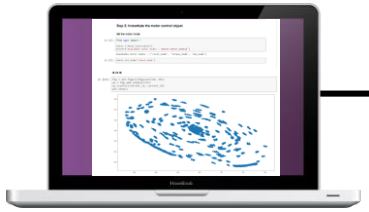
Getting started

© Copyright 2018 Xilinx

PYNQ ython Productivity for Zynq

Jupyter notebooks,
browser-based interface

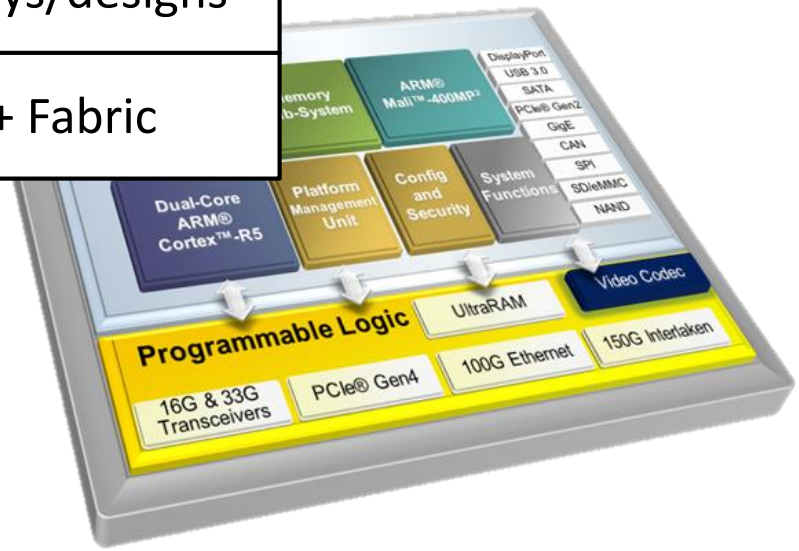
Most competitive, open
and advanced GUI



PYNQ enables JupyterLab
on Zynq and ZU+

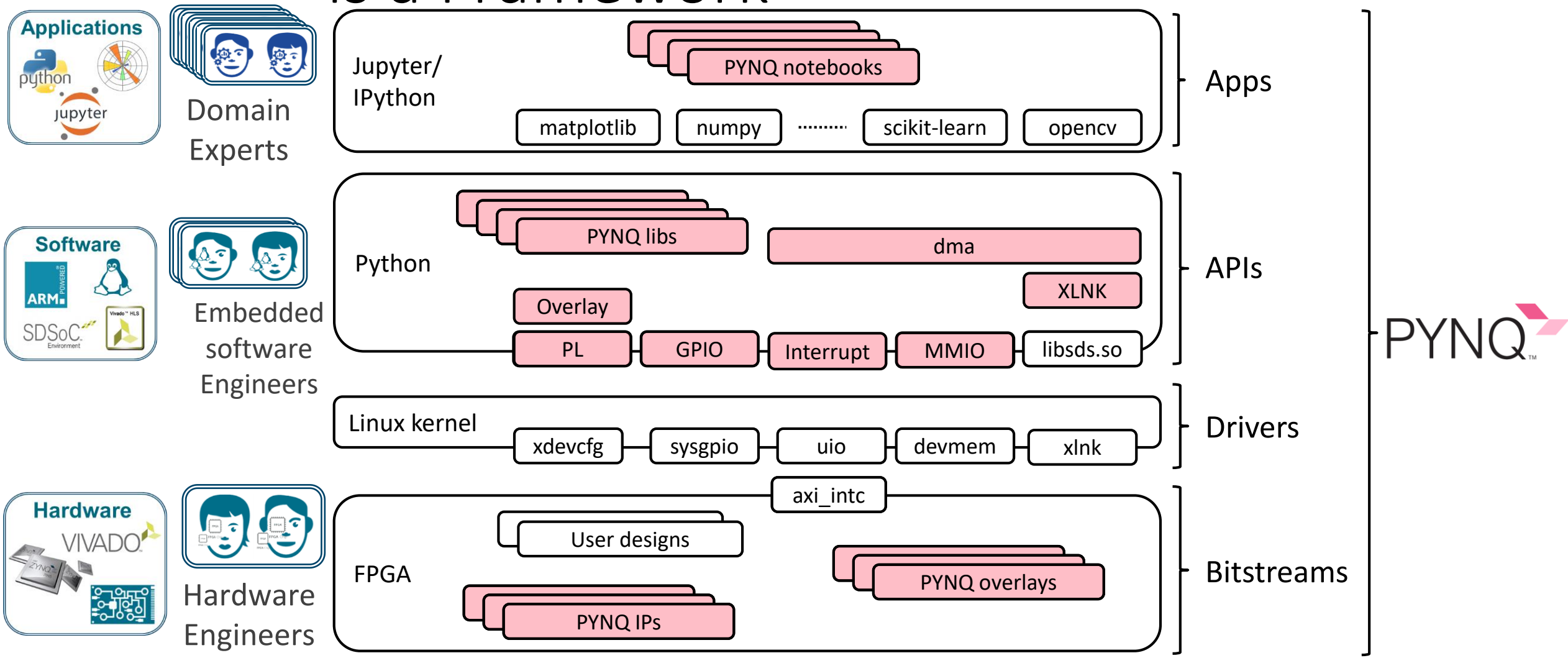
Jupyter web server	
IPython kernel	
Ubuntu-based Linux	Overlays/designs
ARM A9 / A53	ZU+ Fabric

Hardware C-drivers wrapped in Python packages



PYNQ

is a Framework



Outline

- Configure board, SD card
- Connect to the board
- Login in to portal
- Jupyter Notebook

© Copyright 2018 Xilinx

Prerequisites

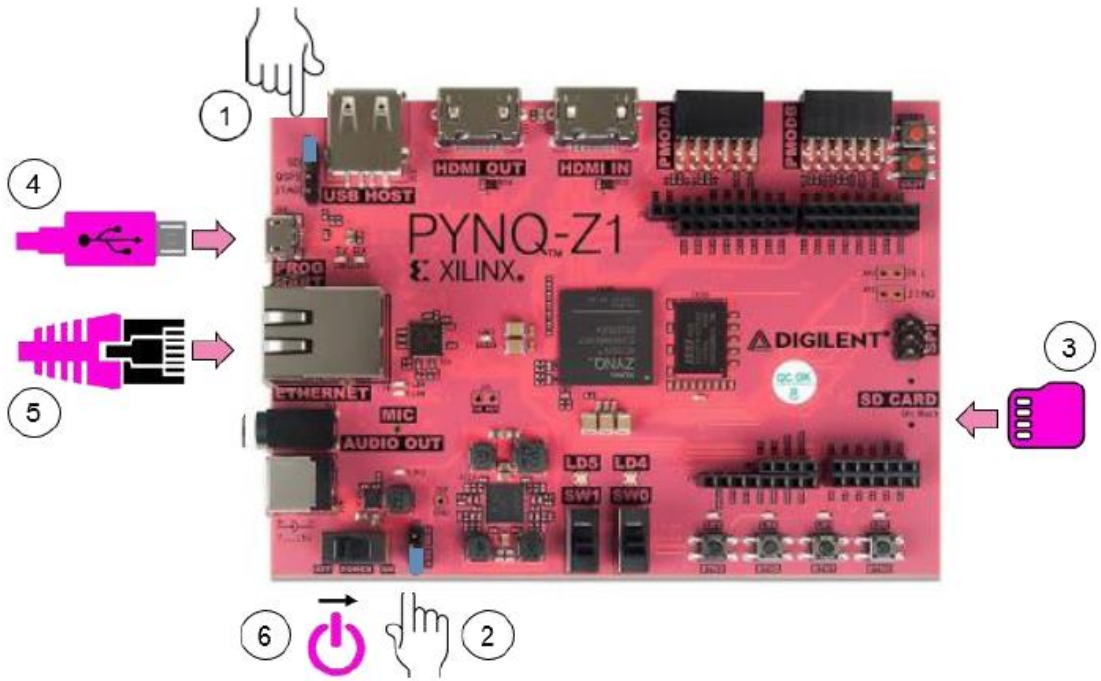
*Instructions to download and prepare SD card:
http://pynq.readthedocs.io/en/latest/getting_started.html

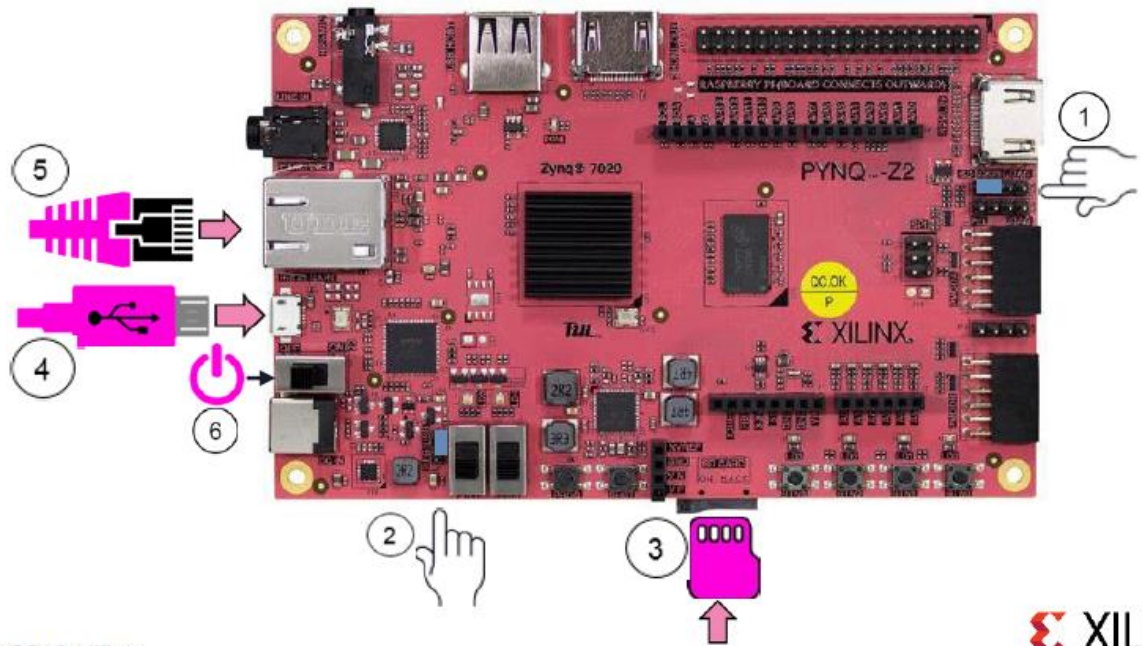
© Copyright 2018 Xilinx

XILINX

Connecting to the board

1. Configure board to boot from SD Card
2. Set Jumper to power from USB
3. Insert SD Card
4. Connect USB cable
5. Connect Ethernet cable to PC or to a Switch/Router
6. Power On



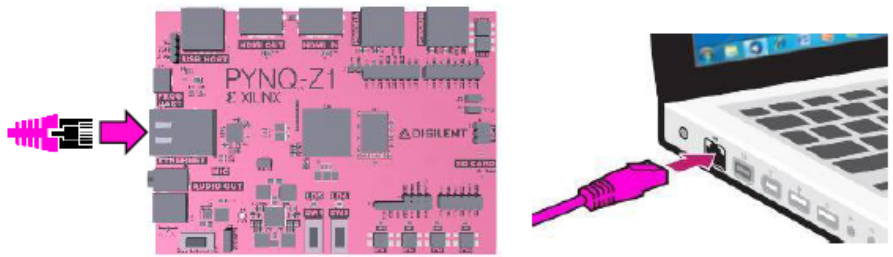


© Copyright 2018 Xilinx

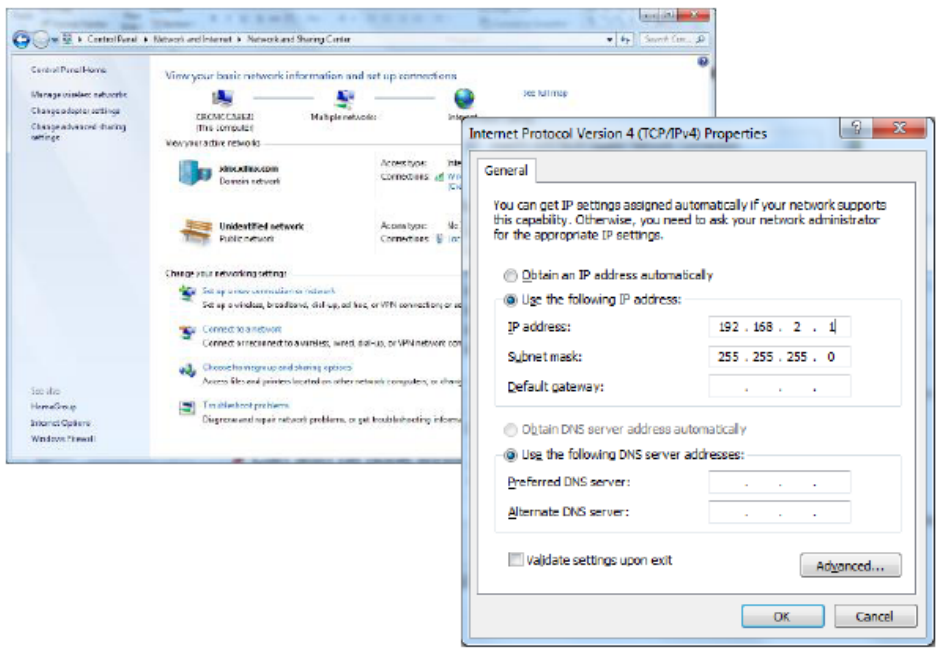
XILINX

Connecting to the board – Direct connection

- **Connect board directly to Ethernet port on PC**
  - USB to Ethernet adapter if no Ethernet port available
- **Board IP will default to 192.168.2.99**
- **Manually specify static IP for PC**
  - Must be in same range as board:
    - E.g. 192.168.2.1
  - No internet access unless you bridge network connections



Connect board directly to PC

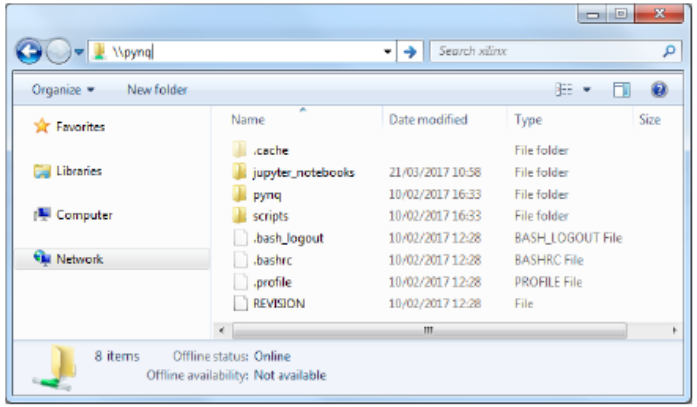


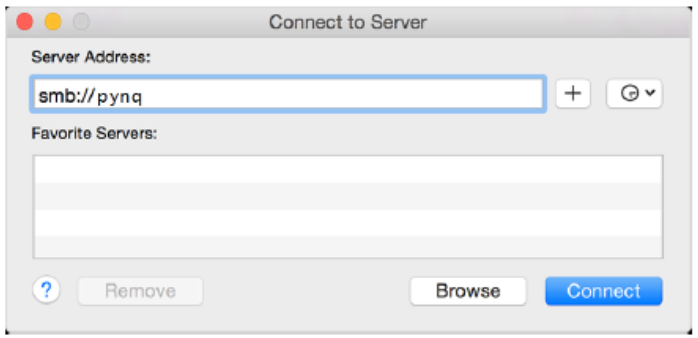
© Copyright 2018 Xilinx

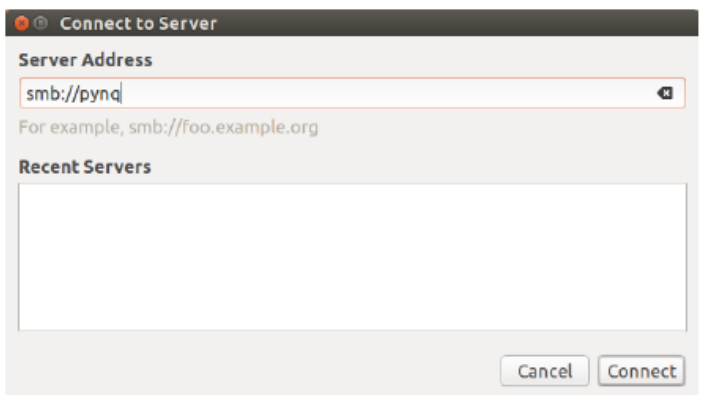
XILINX.

Samba share

- **Board can also be accessed as a shared drive**
  - Windows: \\192.168.2.99\xilinx
  - Linux: smb://192.168.2.99/xilinx
  - MAC OS: smb://pynq/xilinx
    - Hit Command+K to bring up the 'Connect to Server' window
- **Log-in**
  - Username = xilinx
  - Password = xilinx
- **Copy files easily between PC and board**



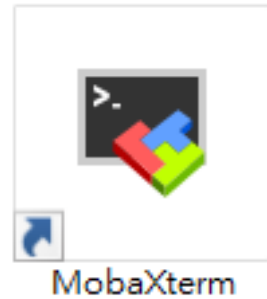




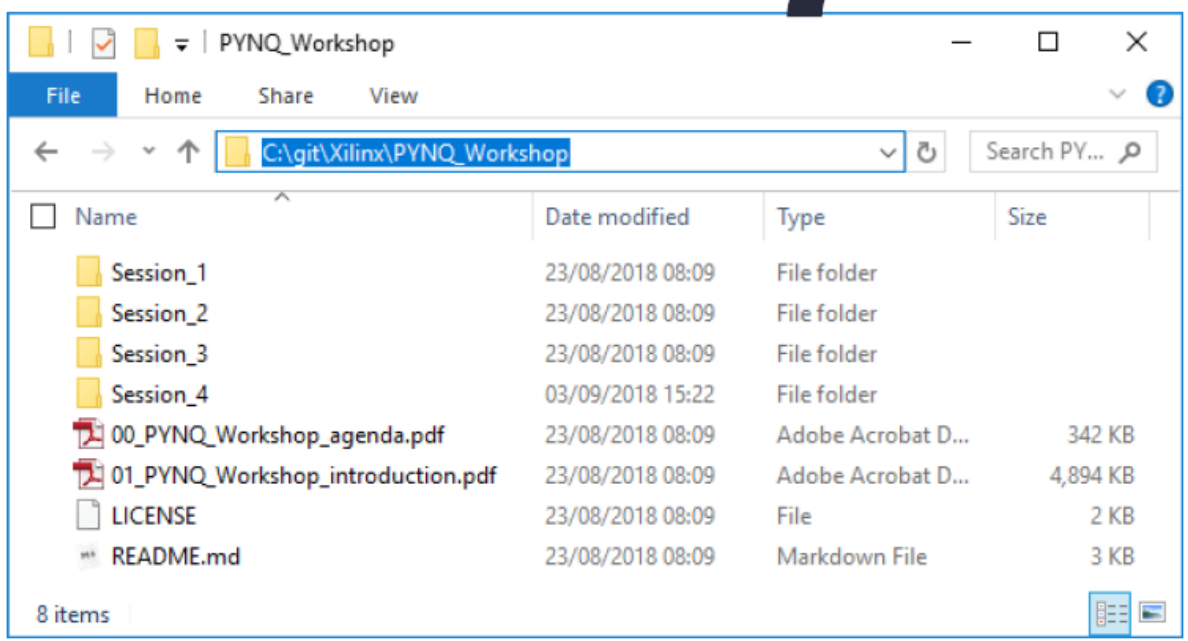
© Copyright 2018 Xilinx

XILINX.

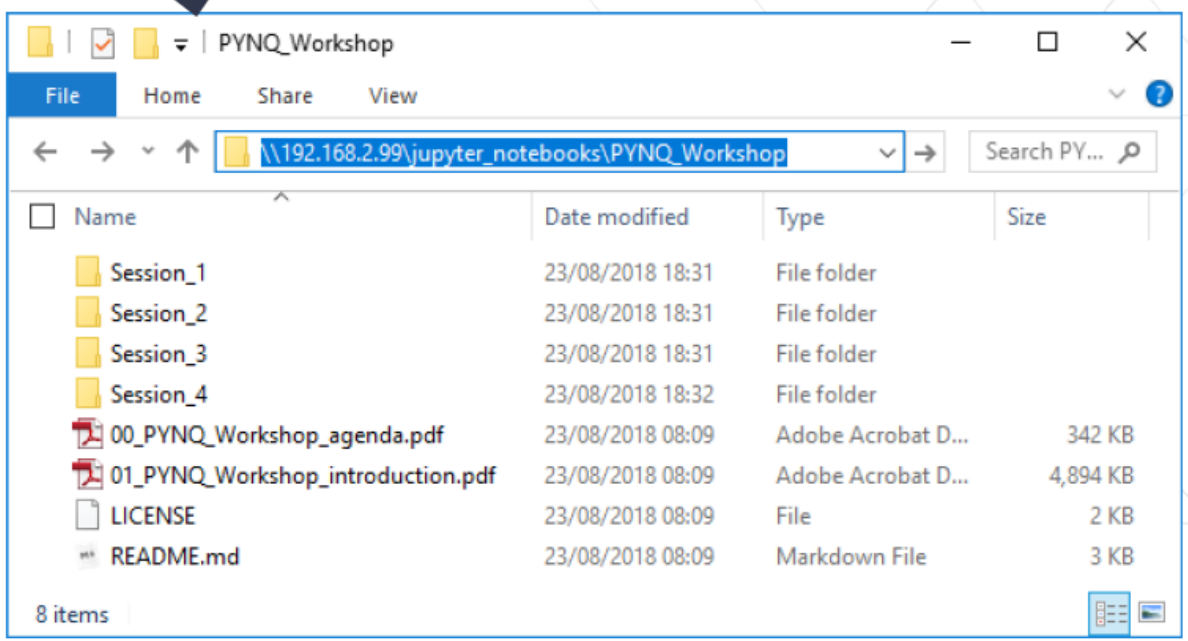
Introduction to Jupyter notebooks



> **Copy workshop files to the board: \\192.168.2.99\xilinx\jupyter_notebooks**



From laptop

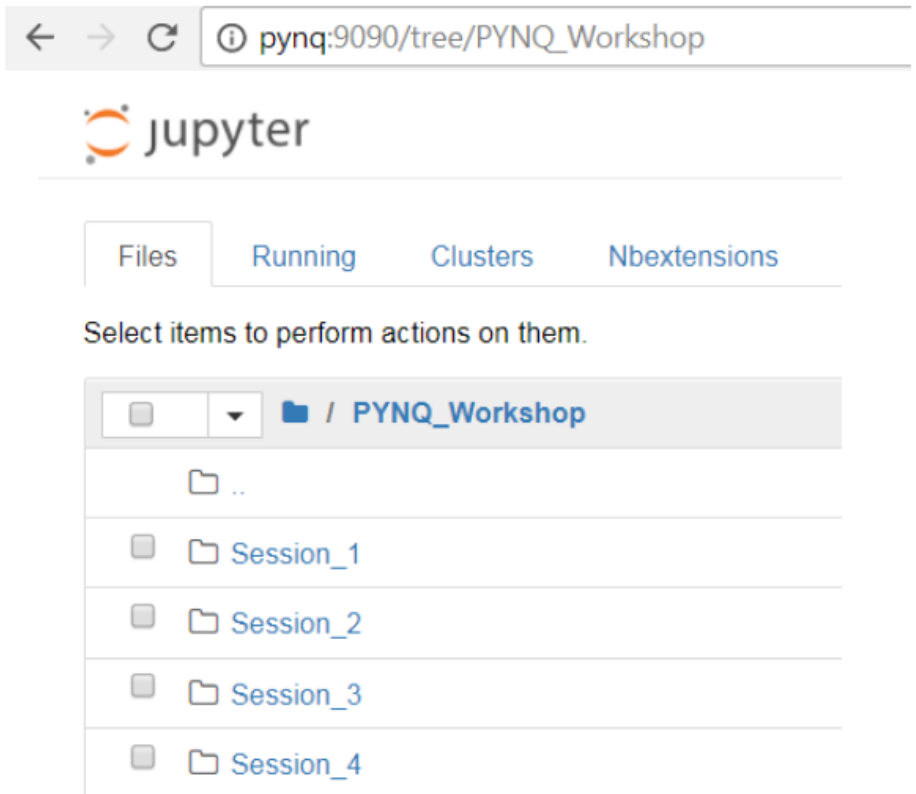


To the board

© Copyright 2018 Xilinx

Log in to Jupyter portal

- **Open a browser***
  - Chrome (preferred)
- **Browse to: http://*192.168.2.99*:*9090***
- **password = xilinx**
- **Browse to PYNQ_W*orkshop* folder**



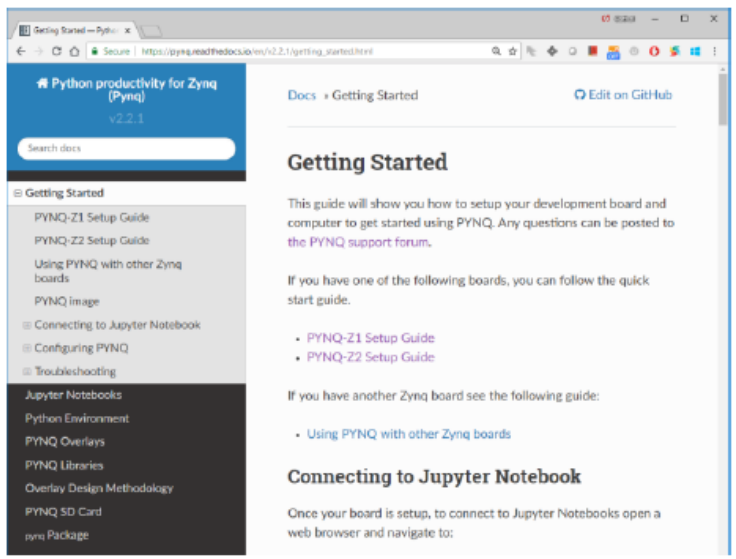
*http://jupyter-notebook.readthedocs.io/en/latest/notebook.html#browser-compatibility

The board doesn't have a realtime clock. The first time a board is used, the date and time of the system may be too far out of sync, and cause some browser (e.g. FireFox) to refuse setting a cookie which prevents log-in to the board. Chrome does not have this issue. To resolve this issue, update the date on the board. In a terminal execute: sudo date +%Y%m%d -s "20180920 "
"20180920" is YYYYMMDD

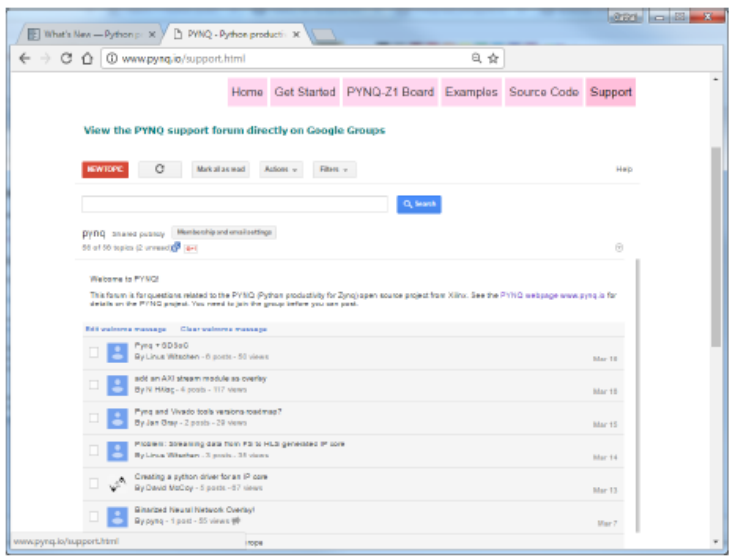
© Copyright 2018 Xilinx

Documentation and Support

- **Documentation**
  - pynq.readthedocs.io
- **Support**
  - pynq.io/support



- **GitHub**
  - Issue tracker
  - github.com/Xilinx/PYNQ
- **All accessible from**
  - www.pynq.io



© Copyright 2018 Xilinx

XILINX.

Lab exercises: Session 1

- **Getting started with Jupyter Notebooks**
  - Notebook's browser-based interface
  - Writing text with Markdown
  - Writing and running Python scripts
  - The IPython interpreter
- **Programming on-board peripherals**
  - Controlling on-board LEDs
  - Interacting with buttons, switches, and LEDs

© Copyright 2018 Xilinx

Questions?

XILINX

PYNQ

© Copyright 2018 Xilinx

© Copyright 2018 Xilinx

Outline

- Overlay Concept
- Base Overlay
- External devices support
- Python programmer's view

FPGA overlays – hardware libraries

> **Overlays are generic FPGA designs that target multiple users with new design abstractions and tools**

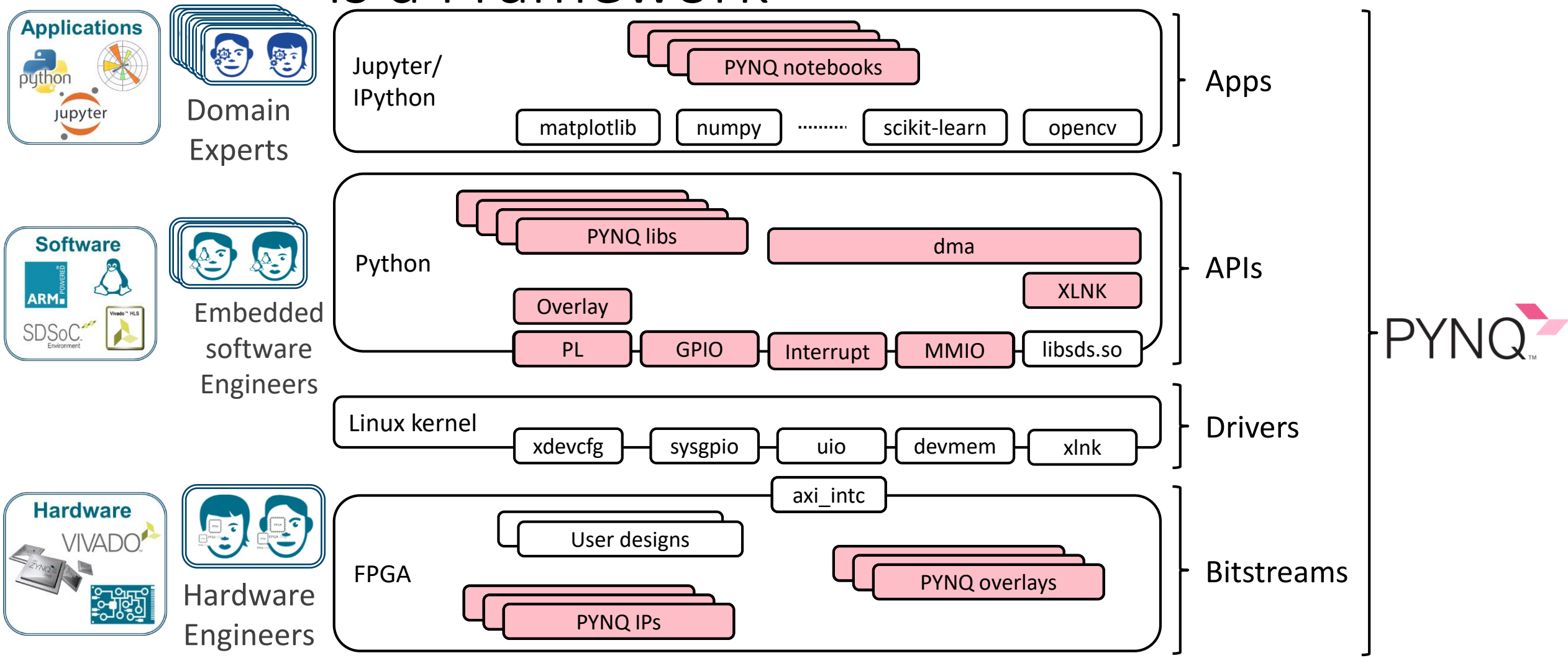
> **Overlay characteristics**

- Post-bitstream programmable via software APIs
- Typically optimized for given application domains
- Encourages the use of open source tools & fast compilation
- Enables productivity by re-using pre-optimized designs
- Makes benefits of FPGAs accessible to new users

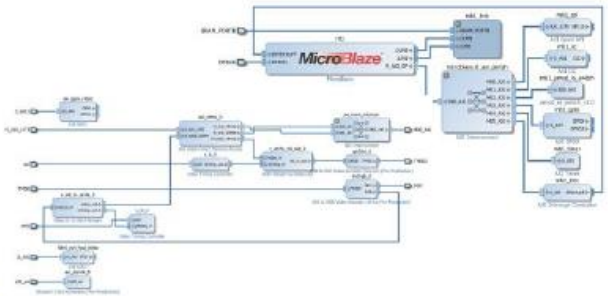
© Copyright 2018 Xilinx

PYNQ

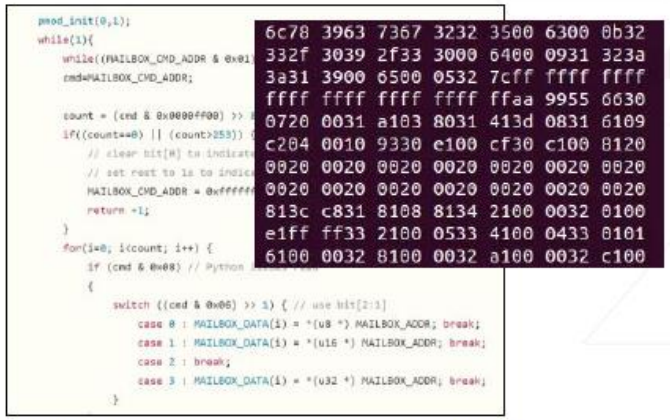
is a Framework



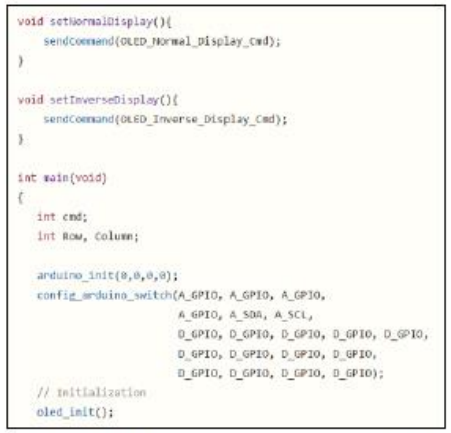
FPGA overlays – hardware libraries



Step 1:
Create an FPGA design for a class of related applications



Step 2:
Export the bitstream and a C API for programming the design



Step 3:
Wrap the C API to create a Python library

Step 4:
Import the bitstream and the library in your Python scripts and program

© Copyright 2018 Xilinx

XILINX.

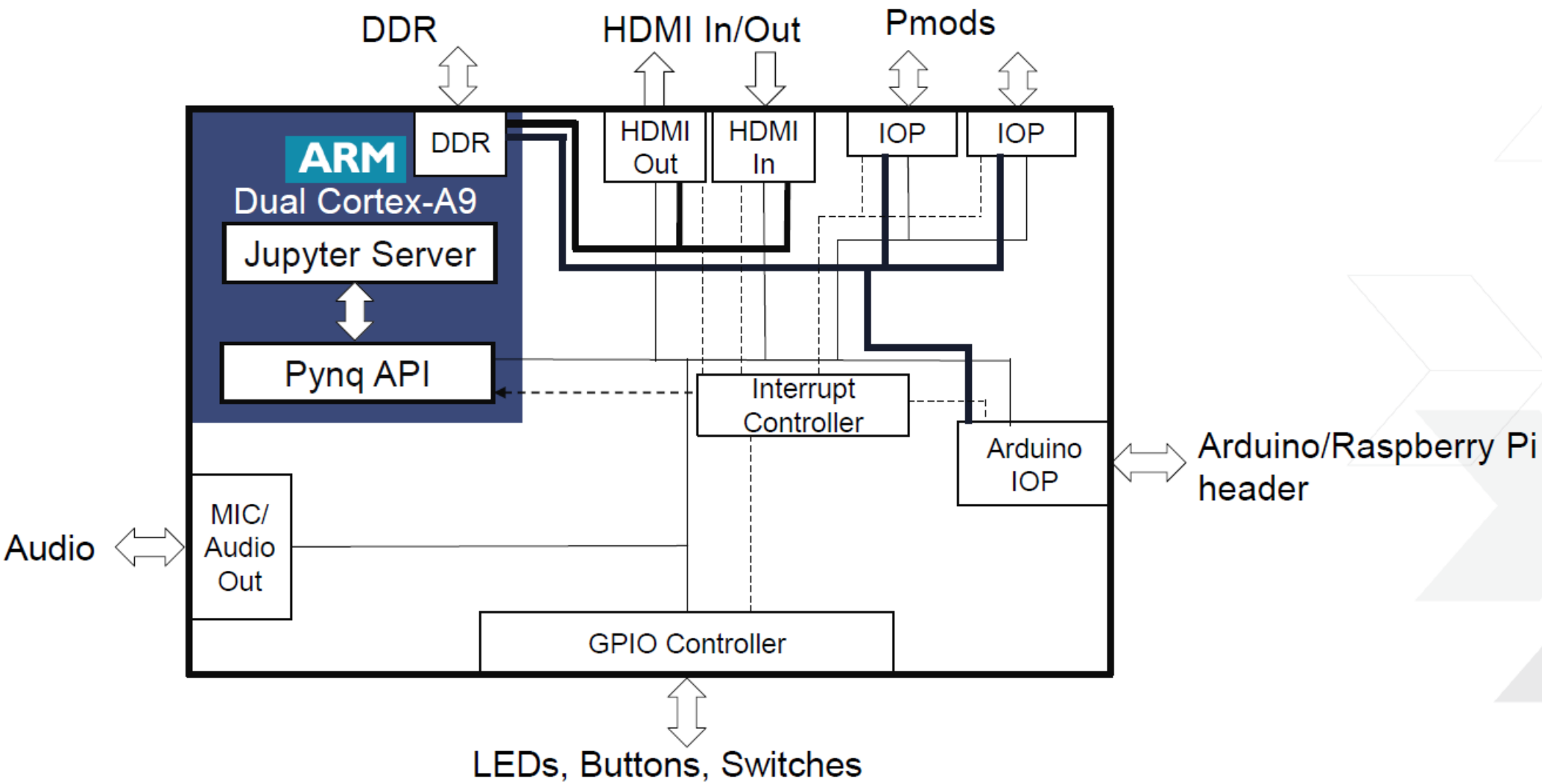
The rest of the *base* Overlay

XILINX

PYNQ

© Copyright 2018 Xilinx

Complete *base* Overlay (base.bit)

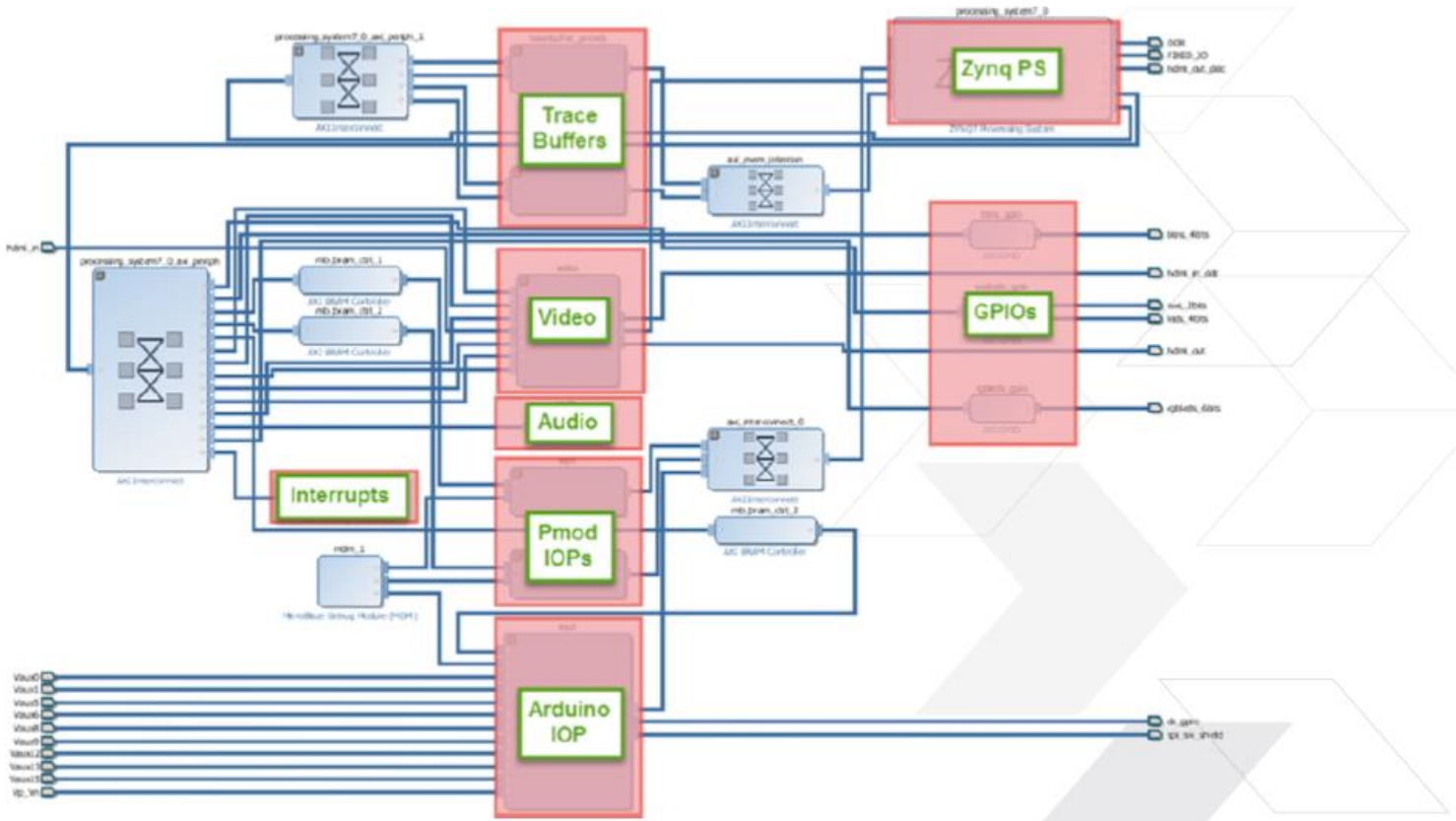


© Copyright 2018 Xilinx

XILINX.

base Overlay – Vivado Interface View

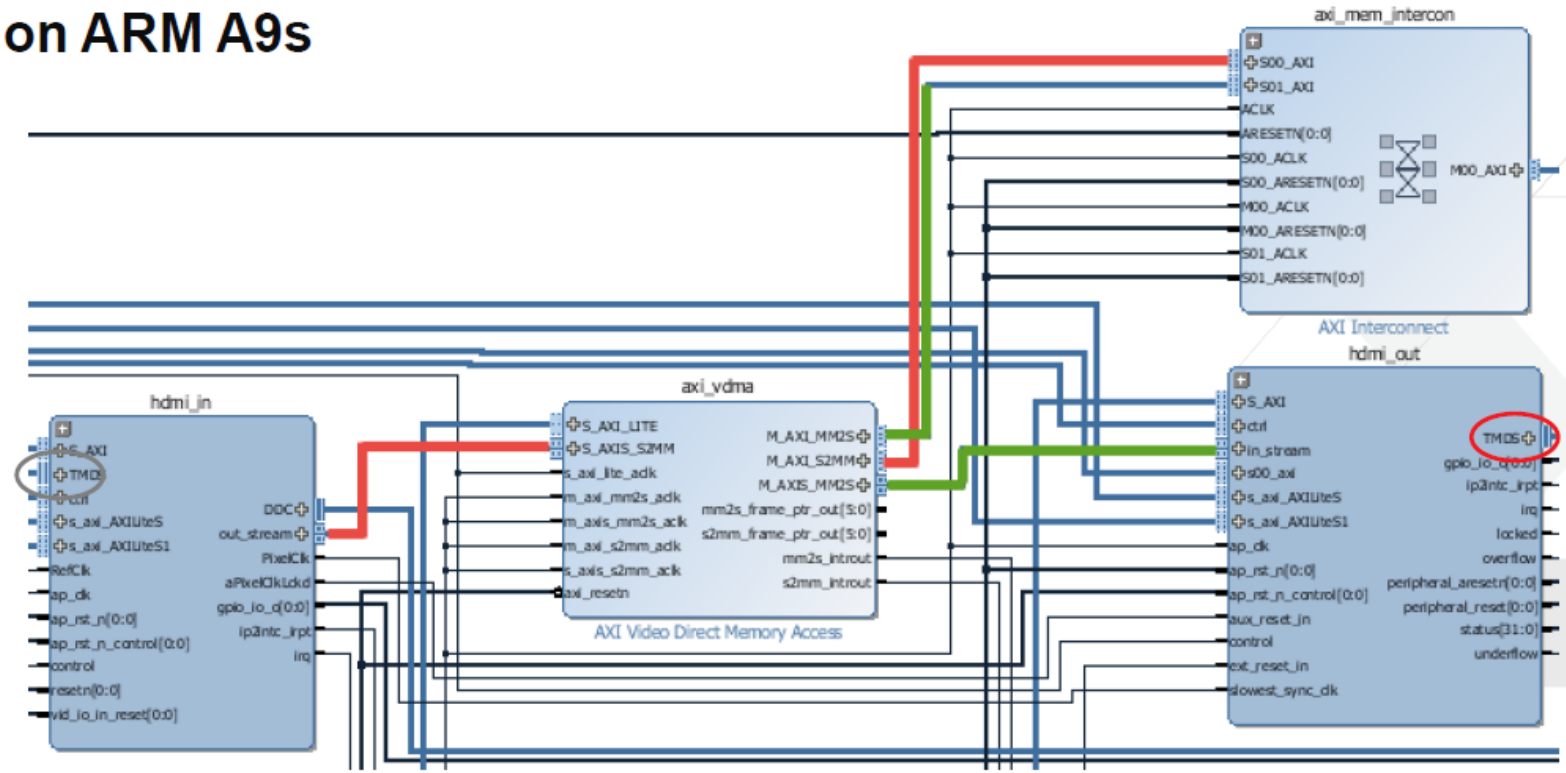
- PL design of the base Overlay
- Standard FPGA design flow used
- Vivado IPI
- Interface to Python
- Memory Map
- Open source
- Components are re-useable



© Copyright 2018 Xilinx

Video

- **HDMI_in, HDMI_out**
  - Stream from HDMI_in to DRAM; stream from DRAM to HDMI_out
  - 3 separate DRAM framebuffers available
- **Image processing on ARM A9s**
  - E.g. OpenCV

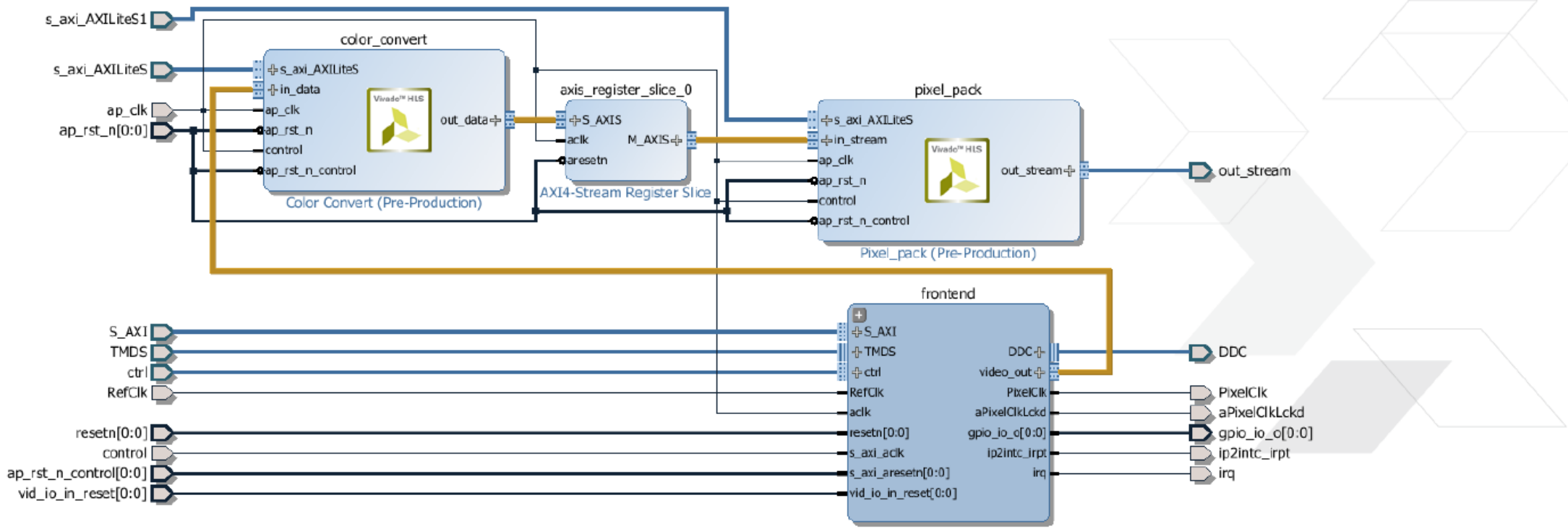


© Copyright 2018 Xilinx

XILINX.

Video In path

- **HDMI_in (TMDS) -> frontend -> color_convert -> axis_register_slice -> pixel_clock -> axis_vdma -> HP0 (PS7)**
  - The frontend module wraps all of the clock and timing logic

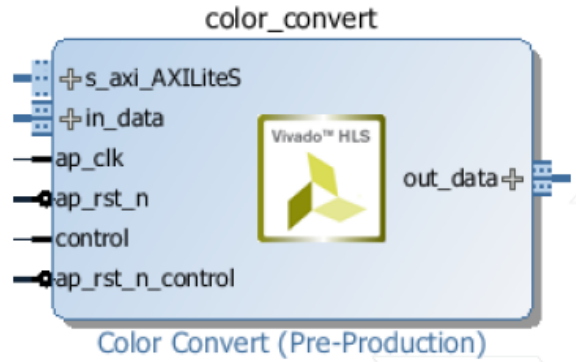


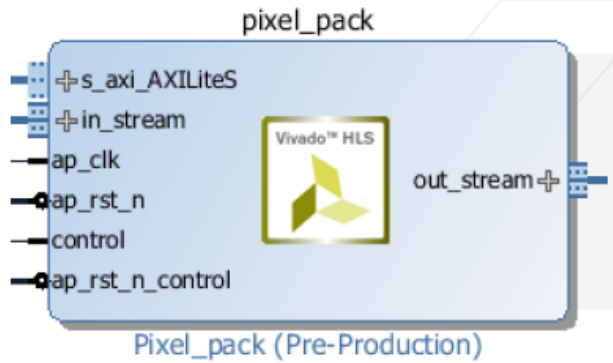
© Copyright 2018 Xilinx

XILINX.

Color conversion and pixel packing

- **color_convert**
  - Transform the input signal into different color spaces
- **pixel_pack**
  - Convert between 8/24/32-bit
- **Default: BGR (24-bit)**
  - RGB (24-bit)
  - RGBA (32-bit)
  - BGR (24-bit)
  - YCbCr (24-bit)
  - Grayscale (8-bit)
- **HLS source available**



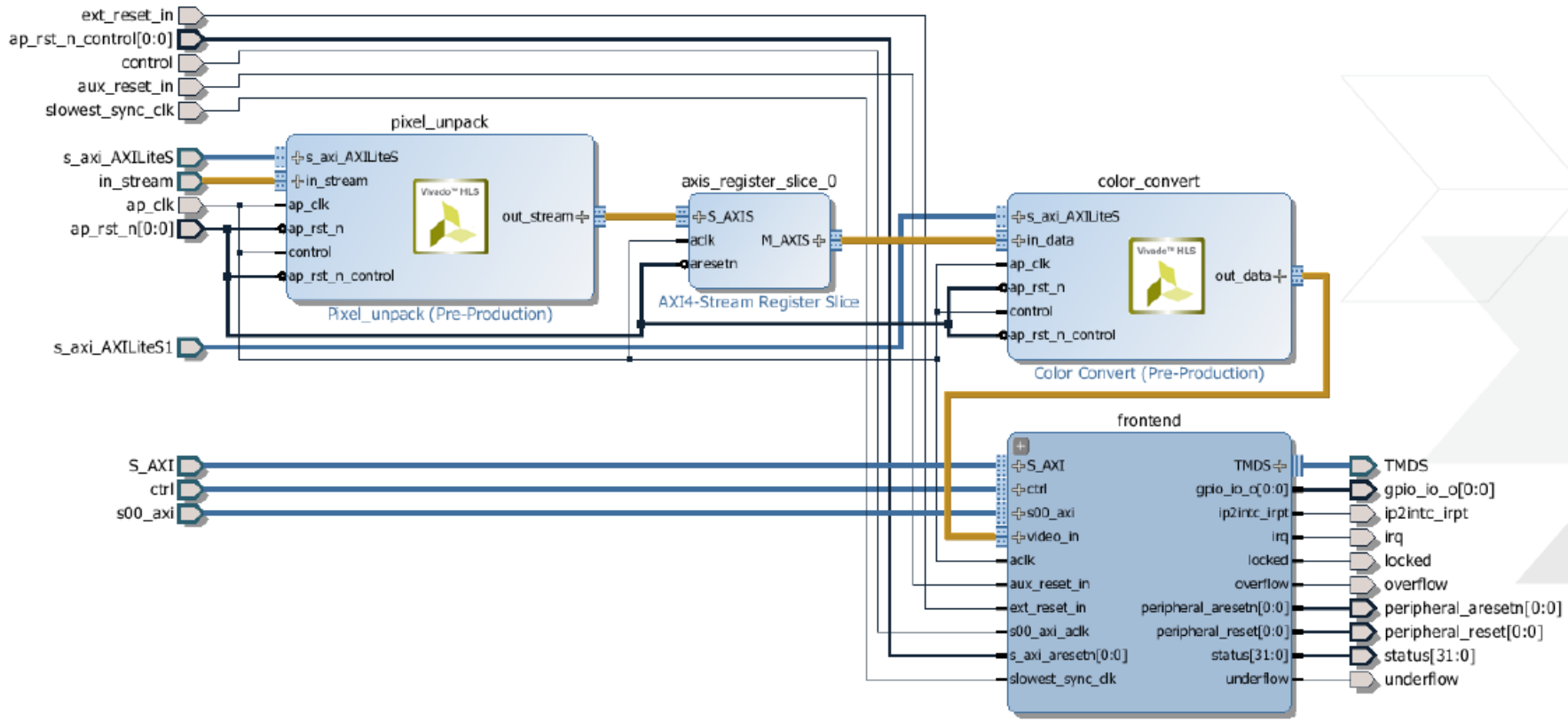


© Copyright 2018 Xilinx

XILINX.

Video Out path

- **HP0 (PS7) -> axi_vdma -> pixel_unpack -> axis_register_slice -> color_convert -> frontend -> HDMI_out (TMDS)**
  - All sub-modules perform reverse operations of the HDMI IN block



© Copyright 2018 Xilinx

XILINX.

Base overlay resource utilization

- **Z-7020, LUTs resource utilization ~50%**
  - IOP (Pmod) ~ 6%
  - IOP (Arduino) ~ 12%
  - Video ~ 15%
  - IOP (RPi) ~ 10%

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	LUT Flip Flop Pairs (53200)	Block RAM Tile (140)	DSPs (220)
base_wrapper	37339	52149	963	104	13100	35699	1640	18304	79	18
base_i (base)	37339	52149	963	104	13100	35699	1640	18304	79	18
video (video_imp_1KRFORE)	8648	16255	323	20	4036	8320	328	4790	10	18
iop_arduino (iop_arduino_im...	6563	7234	137	7	2229	6380	183	3016	16	0
iop_rpi (iop_rpi_imp_RNFCEZ)	5116	5747	129	8	1687	4923	193	2299	16	0
ps7_0_axi_periph (base_ps7...	3748	5888	61	61	1680	3389	359	1731	0	0
iop_pmodb (iop_pmodb_imp...	3488	3885	128	4	1289	3332	156	1433	16	0
iop_pmoda (iop_pmoda_imp...	3486	3885	128	4	1322	3330	156	1425	16	0
trace_analyzer_pi (trace_anal...	1644	2545	0	0	689	1565	79	1022	3	0
trace_analyzer_pmodb (trace...	1318	1927	0	0	523	1244	74	796	2	0

The cost of handling the interfaces is modest

© Copyright 2018 Xilinx

XILINX.

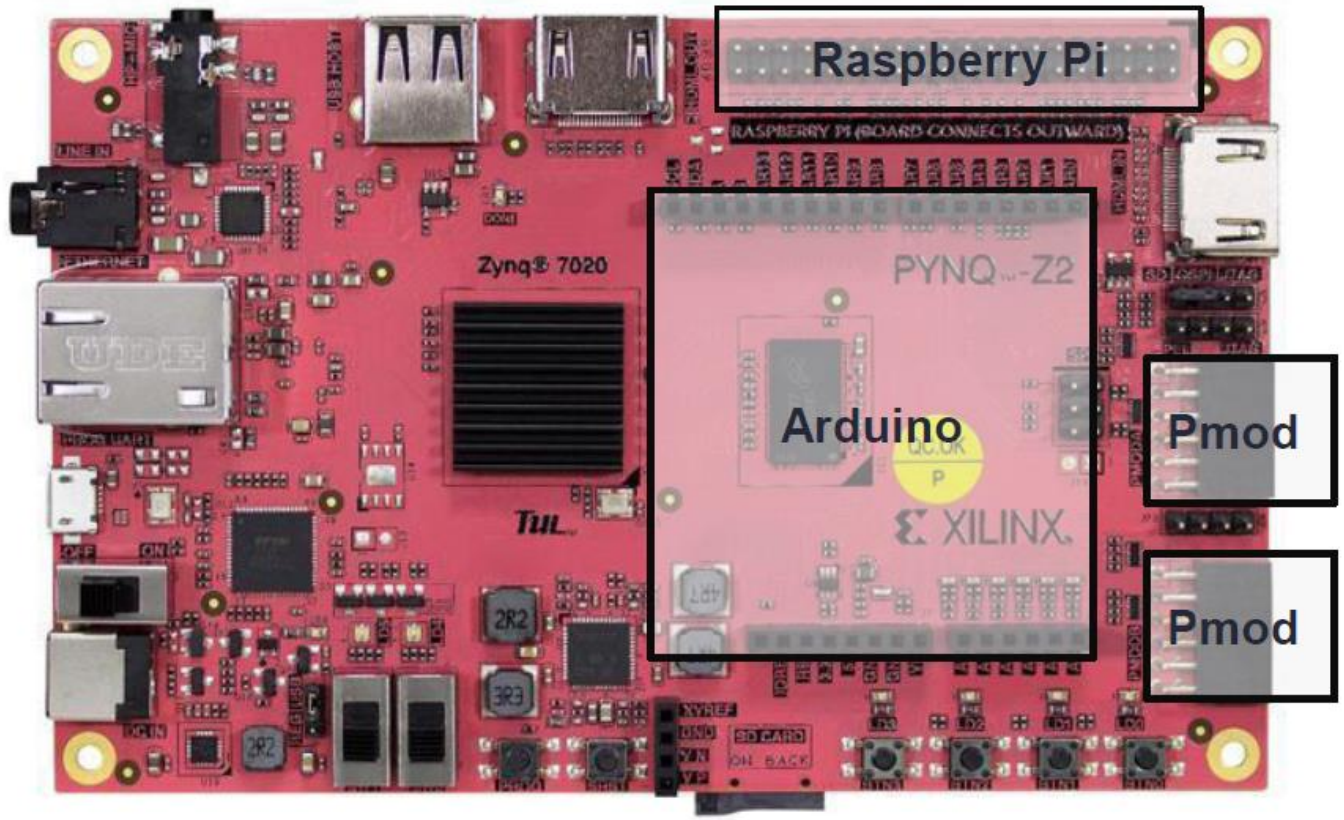
External interfacing with the Base Overlay

XILINX

PYNQ

© Copyright 2018 Xilinx

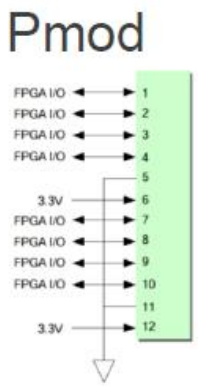
Low-cost PYNQ boards: Pmod, RPi, Arduino Interfaces

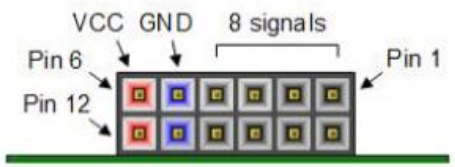


Typically every new Pmod, Raspberry Pi, or Arduino module requires a new design/seperate bitstream

© Copyright 2018 Xilinx

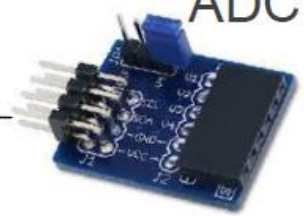
Pmod: many physical & electrical instances

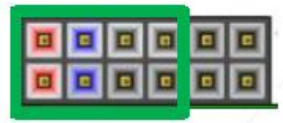




Pin	Signal	Description
1 & 5	SCL	Serial Clock
2 & 6	SDA	Serial Data
3 & 7	GND	Power Supply Ground
4 & 8	VCC	Power Supply (3.3V/5V)

ADC

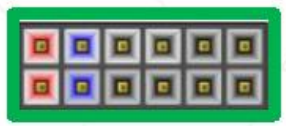




Connector J1

Pin	Signal	Description
1	CS	SPI Chip Select (Slave Select)
2	SDIN	SPI Data In (MOSI)
3	None	Unused Pin
4	SCLK	SPI Clock
7	D/C	Data/Command Control
8	RES	Power Reset
9	VBATC	V_{BAT} Battery Voltage Control
10	VDDC	V_{DD} Logic Voltage Control
5, 11	GND	Power Supply Ground
6, 12	VCC	Power Supply

OLED



Pin	Signal	Description
1	~CS	Chip Select
2	MOSI	Master-Out-Slave-In
3	(NC)	Not Connected
4	SCLK	Serial Clock
5	GND	Power Supply Ground
6	VCC	Power Supply (3.3V/5V)

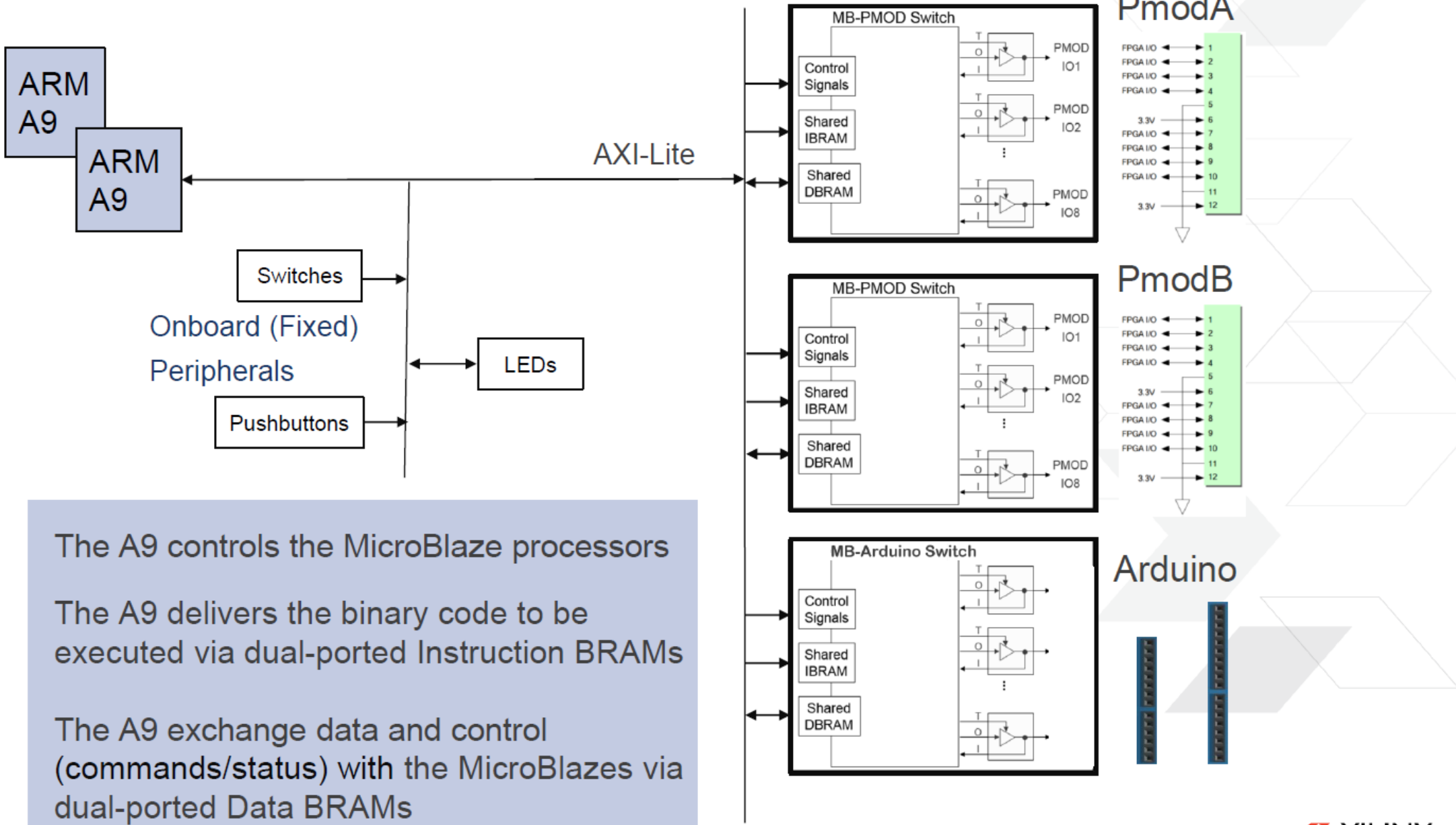
DAC

What if we could handle all Pmod instances with a single bitstream?

© Copyright 2018 Xilinx

XILINX.

base Overlay MicroBlaze IO Processor (IOP)



The A9 controls the MicroBlaze processors

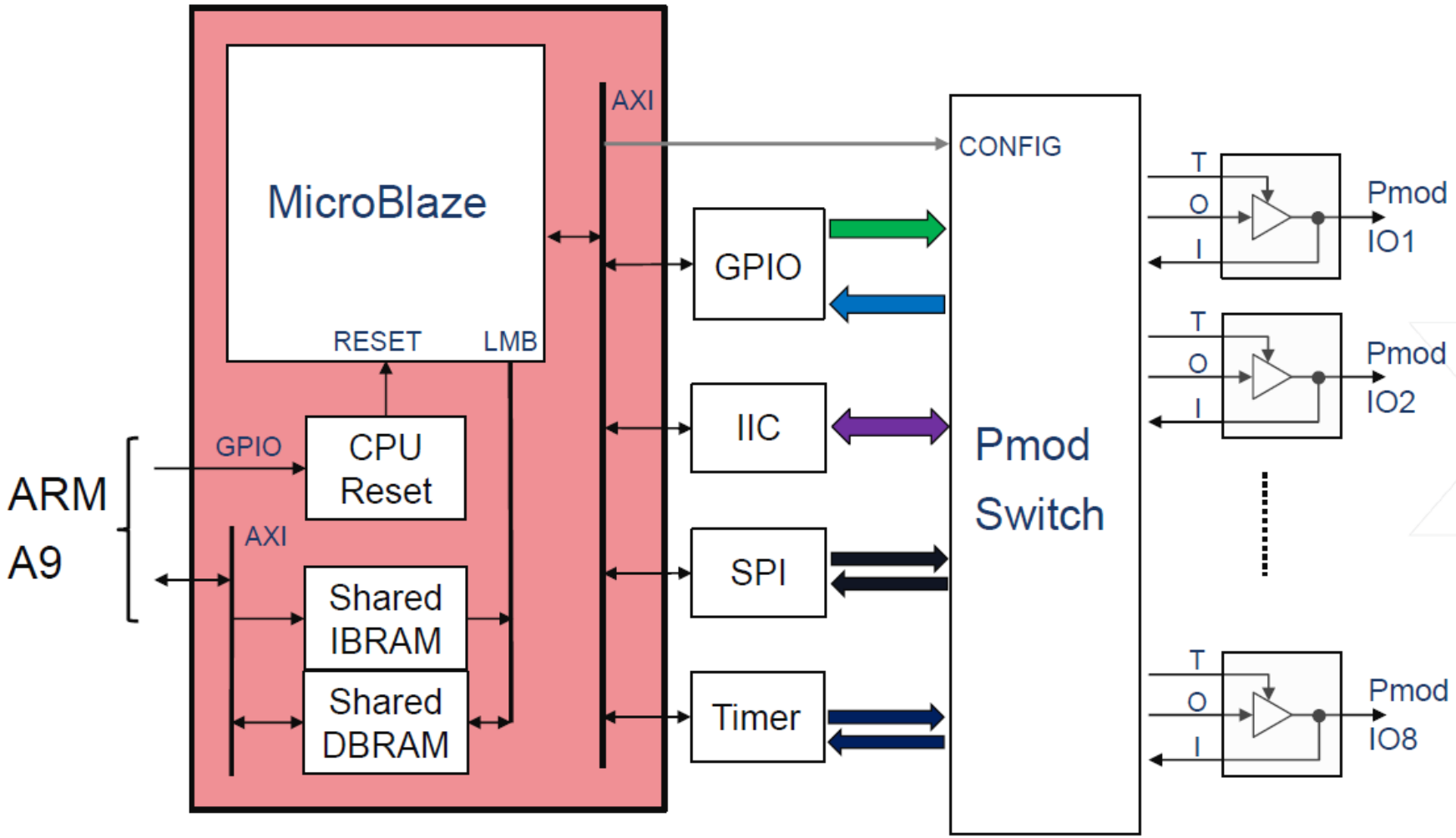
The A9 delivers the binary code to be executed via dual-ported Instruction BRAMs

The A9 exchange data and control (**commands/status**) with the MicroBlazes via dual-ported Data BRAMs

© Copyright 2018 Xilinx

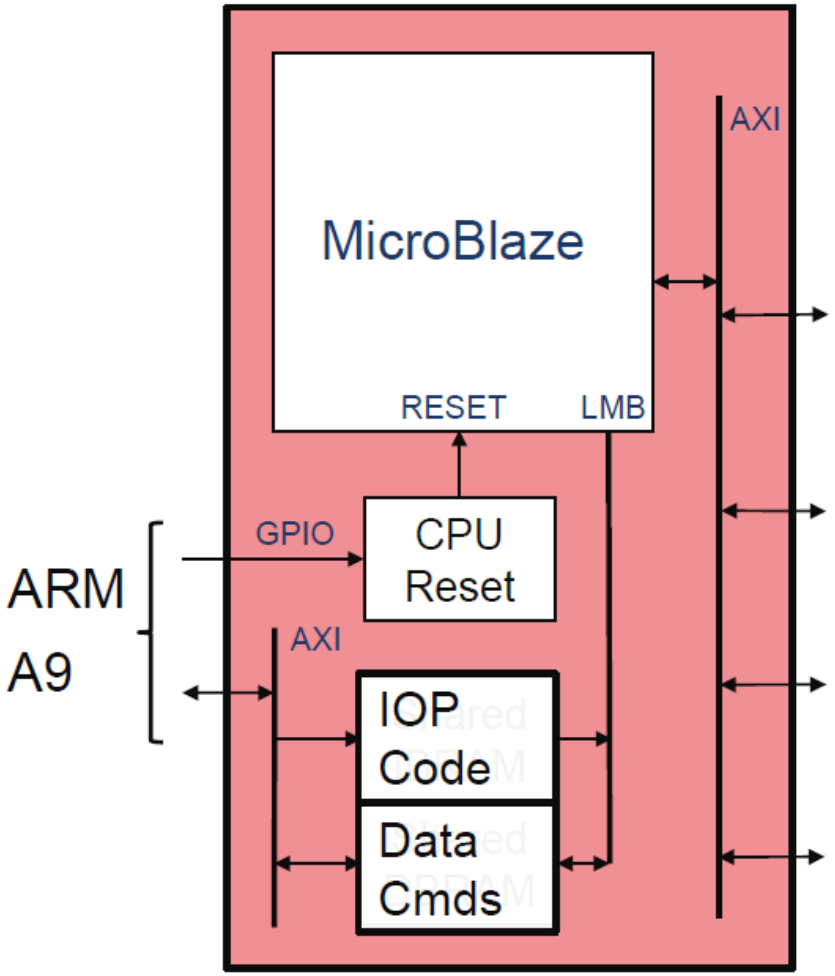
XILINX.

Configure IO Processor for Pmod



© Copyright 2018 Xilinx

Soft Processor Subsystem (SPS)



The A9 controls the MicroBlaze processors

The A9 delivers the binary code to be executed via dual-ported Instruction BRAMs

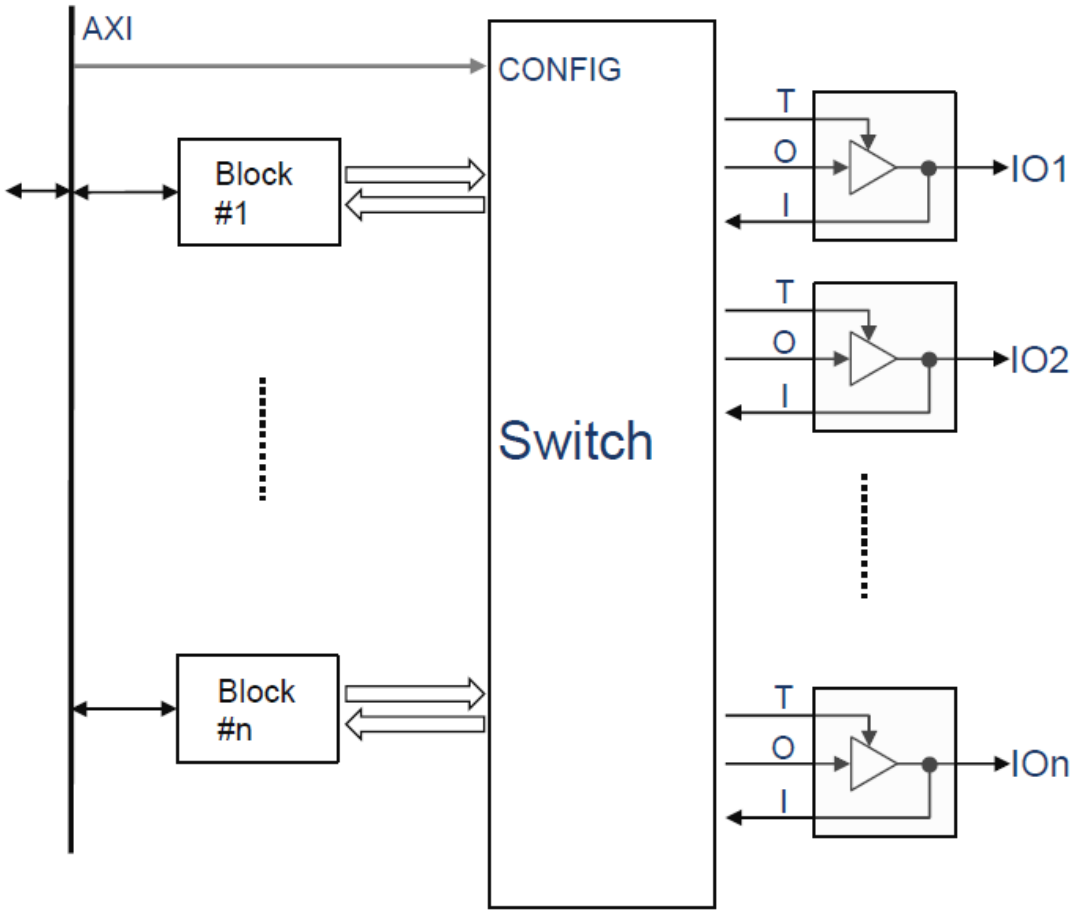
The A9 exchange data and control (commands/status) with the MicroBlazes via dual-ported Data BRAMs

Multiple SPS units can be used to control subsystems in the PL fabric: e.g. IO interfaces, internal interfaces, data-path units and instrumentation

SPS can also be used for distributed processing

© Copyright 2018 Xilinx

IO Switch (IOS)



The IO Switch
can be re-used to control
any external interface

© Copyright 2018 Xilinx

XILINX.

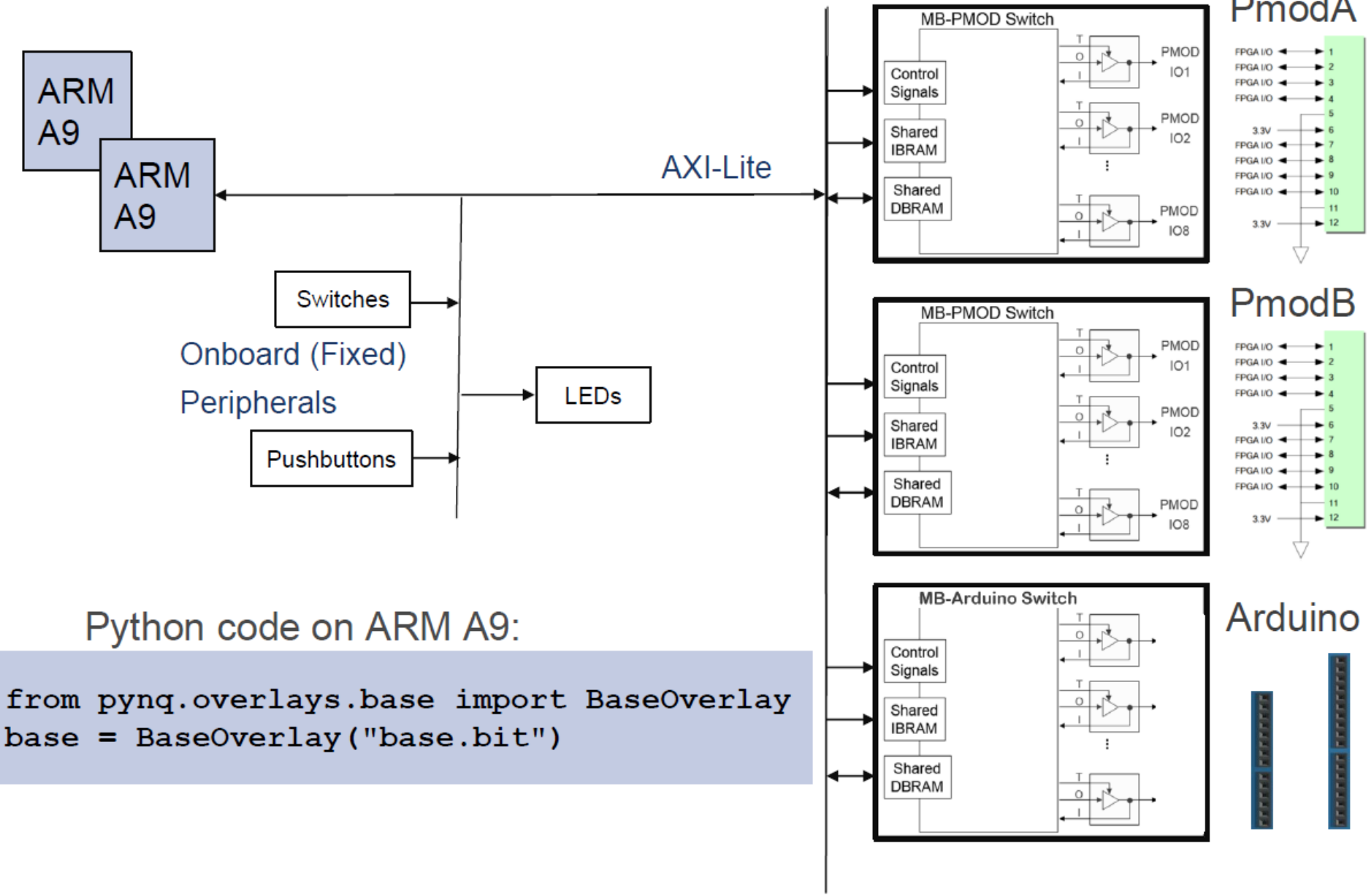
The Python programmer's view

XILINX

PYNQ

© Copyright 2018 Xilinx

Load *base* overlay on PL



Python code on ARM A9:

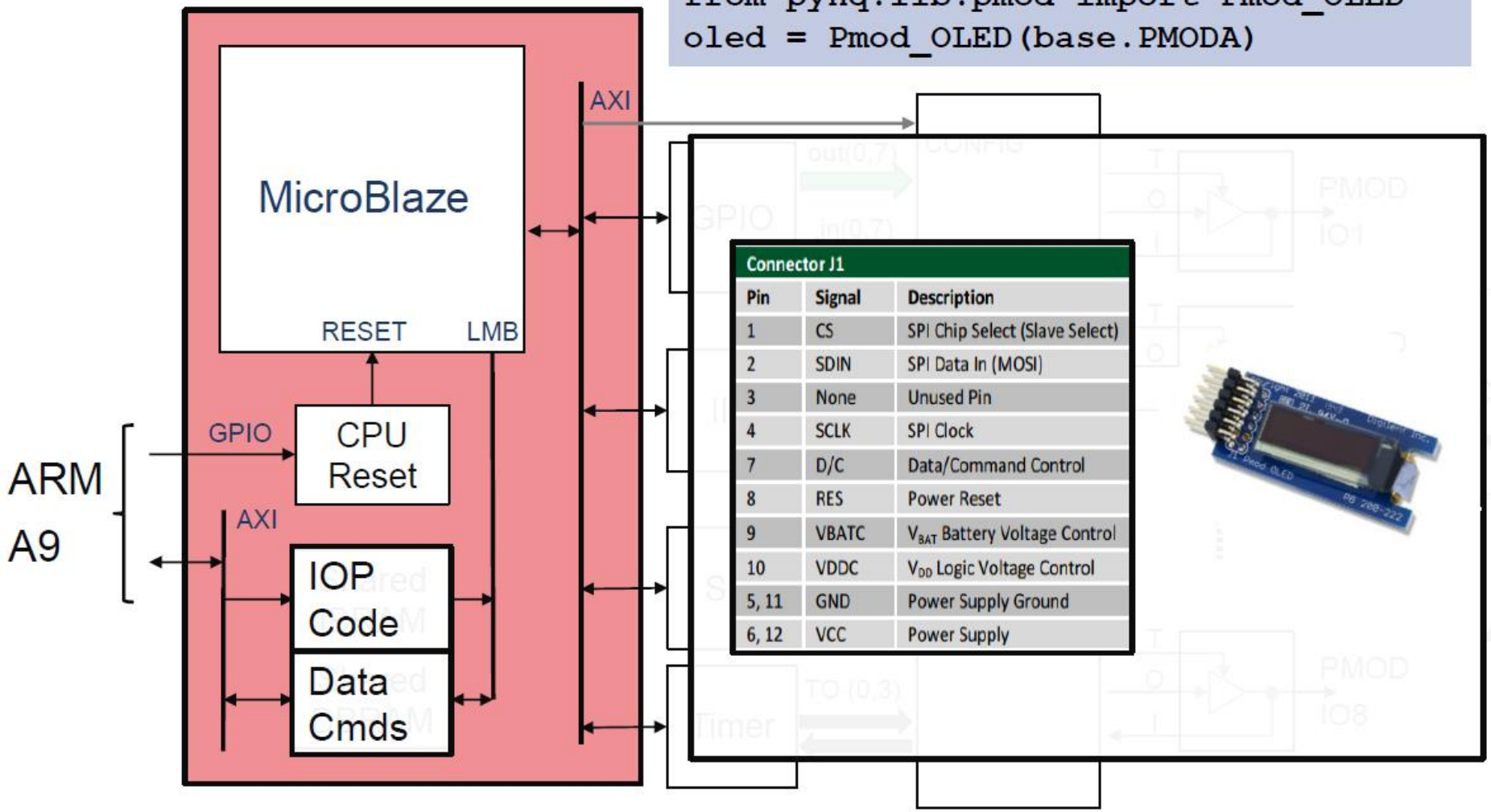
```
from pynq.overlays.base import BaseOverlay
base = BaseOverlay("base.bit")
```


© Copyright 2018 Xilinx

XILINX.

Configure IO Processor

```
from pynq.lib.pmod import Pmod_OLED
oled = Pmod_OLED(base.PMODA)
```



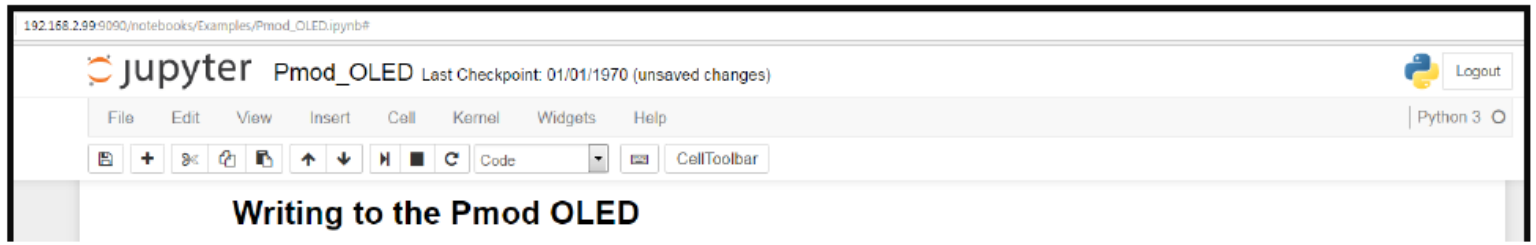
Connector J1		
Pin	**Signal**	**Description**
1	CS	SPI Chip Select (Slave Select)
2	SDIN	SPI Data In (MOSI)
3	None	Unused Pin
4	SCLK	SPI Clock
7	D/C	Data/Command Control
8	RES	Power Reset
9	VBATC	V_{BAT} Battery Voltage Control
10	VDDC	V_{DD} Logic Voltage Control
5, 11	GND	Power Supply Ground
6, 12	VCC	Power Supply


```
oled.write("1 2 3 4 5 6")
```


© Copyright 2018 Xilinx

XILINX.

Jupyter Notebook



In [1]:
```
from pynq.overlays.base import BaseOverlay
base = BaseOverlay("base.bit")
```

In [2]:
```
from pynq.lib.pmod import Pmod_OLED
oled = Pmod_OLED(base.PMODA)
```

In [3]:
```
oled.write("1 2 3 4 5 6")
```

5 lines of user code … thanks to Python, FPGA overlays, abstraction & re-use

© Copyright 2018 Xilinx

XILINX.

Lab exercises:

Lab (Choose 1)

- PMOD OLED Example
- PMOD Grove Tmp
- PMOD Grove LEDbar
- PMOD Grove Light

© Copyright 2018 Xilinx

Questions?

XILINX

PYNQ

© Copyright 2018 Xilinx