XILINX
ALL PROGRAMMABLE™

PYNQ™

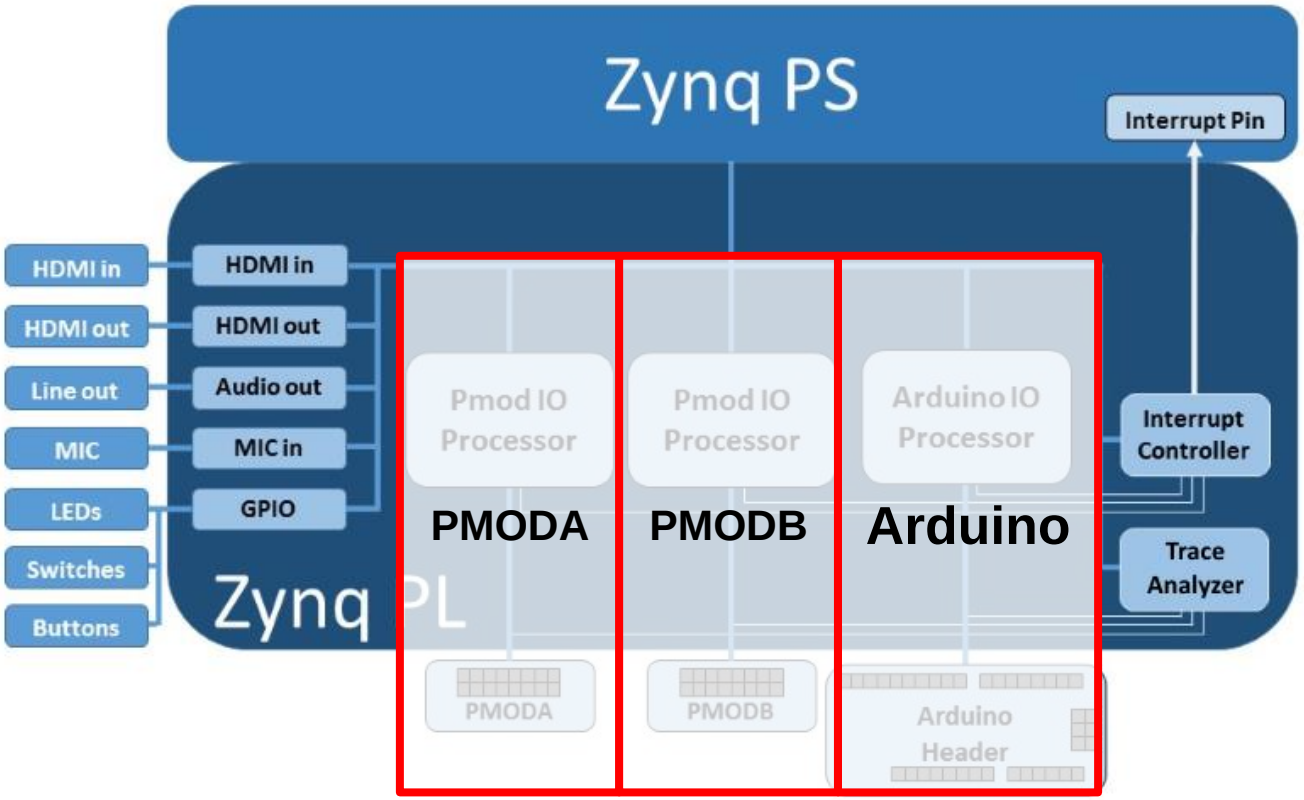
Base Overlay – Microblazes & PYNQ

XILINX
ALL PROGRAMMABLE™

Agenda

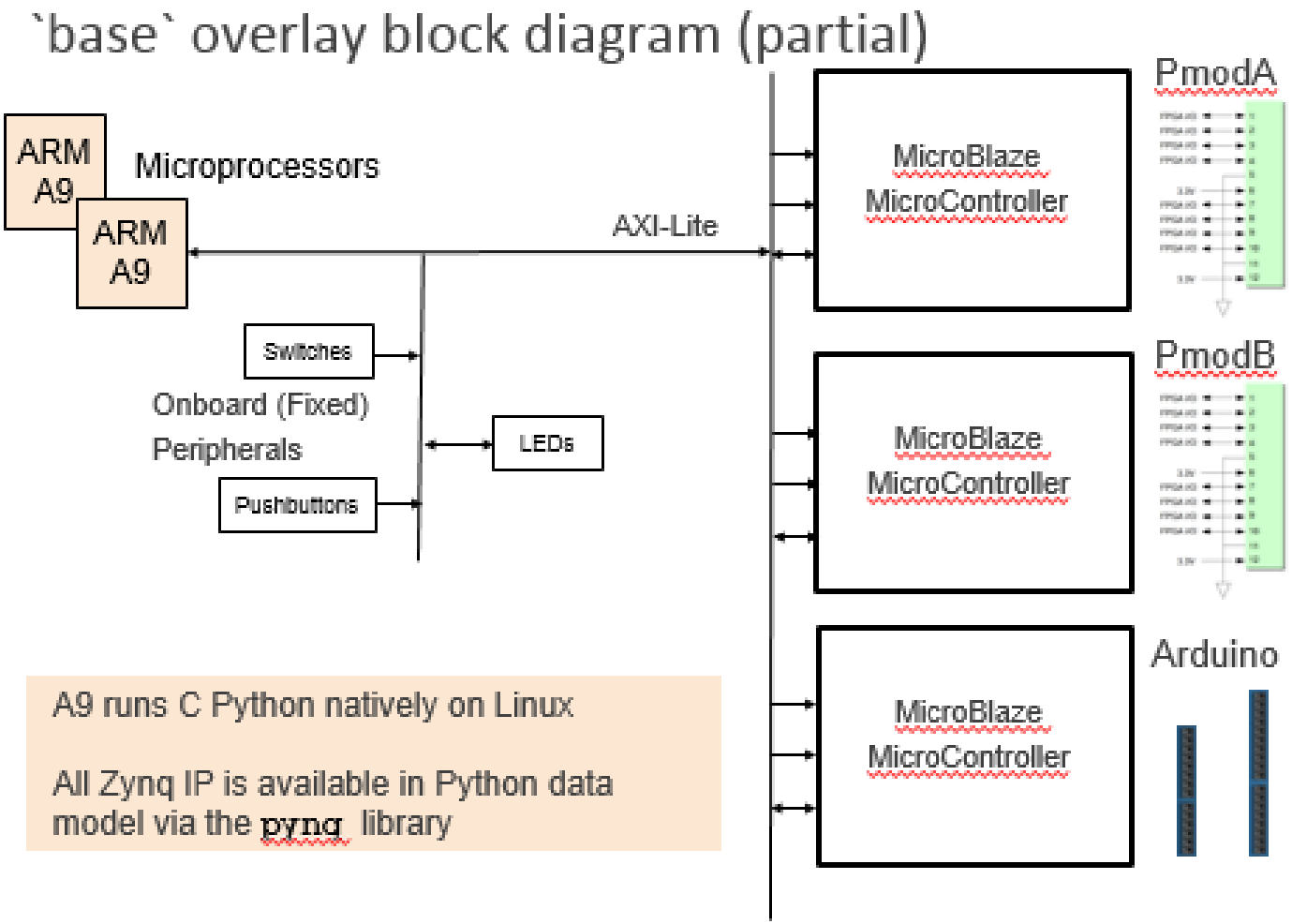
- Base Overlay Microblazes
- PYNQ Microblaze Compilation
- Python to C to Microblaze C

Base Overlay Microblazes

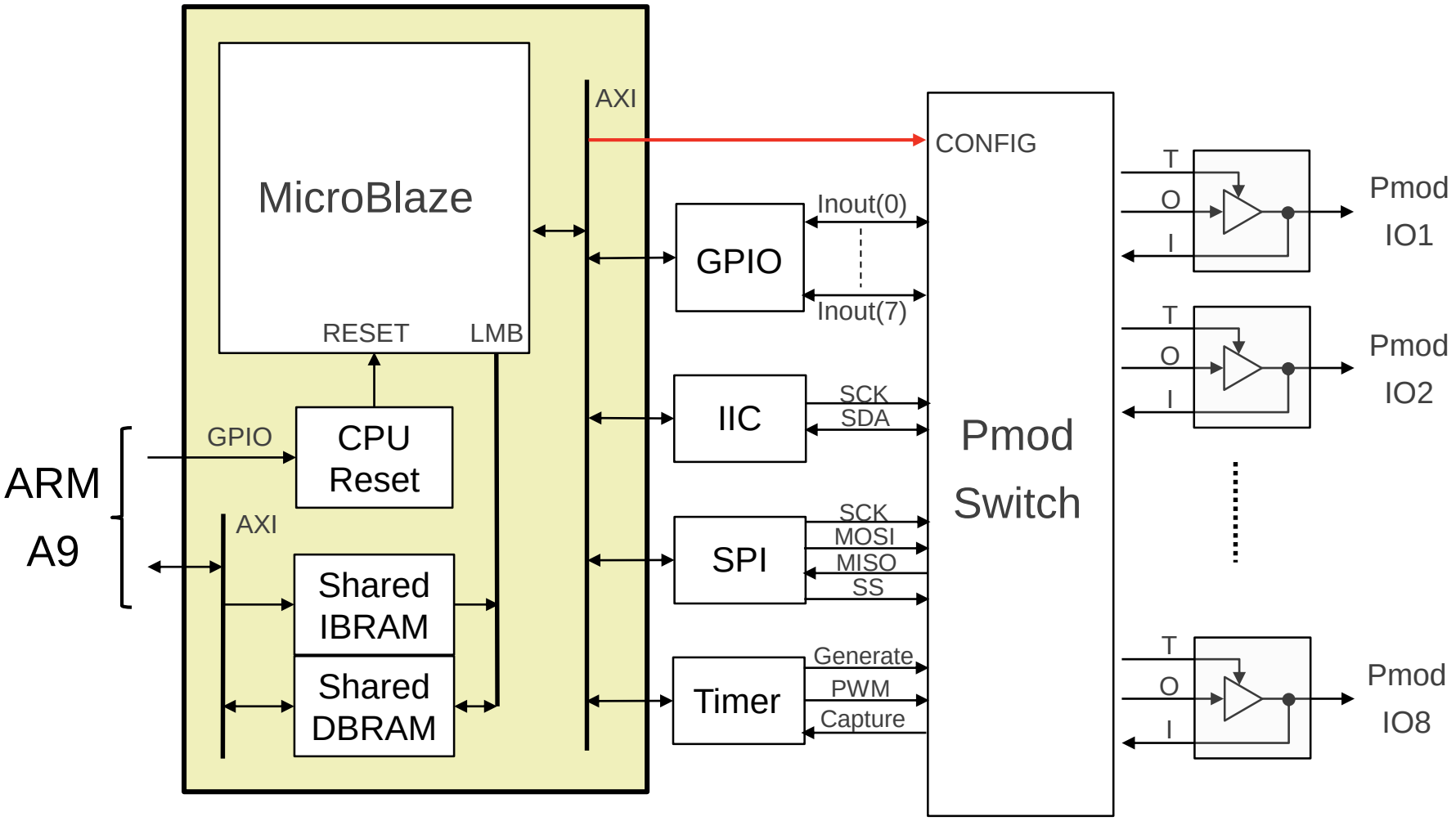


x2 PMOD Microblazes, x1 Arduino Microblaze

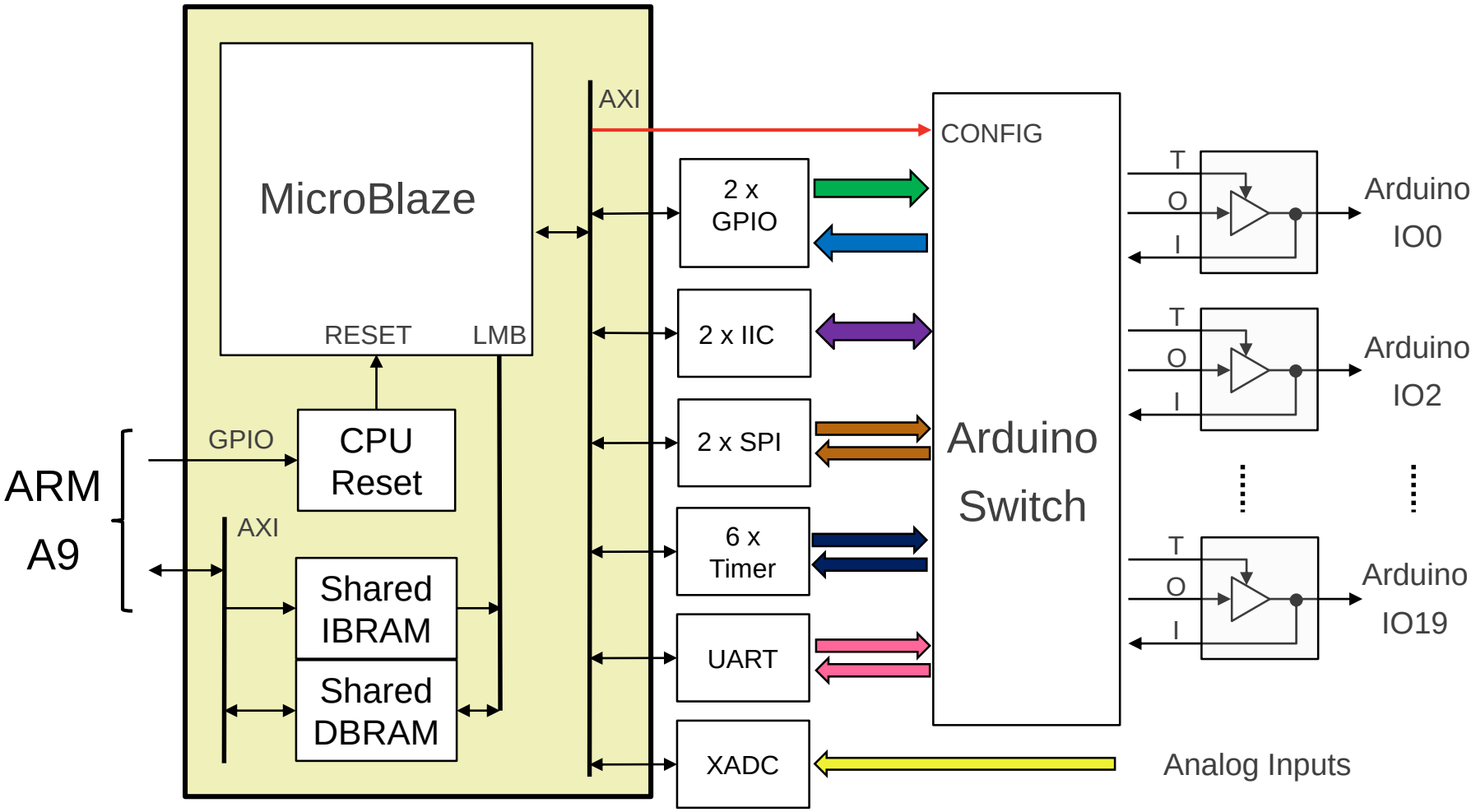
PYNQ Microblazes



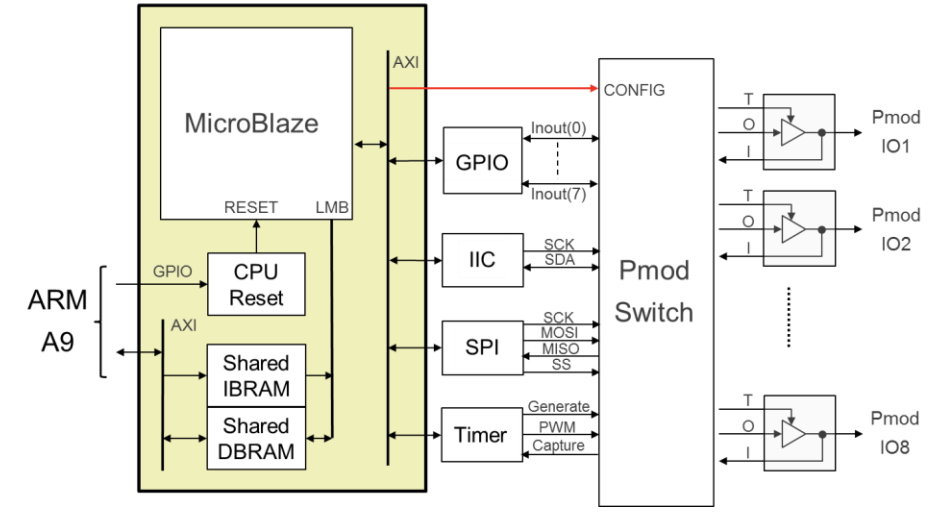
Pmod IO Processor

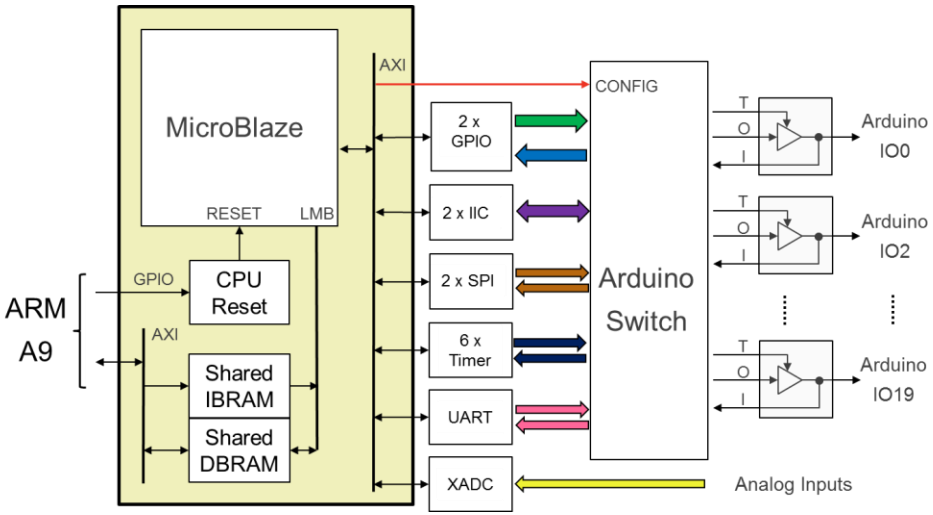


Arduino IO Processor



Different SoCs, Same Processor Subsystem

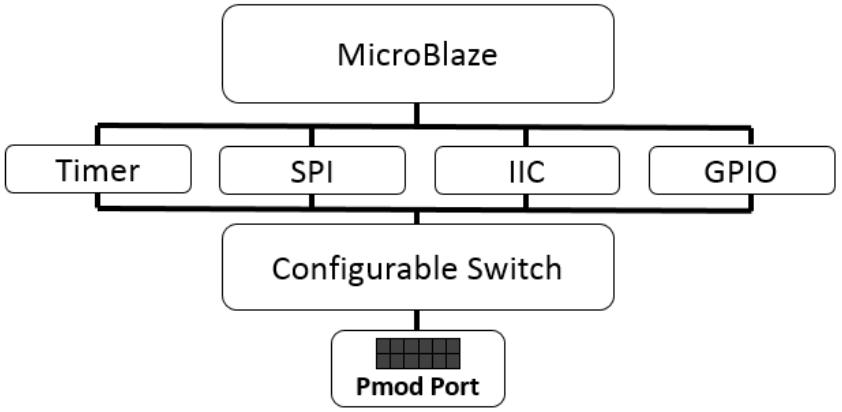




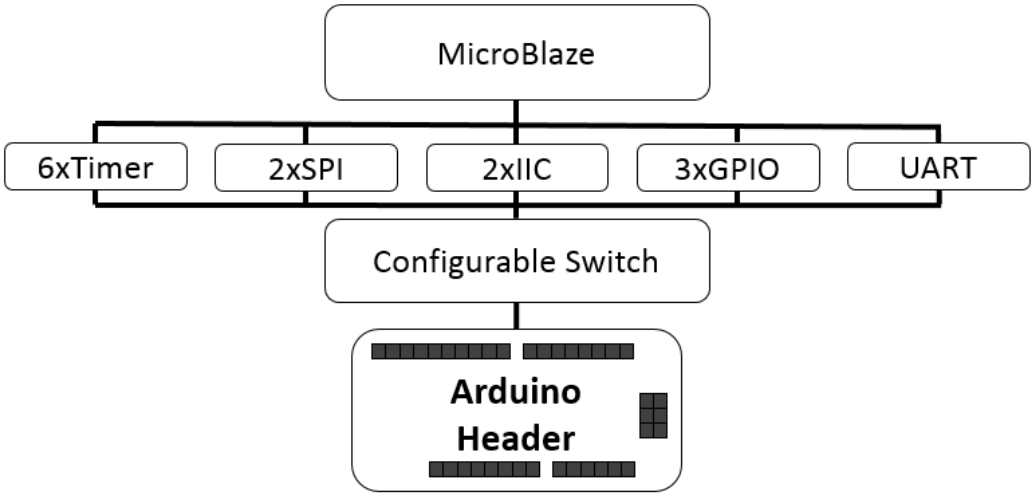
- Microblazes are identical
- The Peripherals are different

Microblaze IO Switch

- Programmable I/O
  - Any controller can talk to any pin



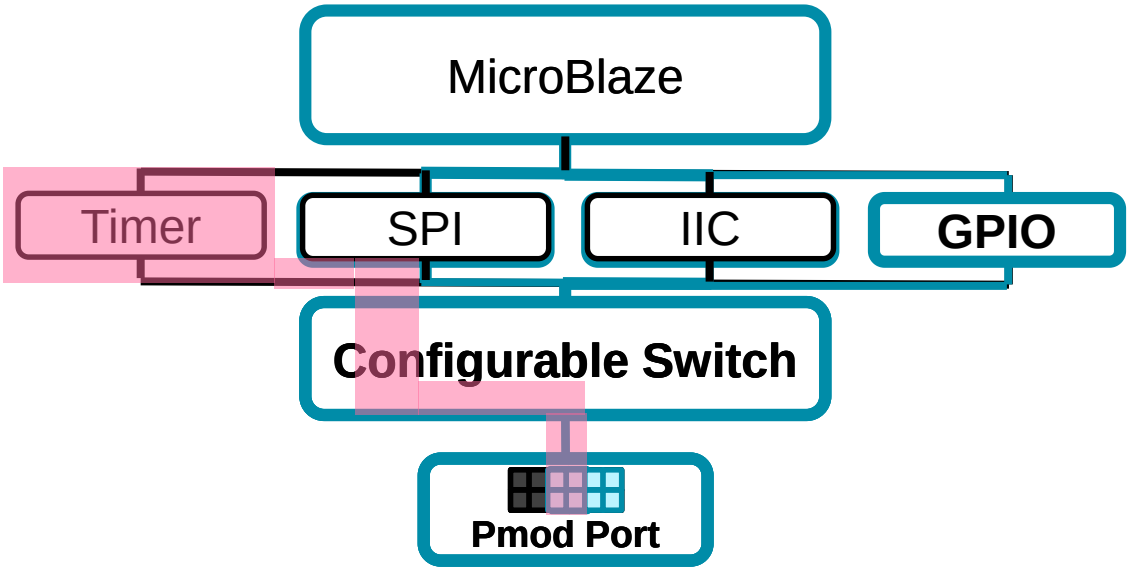
Pmod IOP



Arduino IOP

Microblaze IO Switch

- Allows peripherals with different interfaces to be used in the same overlay without needing a new FPGA design

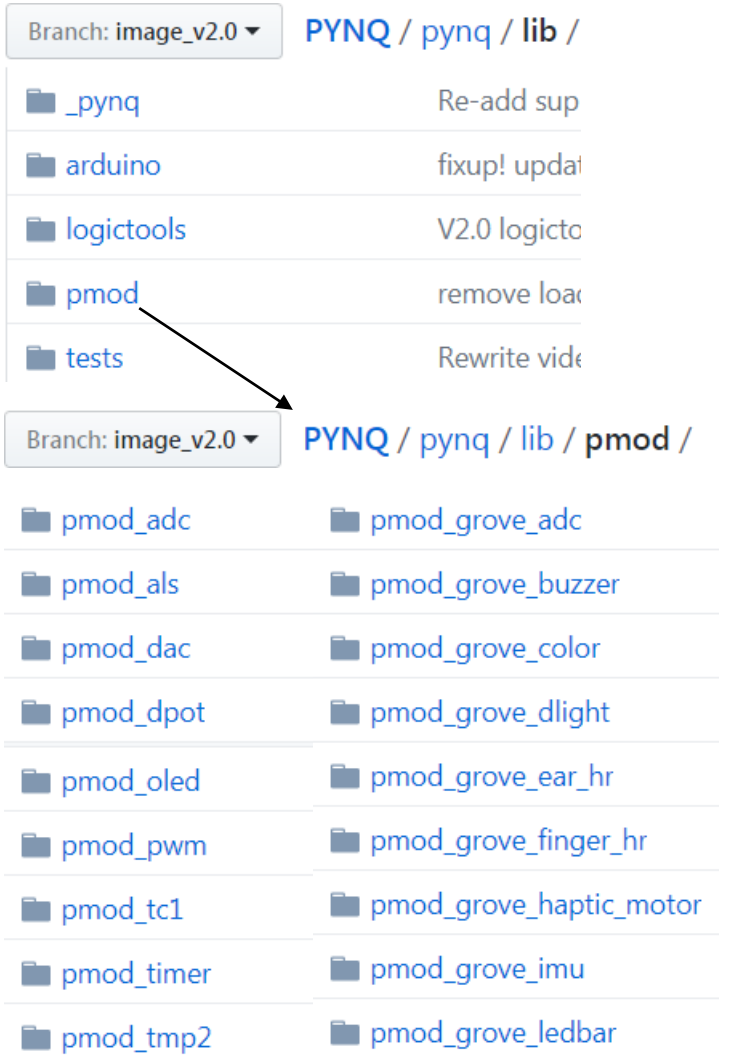


Pythonic Control of the Switch to define pin functionality

Microblaze Software

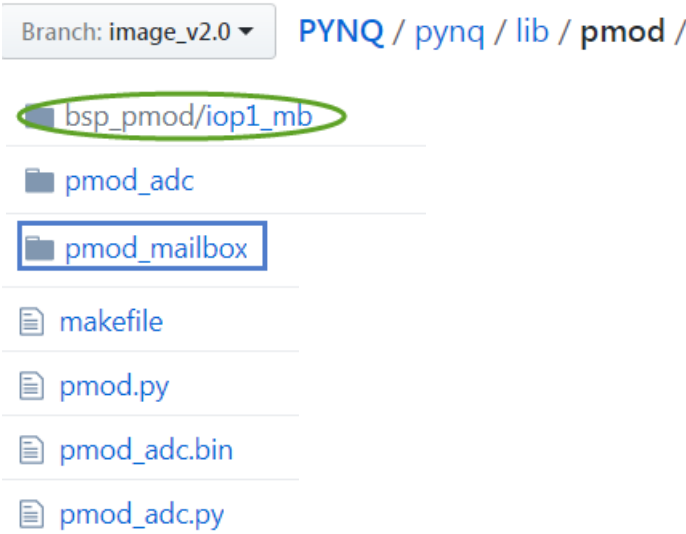
Example projects (GitHub)

- Source code and projects available on GitHub for a range of peripherals
  - Grove and Pmod
  - Some Arduino shield examples
  - Can be used as starting point for a new project
- API available
  - IIC, SPI, GPIO, Configurable switch
    - Simple low level API's; Read( ), Write( )
  - pmod.c, pmod.h; arduino.c, arduino.h
- *Make* flow to build IOP projects available



Software directory (GitHub)

- Various software projects grouped according to interface and overlay related reside under ./pynq/lib/
  - Arduino, logictools, Pmod
- Under each group reside related software projects, bsp, makefile, bin (binary executable files), and Python class file
- mailbox
  - Enables data and command/status exchanges between AP and IOP



Pythonic Microblaze Programming

In [3]:
```
%%microblaze base.ARDUINO lower_case
#include <unistd.h>
#include <ctype.h>

int main() {
        char buf[1024];
        int bytes;
        int remain = 0;
        while (1) {
                bytes = read(STDIN_FILENO, buf, 1024);
                for (int i = 0; i < bytes; ++i) {
                        buf[i] = (char)tolower(buf[i]);
                }
                remain = bytes;
                while (remain > 0) {
                        remain -= write(STDOUT_FILENO, buf + bytes - remain, rema
                }
        }
}
```

In [4]:
```
test_string = 'HELLO, WORLD!'.encode()
lower_case.stream.write(test_string)
result = None
while not result:
    result = lower_case.stream.read()
print(result.decode())
```

```
hello, world!
```

Cell magic
%%microblaze

Builds executable

stdin | stdout

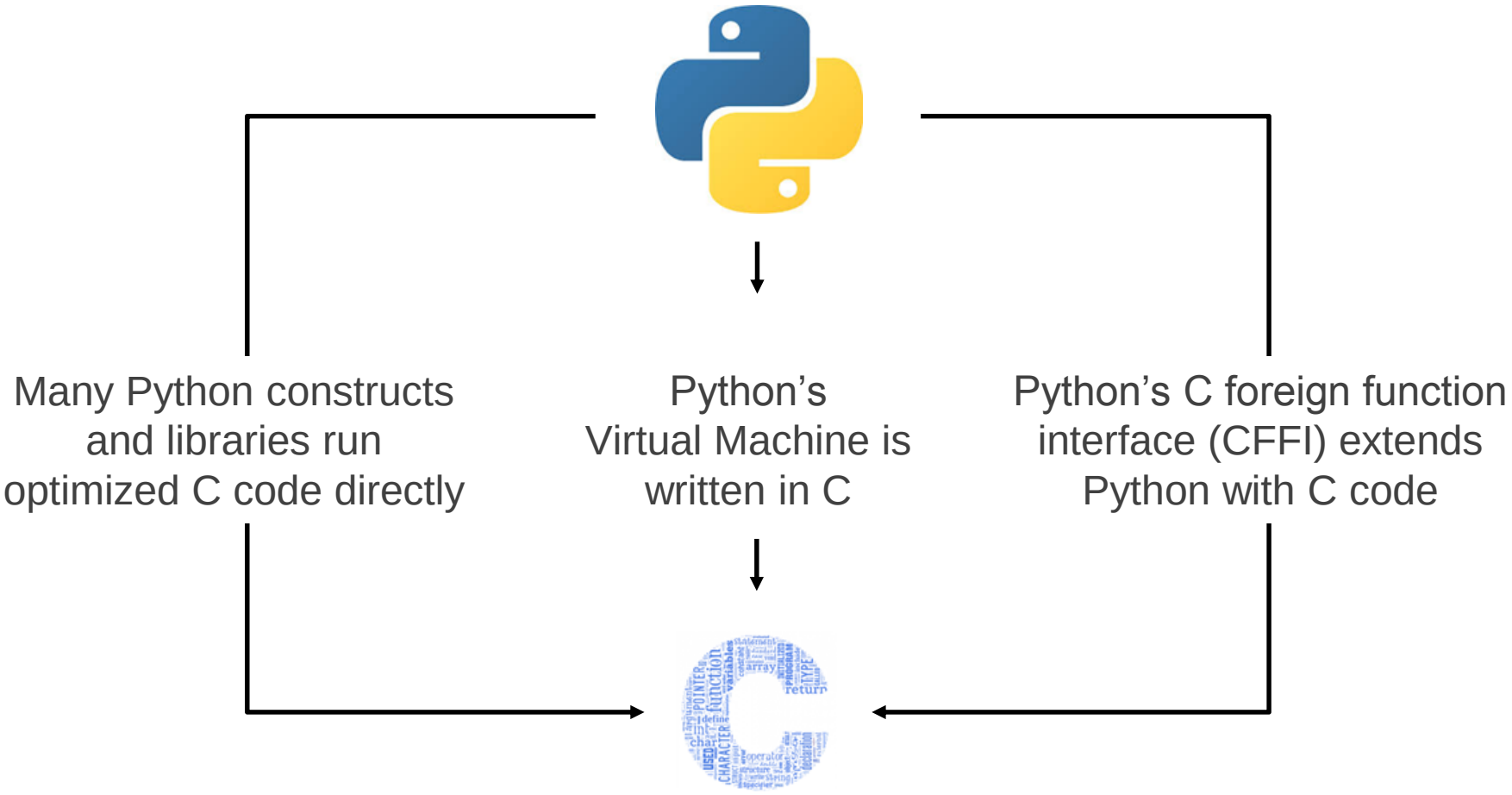
Pipes for Microblaze printf

Conclusions
Python, ARM C, Microblaze C

IOP as standalone processor

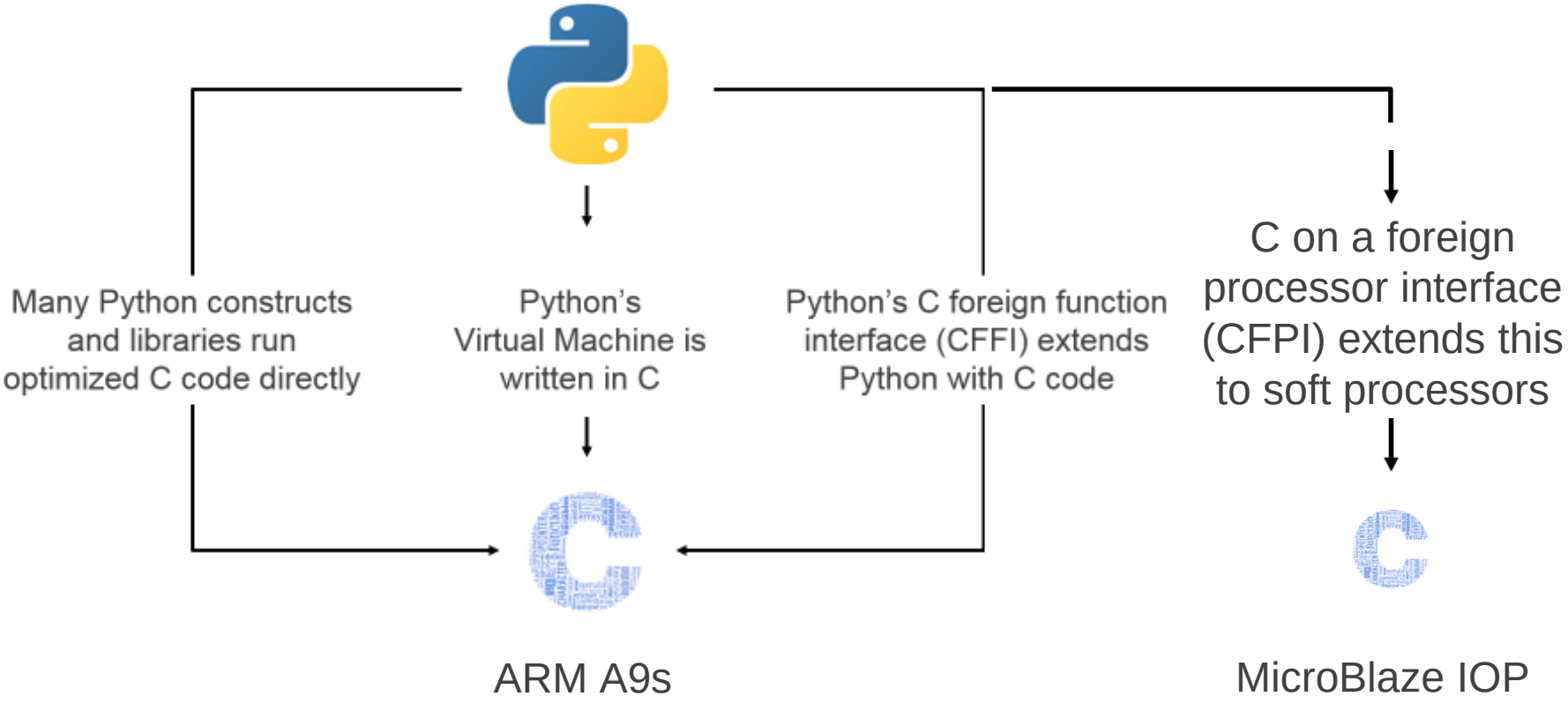
- Dual core ARM Cortex A9 vs MicroBlaze
  - 32-bit ARM Cortex A9: ~650 MHz
  - 32-bit MicroBlaze: ~100MHz
- Several IOPs can execute in parallel
  - ~5% device utilization PYNQ-Z1
- MicroBlaze can be dedicated to real-time applications
- MicroBlaze can offload background processing tasks

Python and C: a symbiotic relationship



Code Pythonically; exploit libraries; extend with C if needed

PYNQ's CFPI enables multiple soft processors



Summary: Soft processors in the fabric as offload engines for real-time performance