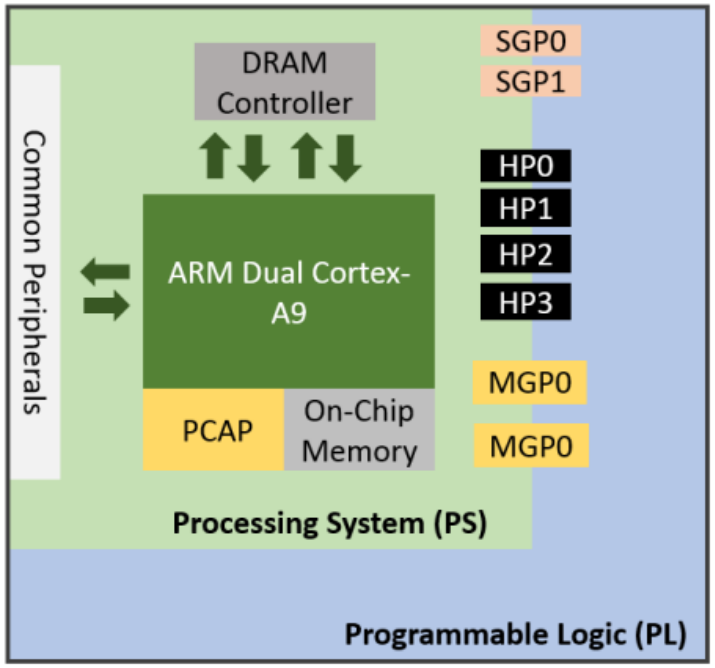
Zynq, Embedded Linux, and Tutorial Summary

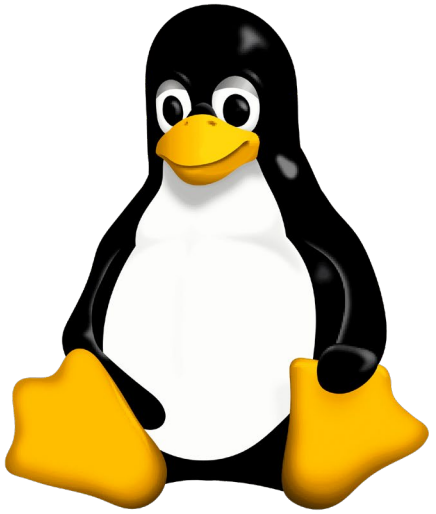
Erwin Setiawan

Zynq and Embedded Linux

Zynq and Linux

- Zynq mengunakan ARM Cortex-A series, sehingga cukup powerful untuk menjalankan Linux OS.





What We Need?

- Xilinx PetaLinux Tools **(the hard way), or**
  - Melakukan konfigurasi sendiri dari source code Linux.
  - Melakukan compile dan build kernel sendiri.
  - Develompment kernel module.
- Pre-built image **(the easy way).**
  - Banyak board Zynq sudah menyediakan image yang siap pakai (sudah di compile) baik **official** ataupun **third party**.
  - Dengan cara ini kita tidak perlu melakukan build sendiri.

PetaLinux Requirement

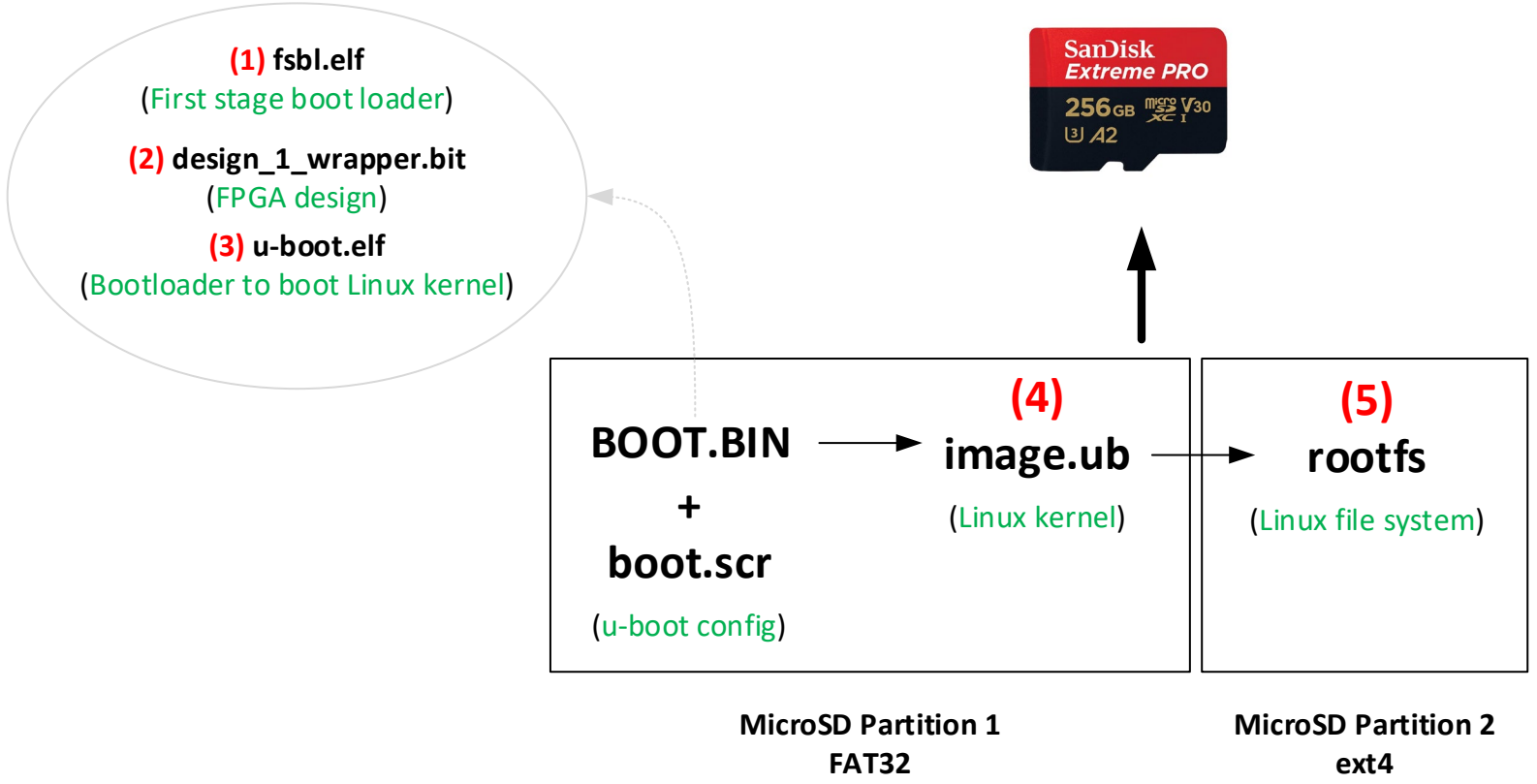
- The PetaLinux tools installation requirements are:
  - Minimum workstation requirements:
    - 8 GB RAM (recommended minimum for AMD tools)
    - 2 GHz CPU clock or equivalent (minimum of eight cores)
    - **100 GB free HDD space**
    - Supported OS:
      - Completely removed RHEL and CENTOS to align with upstream Yocto.
      - **Ubuntu Desktop/Server 18.04.1 LTS, 18.04.2 LTS, 18.04.3 LTS, 18.04.4 LTS, 18.04.5 LTS, 18.04.06 LTS, 20.04.2 LTS, 20.04.3 LTS, 20.04.4 LTS, 20.04.5 LTS(64-bit),20.04.6 LTS, 22.04 LTS, 22.04.1 LTS and 22.04.2 LTS (can’t use Windows)**
      - OpenSuse Leap 15.3 and 15.4
      - AlmaLinux 8.7 and 9.1

PetaLinux Flow

- Build FPGA design -> menggunakan **Vivado** -> menghasilkan file **bitstream (file .bit)** dan **.HDF** atau **.XSA**.
- Create PetaLinux project -> based on **HDF** or **HSA**.
- Configure and build PetaLinux.
- Build output:
  - BOOT.BIN
  - boot.scr
  - image.ub
  - rootfs.tar.gz

PetaLinux Flow (cont.)

- MicroSD Partition and BOOT process



Pre-built Image Flow

- Download image
- Write to MicroSD using Win32DiskImager or Balena Etcher

PYNQ 3.0.1 for Digilent Zybo (2023 update)

Inspired by the thread PYNQ 2.7 forum for Zybo Z7, this repo provides an attempt to port PYNQ for the Zybo boards (including old ones). The goal of this project is quite minimalistic (no complicated overlays, only simple GPIO cores to demostrate that Linux & PYNQ tools are working as expected). Unlike other tutorials, this implementation provides support of newer software distribution (PYNQ v3.0.1) and the correct ethernet MAC address. I attempted to add complementary remarks and thoughts regarding installation of Xilinx tools on headless (i.e using a text-mode terminal only) Linux-based machine.

Prebuild SD images

Lastest release

Be carefull, there is exists atleast 3 majour versions of Zybo board:

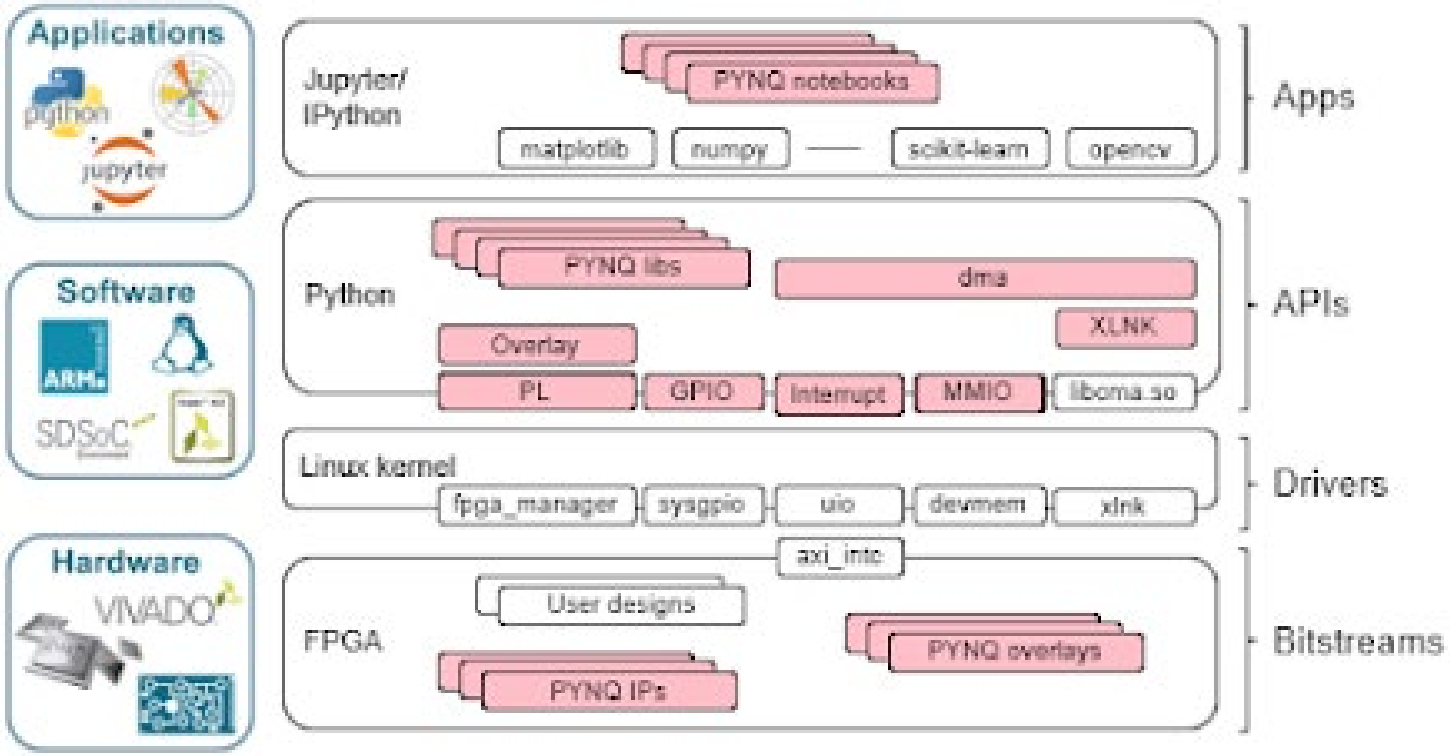
Board	SoC part	Image	MD5
Retired Zybo (with VGA port)	`xc7z010clg400-1`	Zybo-3.0.1-fix-boot-bin-fix-havege.img.xz	`ef7fd9fbcd1dc035c02347687127a44d`
Zybo-Z7-10	`xc7z010clg400-1`	Zybo-Z7-10-3.0.1-fix-havege.img.xz	`35311dc43e11cdf091c199fd2c7a41fe`
Zybo-Z7-20 (2 RGB leds and fan) **NOT TESTED**	`xc7z020clg400-1`	Zybo-Z7-20-3.0.1-fix-havege.img.xz	`b1500e51c65cd65e2eb2d25f555f9640`

Win32DiskImager

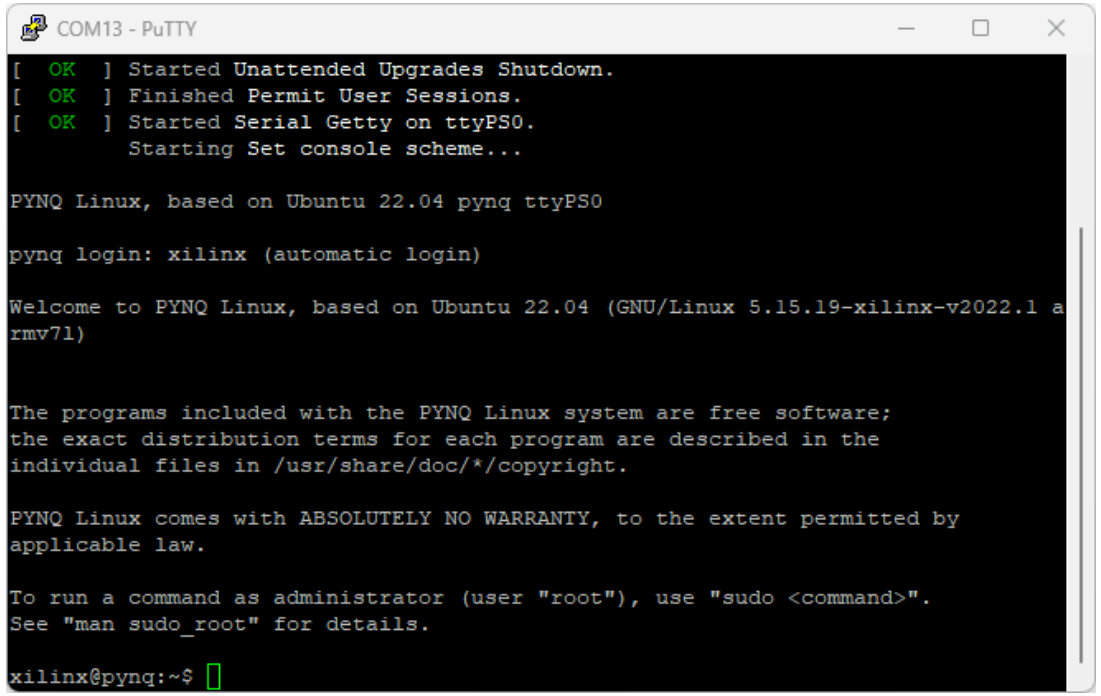
balenaEtcher

Why PYNQ?

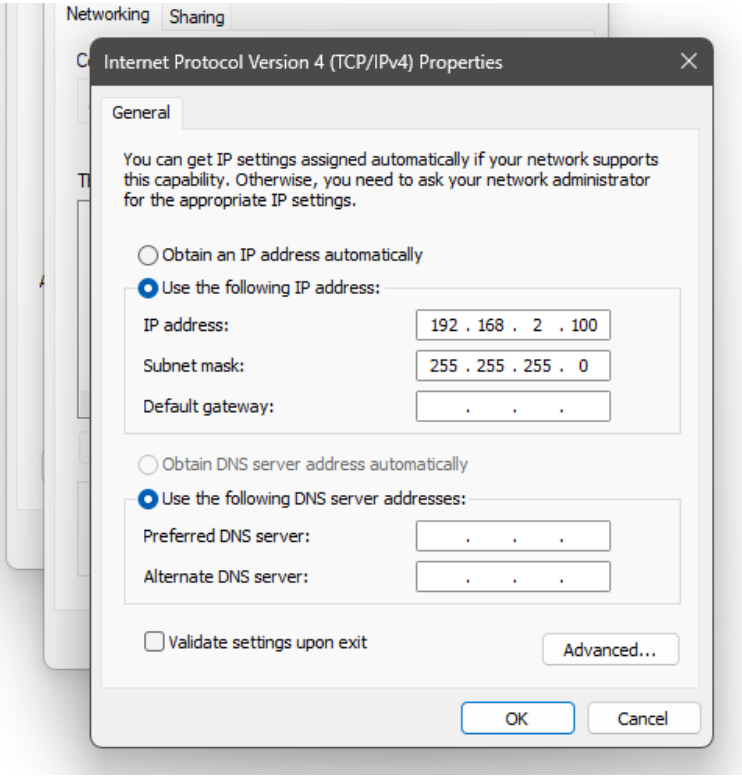
- Pynq berisi library dan kernel module untuk mengakses FPGA dari Linux OS



BOOT Process

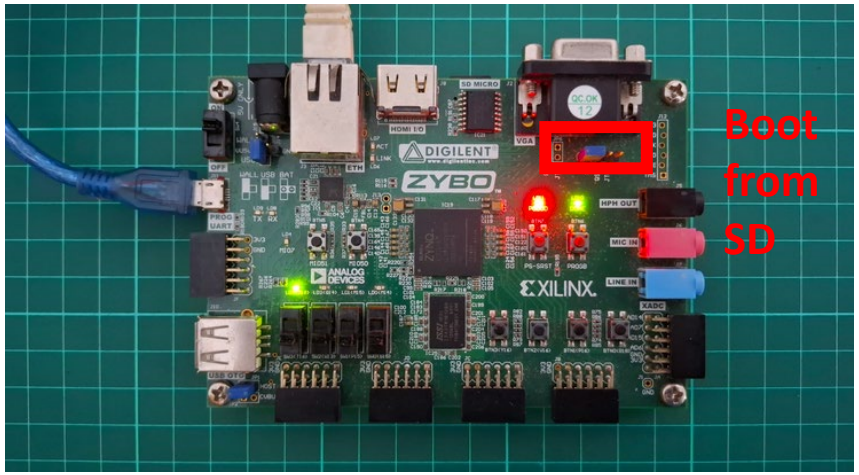


Serial
- **COM port**
- **115200**
- **8 bit**
- **1 stop**
- **No parity**

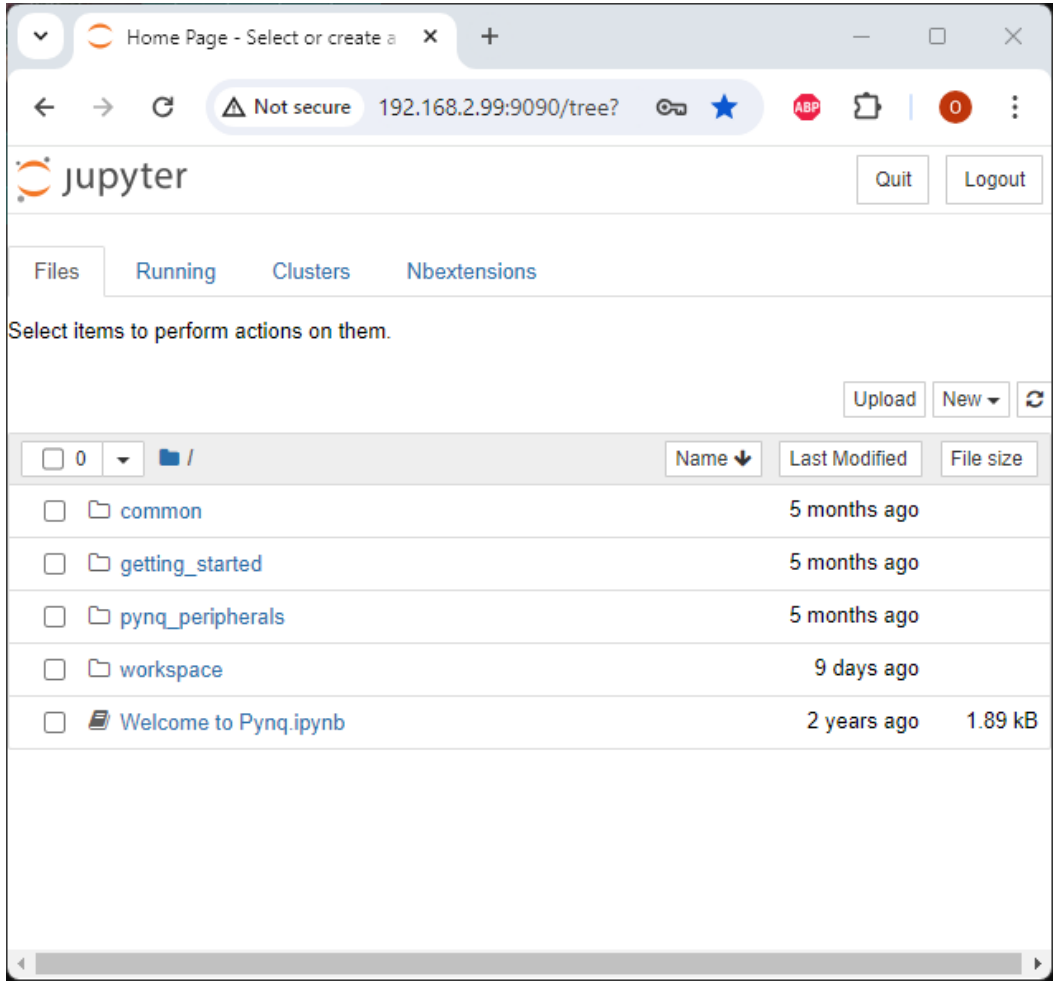


Ethernet:
IP PC

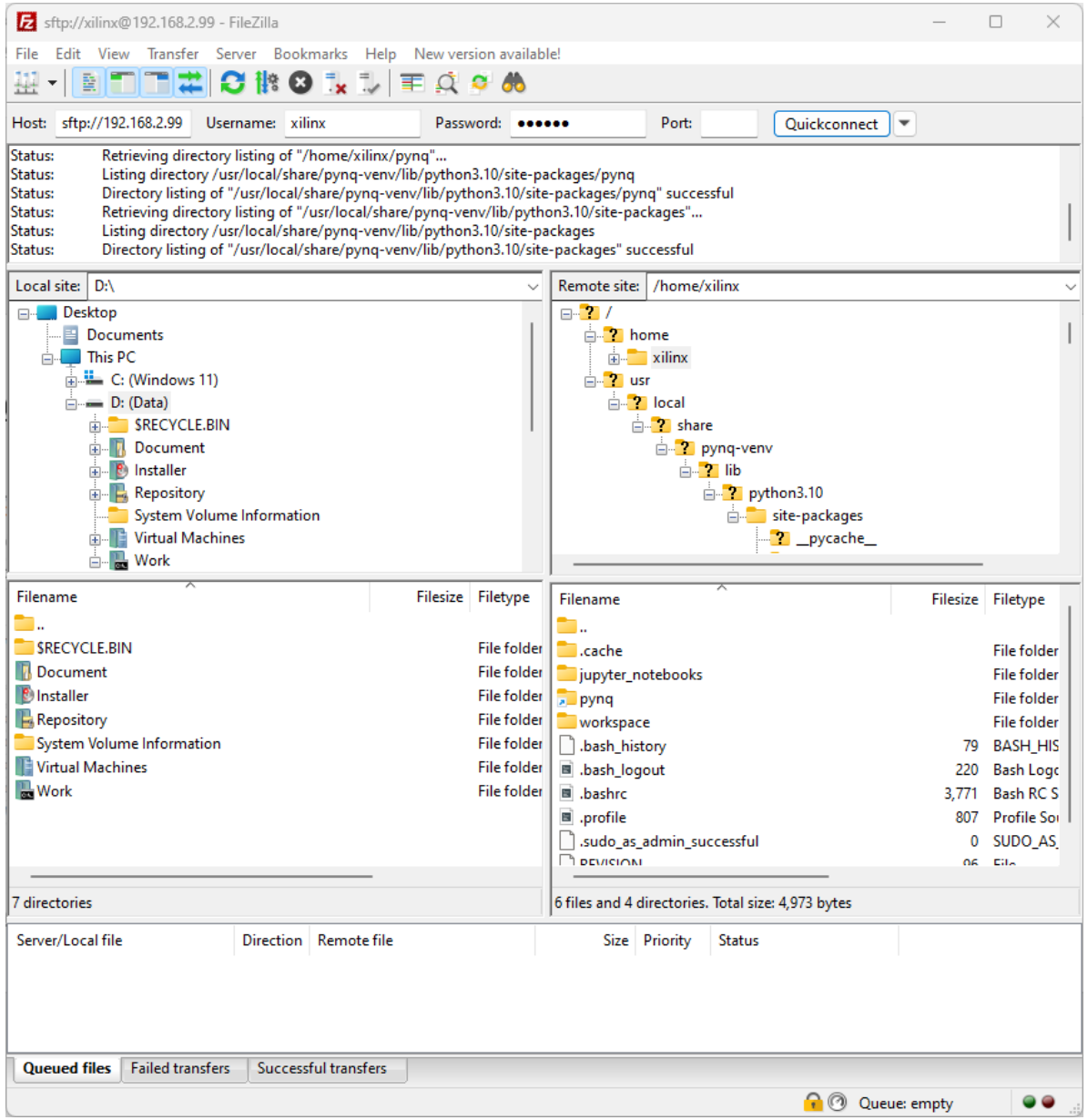
Ethernet: IP default board 192.168.2.99



Access to Board

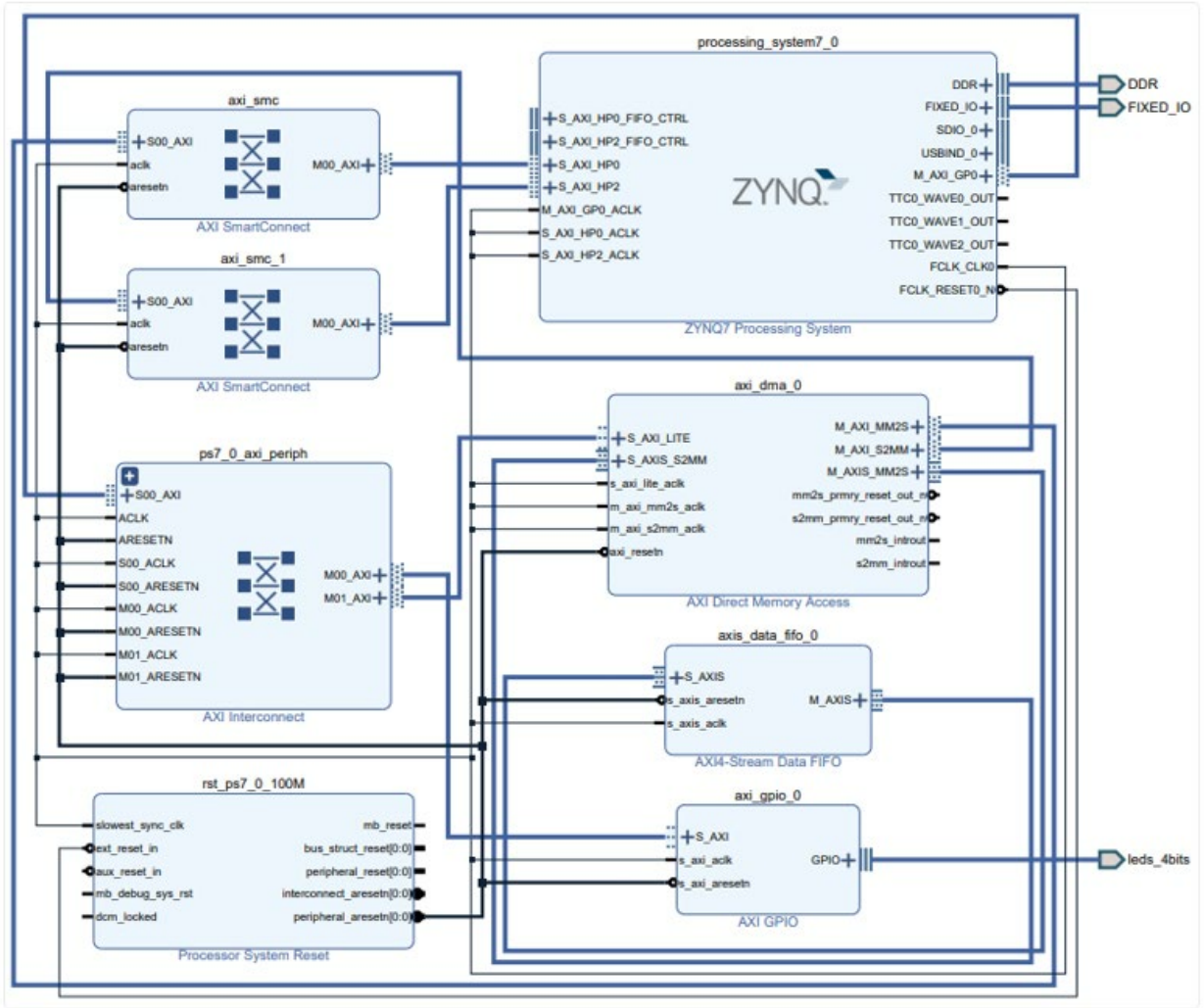


Jupyter Notebook



Transfer file between PC and board using FTP FileZilla

Contoh Design dengan PYNQ



GPIO to LED

GPIO Control

In [1]:
```
from pynq import Overlay
from pynq import MMIO
from pynq import allocate
import numpy as np
```

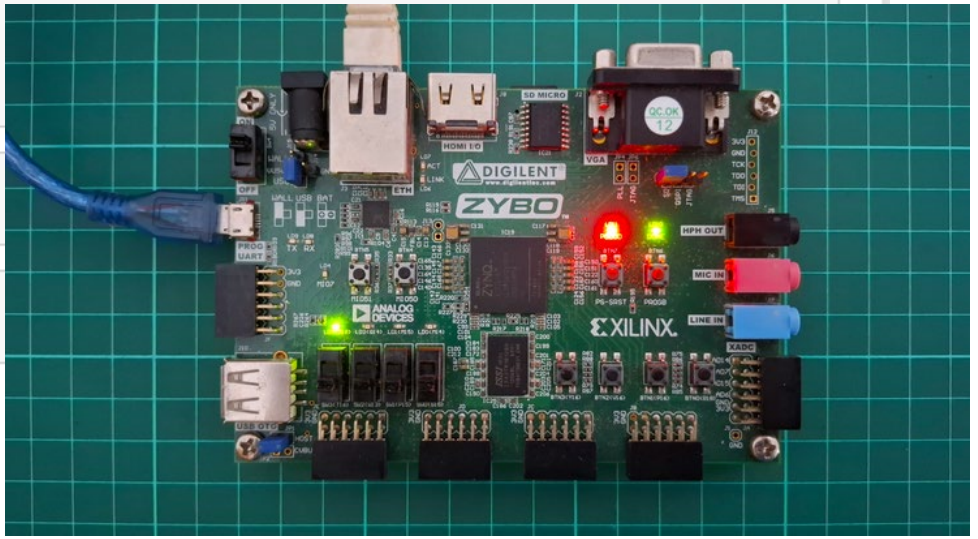
In [2]:
```
# Program bitstream to FPGA
overlay = Overlay('/home/xilinx/workspace/zybo_pynq.bit')
```

In [3]:
```
# Access to memory map of the AXI GPIO
ADDR_BASE = 0x41200000
ADDR_RANGE = 0xFFFF
gpio_obj = MMIO(ADDR_BASE, ADDR_RANGE)
```

In [4]:
```
# Write data 8 to GPIO address 0x0
gpio_obj.write(0x0, 8)
```

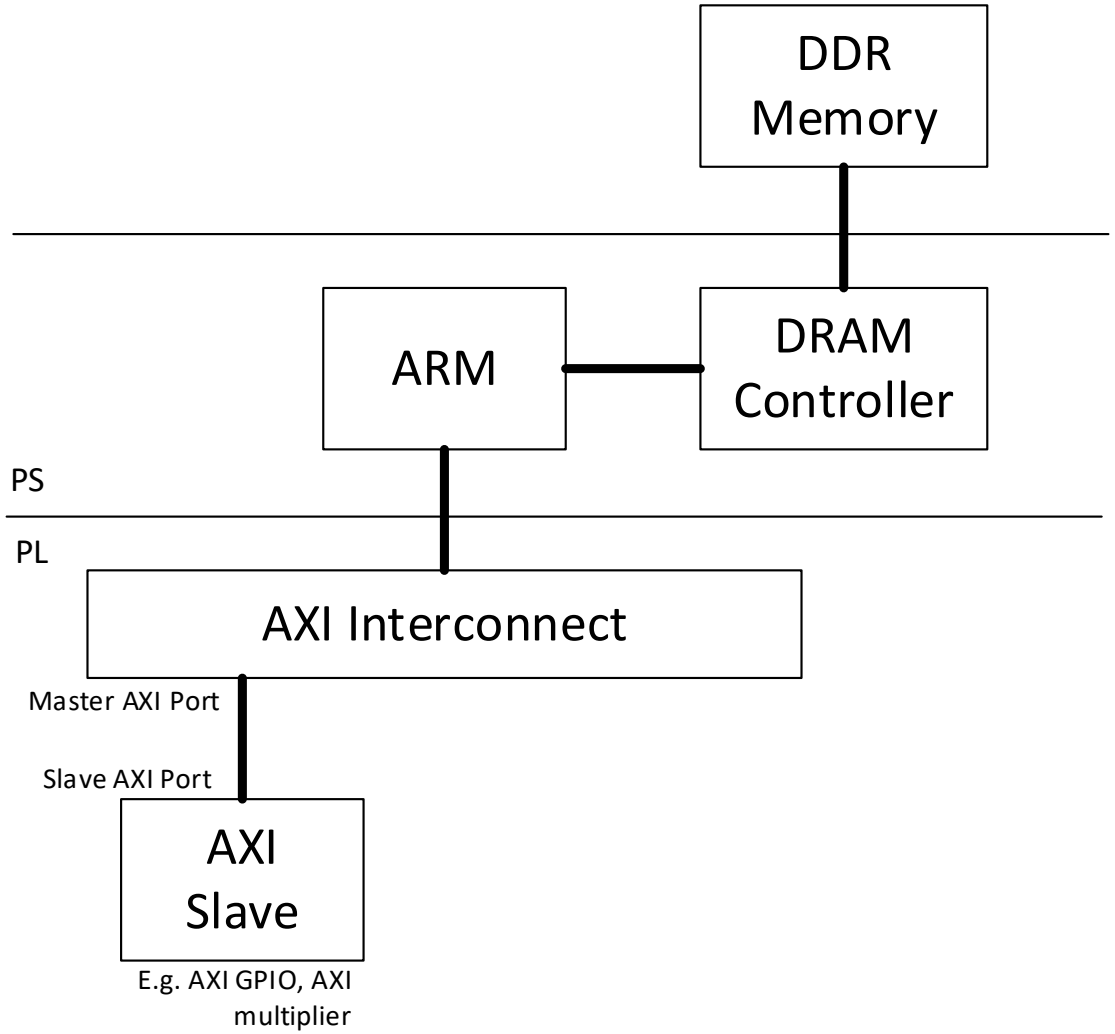
In [5]:
```
# Read data from GPIO address 0x0
gpio_obj.read(0x0)
```

Out[5]: 8



Summary of Zynq HW-SW Design

1. Simple AXI-lite Slave Module

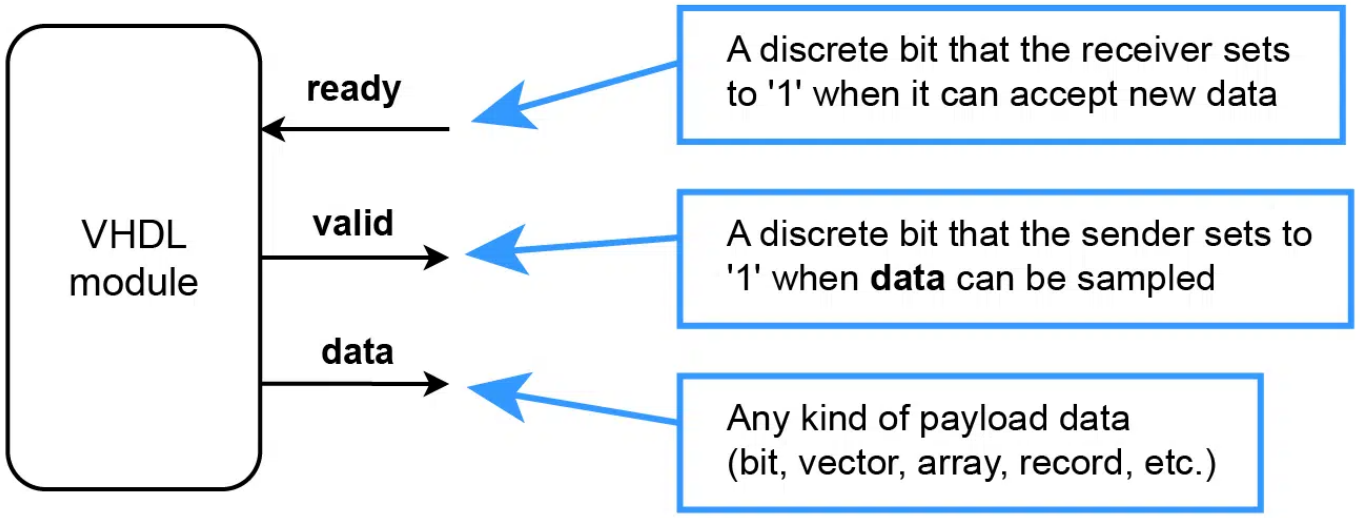


AXI Procotol

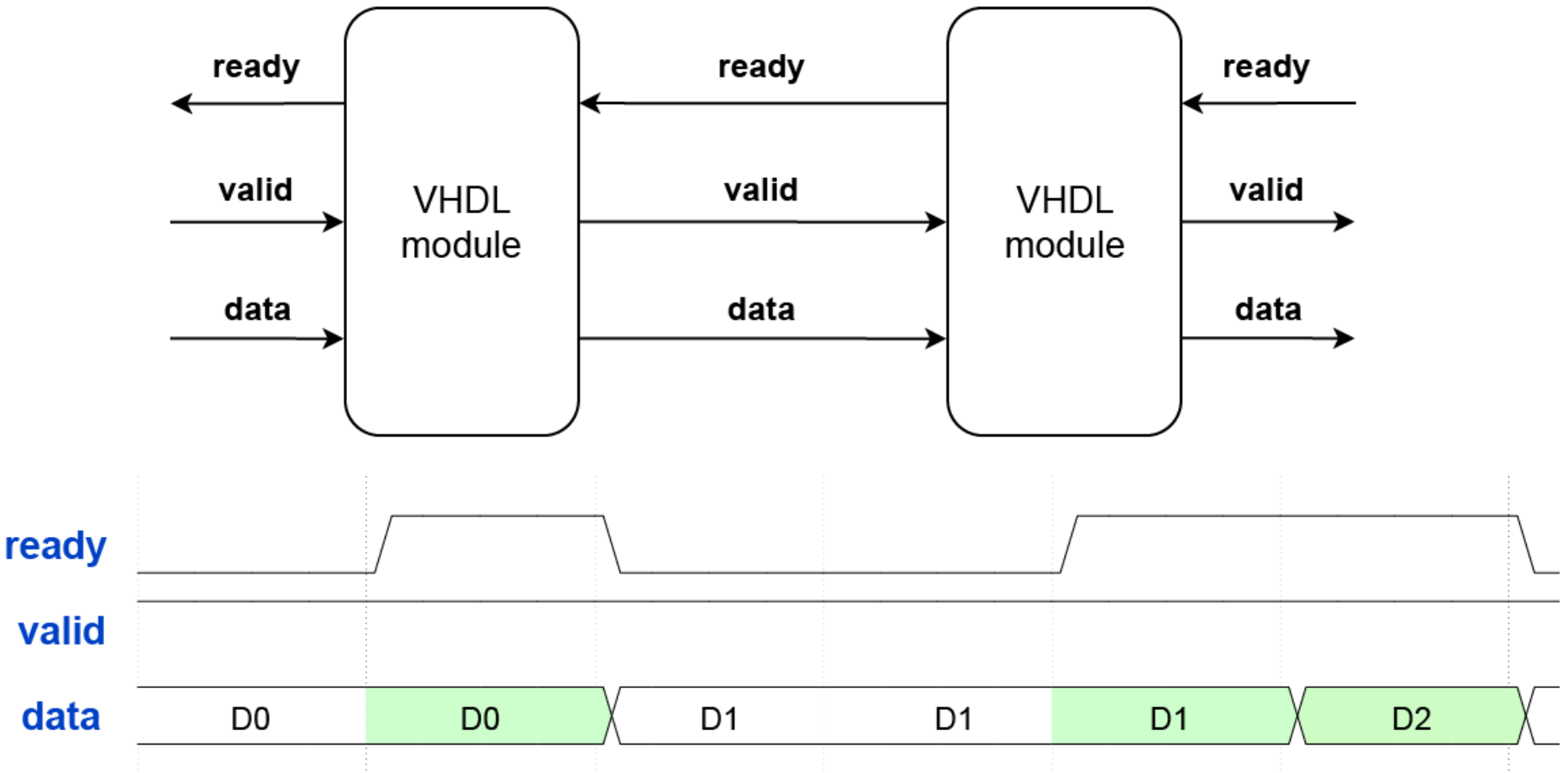
- AXI-lite protocol terdiri dari 5 channel

AXI Procotol (cont.)

- Setiap channel menginpmelentasikan protocol data, valid, ready



AXI Procotol (cont.)

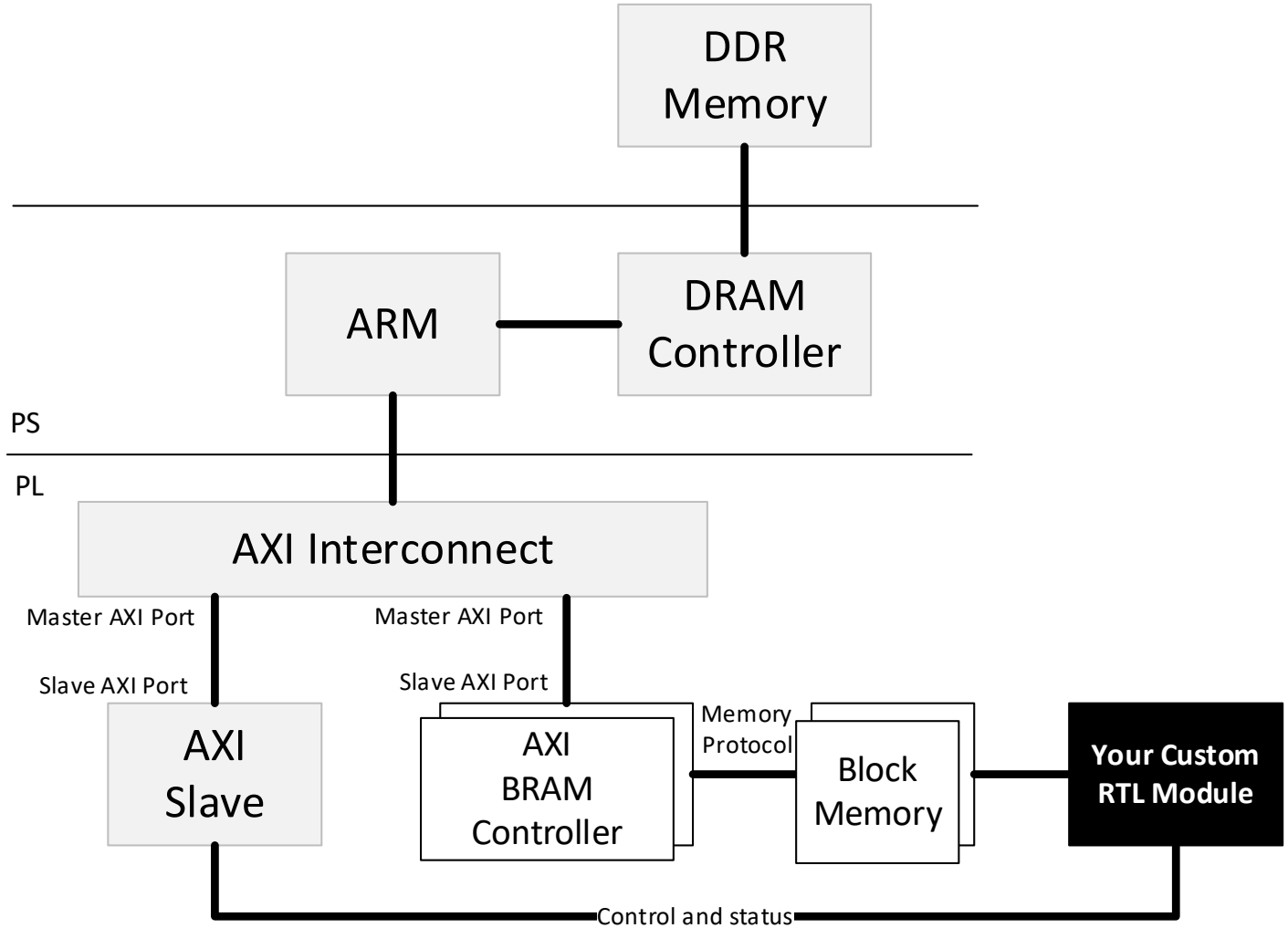


AXI Procotol (cont.)

- AXI-lite protocol terdiri dari 5 channel

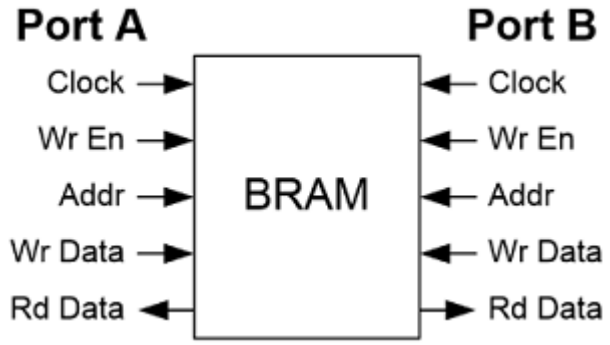

```
// ### AXI4-lite slave signals ###
// *** Write address signals ***
output wire        s_axi_awready,
input  wire [31:0] s_axi_awaddr,
input  wire        s_axi_awvalid,
// *** Write data signals ***
output wire        s_axi_wready,
input  wire [31:0] s_axi_wdata,
input  wire [3:0]  s_axi_wstrb,
input  wire        s_axi_wvalid,
// *** Write response signals ***
input  wire        s_axi_bready,
output wire [1:0]  s_axi_bresp,
output wire        s_axi_bvalid,
// *** Read address signals ***
output wire        s_axi_arready,
input  wire [31:0] s_axi_araddr,
input  wire        s_axi_arvalid,
// *** Read data signals ***
input  wire        s_axi_rready,
output wire [31:0] s_axi_rdata,
output wire [1:0]  s_axi_rresp,
output wire        s_axi_rvalid
```

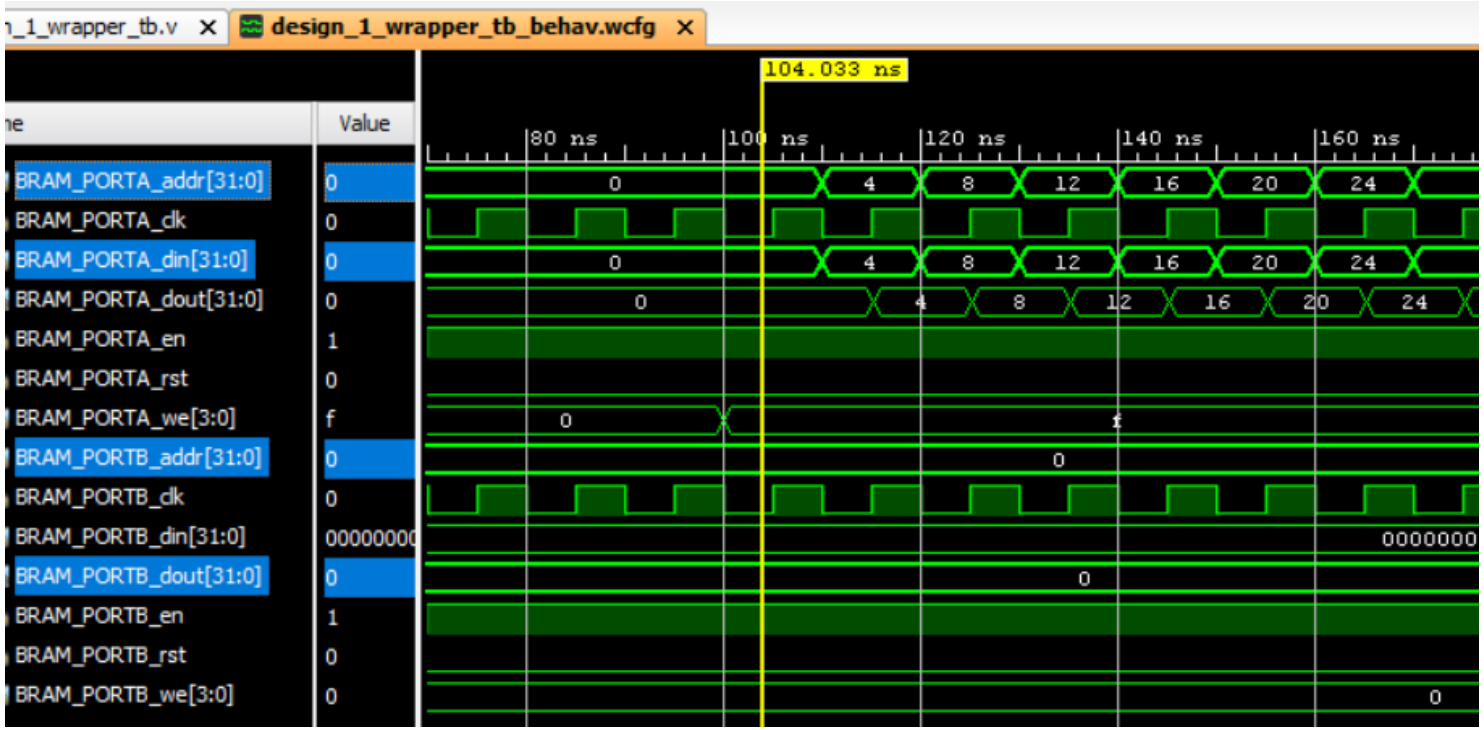
2. AXI BRAM Controller



BRAM Protocol

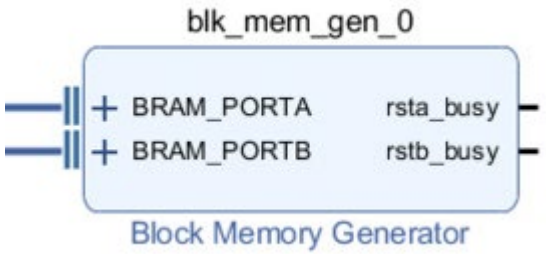
- Dual-port BRAM has two different ports. For example, port A can read to address 0 in the same clock cycle when port B writes to address 200.





BRAM Instantiation

- Instantiation using Block design



- Instantiation using XPM (Xilinx Parameterized Macro) inside your Verilog module.

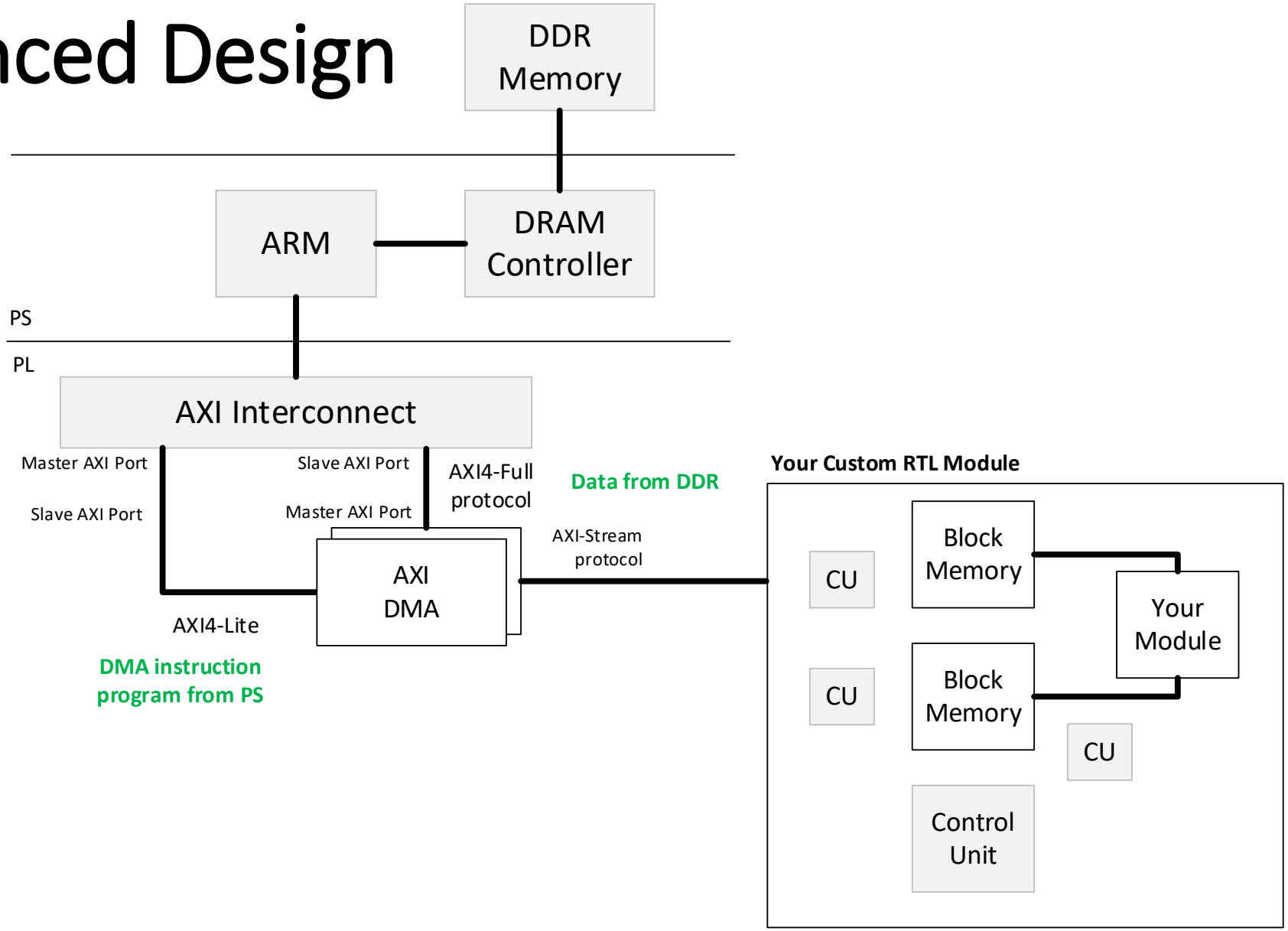
```
// xpm_memory_tdpram: True Dual Port RAM
// Xilinx Parameterized Macro, version 2018.3
xpm_memory_tdpram
#(
    // Common module parameters
    .MEMORY_SIZE(16384),                // DECIMAL, size: 2048x8bit= 16384 bits
    .MEMORY_PRIMITIVE("auto"),          // String
    .CLOCKING_MODE("common_clock"),     // String, "common_clock"
    .MEMORY_INIT_FILE("none"),          // String
    .MEMORY_INIT_PARAM("0"),            // String
    .USE_MEM_INIT(1),                   // DECIMAL
    .WAKEUP_TIME("disable_sleep"),      // String
    .MESSAGE_CONTROL(0),                // DECIMAL
    .AUTO_SLEEP_TIME(0),                // DECIMAL
    .ECC_MODE("no_ecc"),                // String
    .MEMORY_OPTIMIZATION("true"),       // String
    .USE_EMBEDDED_CONSTRAINT(0),        // DECIMAL

    // Port A module parameters
    .WRITE_DATA_WIDTH_A(8),             // DECIMAL, lebar data: 8-bit
    .READ_DATA_WIDTH_A(8),              // DECIMAL, lebar data: 8-bit
    .BYTE_WRITE_WIDTH_A(8),             // DECIMAL
    .ADDR_WIDTH_A(11),                  // DECIMAL, clog2(16384/8)=clog2(2048)= 11
    .READ_RESET_VALUE_A("0"),           // String
    .READ_LATENCY_A(1),                 // DECIMAL
    .WRITE_MODE_A("write_first"),       // String
    .RST_MODE_A("SYNC"),                // String

    // Port B module parameters
    .WRITE_DATA_WIDTH_B(8),             // DECIMAL, lebar data: 8-bit
    .READ_DATA_WIDTH_B(8),              // DECIMAL, lebar data: 8-bit
    .BYTE_WRITE_WIDTH_B(8),             // DECIMAL
    .ADDR_WIDTH_B(11),                  // DECIMAL, clog2(16384/8)=clog2(2048)= 11
    .READ_RESET_VALUE_B("0"),           // String
    .READ_LATENCY_B(1),                 // DECIMAL
    .WRITE_MODE_B("write_first"),       // String
    .RST_MODE_B("SYNC")                 // String
)
xpm_memory_tdpram_0
(
    .sleep(1'b0),
```

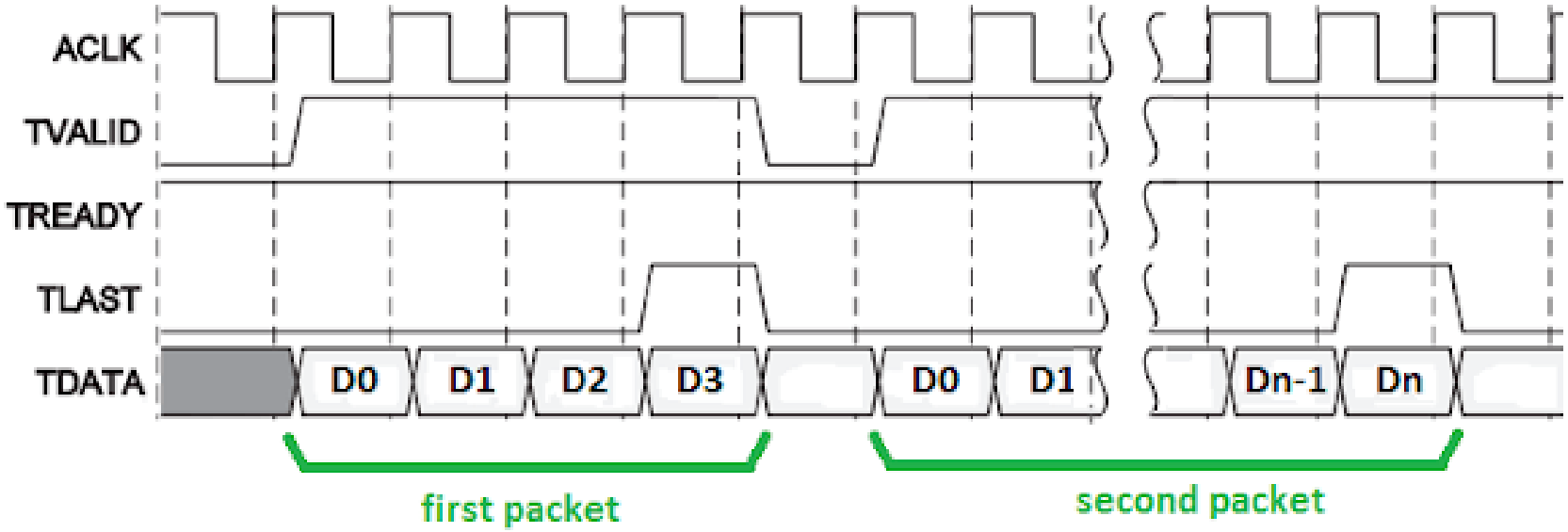
3. More Advanced Design

- Using DMA
- Data directly from DDR
- High throghput transfer
- Data flow control from PS



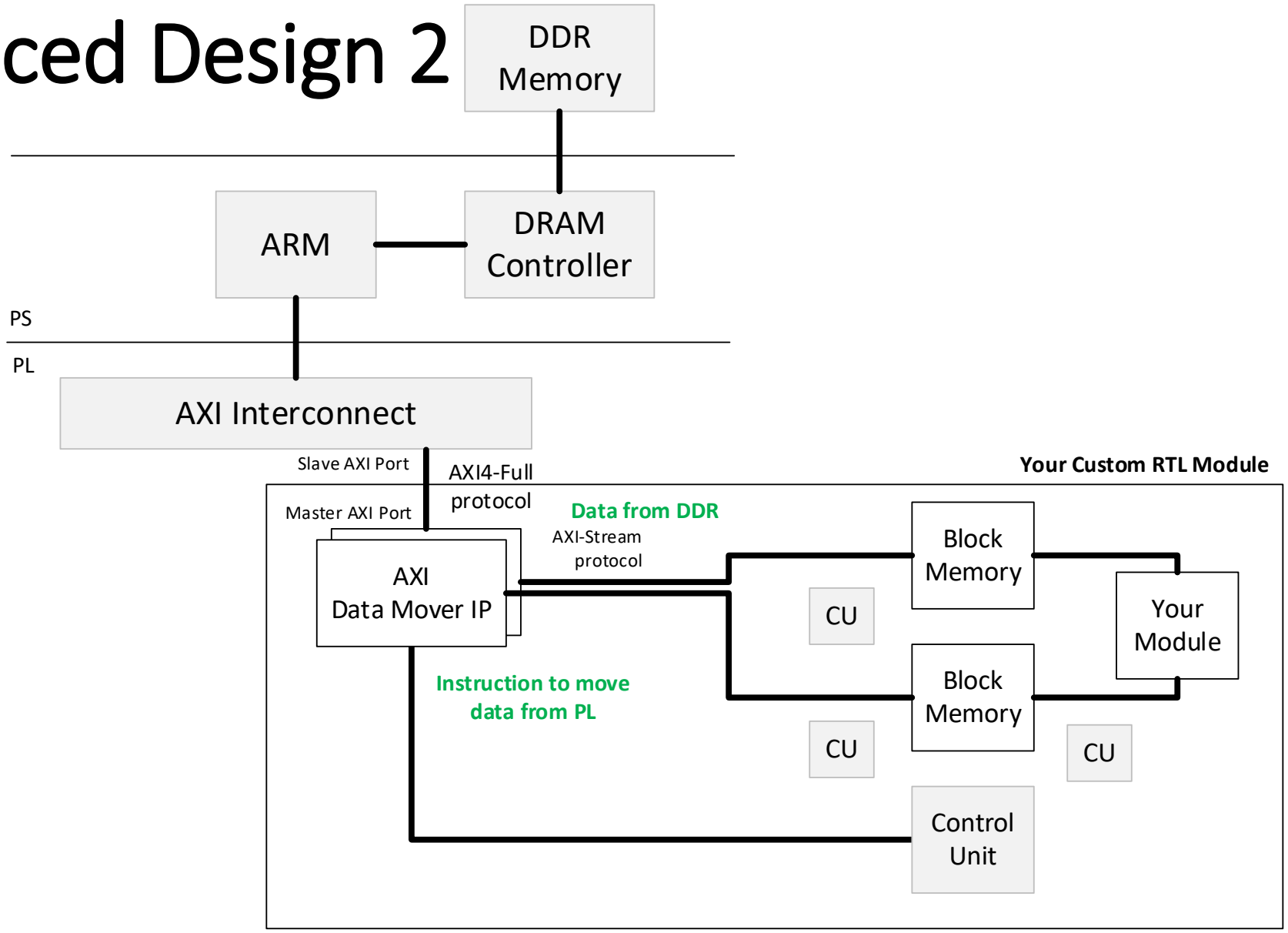
AXI Stream Protocol

- Protocol data, valid, ready



4. More Advanced Design 2

- Using AXI Data Mover IPData directly from DDR
- High throghput transfer
- Data flow control from PL



Perbandingan Arsitektur 1-4

Parameter	Architecture 1	Architecture 2	Architecture 3	Architecture 4
Protocol	AXI-Lite	AXI-Lite/AXI-Full, BRAM	AXI-Lite, AXI-Full, AXI-Stream, BRAM	AXI-Lite, AXI-Full, AXI-Stream, BRAM
Jumlah data	small	medium	large	large
Data speed transfer	slow	slow-medium	fast	fast
Software	Xilinx SDK (C) / PYNQ Python	Xilinx SDK (C) / PYNQ Python	Recommended to use PYNQ Python	Recommended to use PYNQ Python

- Pemilihan arsitektur tergantung aplikasi, jumlah data, kecepatan data yang diperlukan.

Thanks for Your Attention