Hardware Designer

Scott Turnbull has been with Fidus Systems since pretty much the beginning. Through various companies, he started his career as a Hardware Designer and then moved into System Design. These days, Scott, Director of Technology, handles Technical Sales and helps define and design Fidus' next generations of Products. Scott has been responsible for Fidus's Xilinx relationship for the past decade and has often led presentations and discussions with Xilinx's Executives, Business Units, and Field Staff.

6 References

6.1 General Zynq US+ References

For any Zynq US+ design, review and know these!

Zynq® UltraScale+™ MPSoC Data Sheet: DC and AC Switching Characteristics
https://www.xilinx.com/support/documentation/data_sheets/ds925-zynq-ultrascale-plus.pdf

Zynq® UltraScale+™ Device Technical Reference Manual
https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf

UltraScale™ Architecture SelectIO Resources User Guide
https://www.xilinx.com/support/documentation/user_guides/ug571-ultrascale-selectio.pdf

UltraScale™ Architecture PCB Design User Guide
https://www.xilinx.com/support/documentation/user_guides/ug583-ultrascale-pcb-design.pdf

UltraScale™ Architecture Configuration User Guide
https://www.xilinx.com/support/documentation/user_guides/ug570-ultrascale-configuration.pdf

Zynq® UltraScale+™ Device Packaging and Pinouts Product Specification User Guide
https://www.xilinx.com/support/documentation/user_guides/ug1075-zynq-ultrascale-pkg-pinout.pdf

Zynq® UltraScale+™ MPSoC Package Device Pinout Files
https://www.xilinx.com/support/package-pinout-files/zynq-ultrascale-plus-pkgs.html

Xilinx® Power Estimator (XPE)
https://www.xilinx.com/products/technology/power/xpe.html

UltraScale™ Architecture-Based FPGAs Memory IP v1.4 LogiCORE IP Product Guide
https://www.xilinx.com/support/documentation/ip_documentation/ultrascale_memory_ip/v1_4/pg150-ultrascale-memory-ip.pdf

High Speed Serials (link to main transceiver page)
https://www.xilinx.com/products/technology/high-speed-serial.html#documentation

UltraScale™ Architecture System Monitor User Guide
https://www.xilinx.com/support/documentation/user_guides/ug580-ultrascale-sysmon.pdf