FTK IPMC Users Report

Nikolina Ilic on behalf of FTK Team

Radboud University

Oct 9, 2018

FTK Introduction

- Identify hits by coarser grained Pix/SCT segments called super strips (SS)
- Find pre-stored MC track patterns that match the super strips in 8 layers. Matches are called Roads
- Retrieve fine resolution hits for all Roads
- Track fit in 8 layers
- Extrapolate 8 layer tracks to hits in remaining 4 layers. Do 12 layer track fit

IM

Cluster hits

DF

Sort hits into 64 regions

AMB

Match hits to pre-defined track patterns

AUX

Fit 8 layer tracks

SSB

Fit 12 layer tracks

FLIC

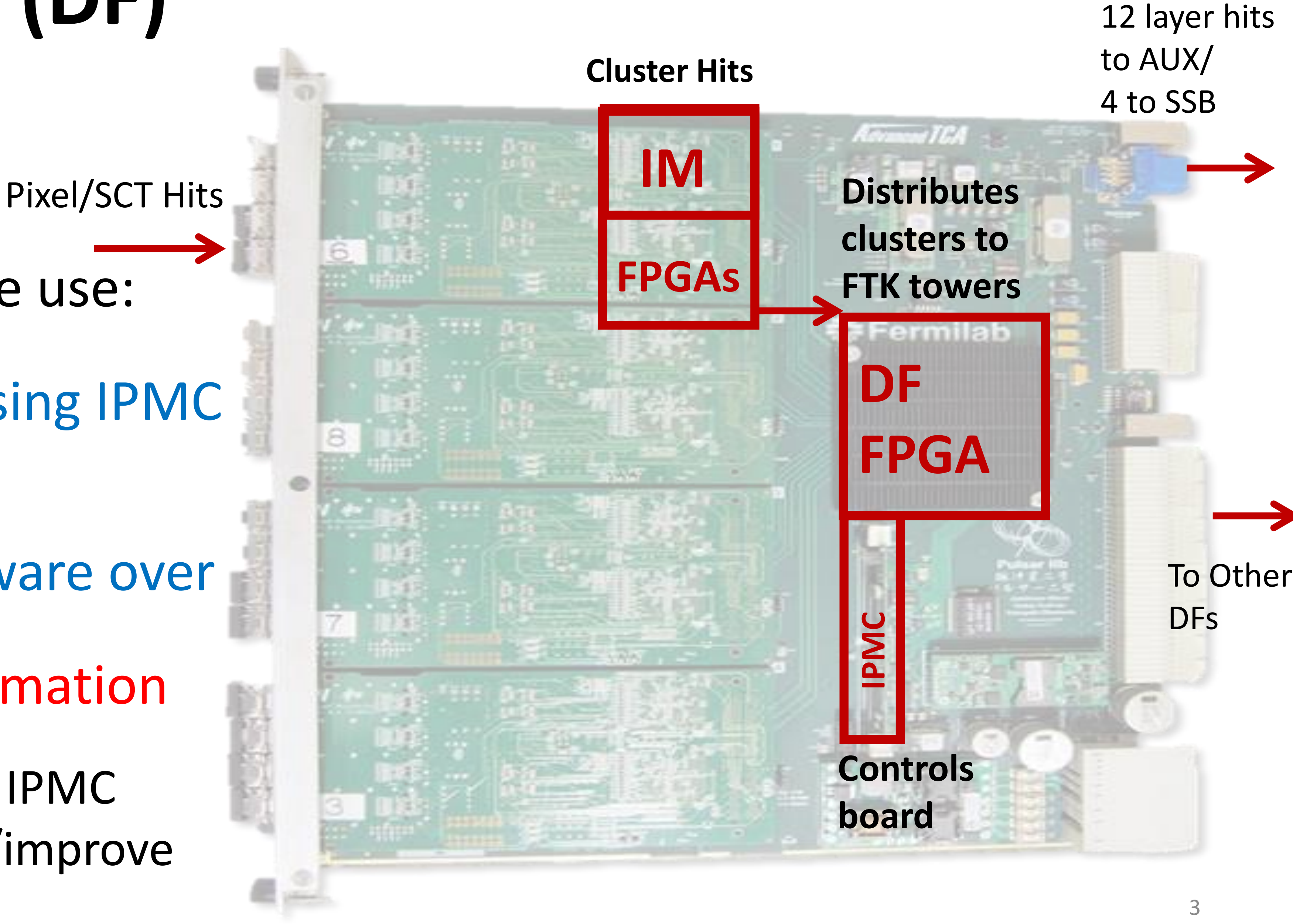
Send to HLT

Data Formatter (DF)

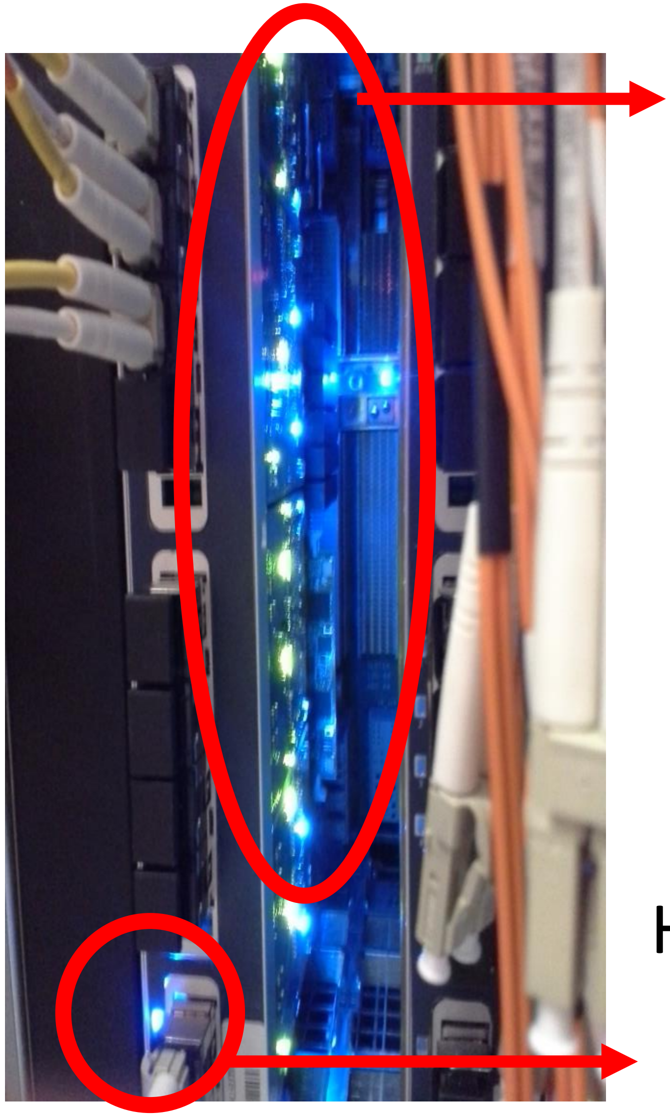
IPMC Functionalities we use:

- Hotswap DF
- Program DF FPGA using IPMC over ethernet
- Turn on RTM
- Program IPMC firmware over ethernet
- Provide sensor information

Working closely with LAPP IPMC engineers to debug issues/improve features



Hotswapping DF



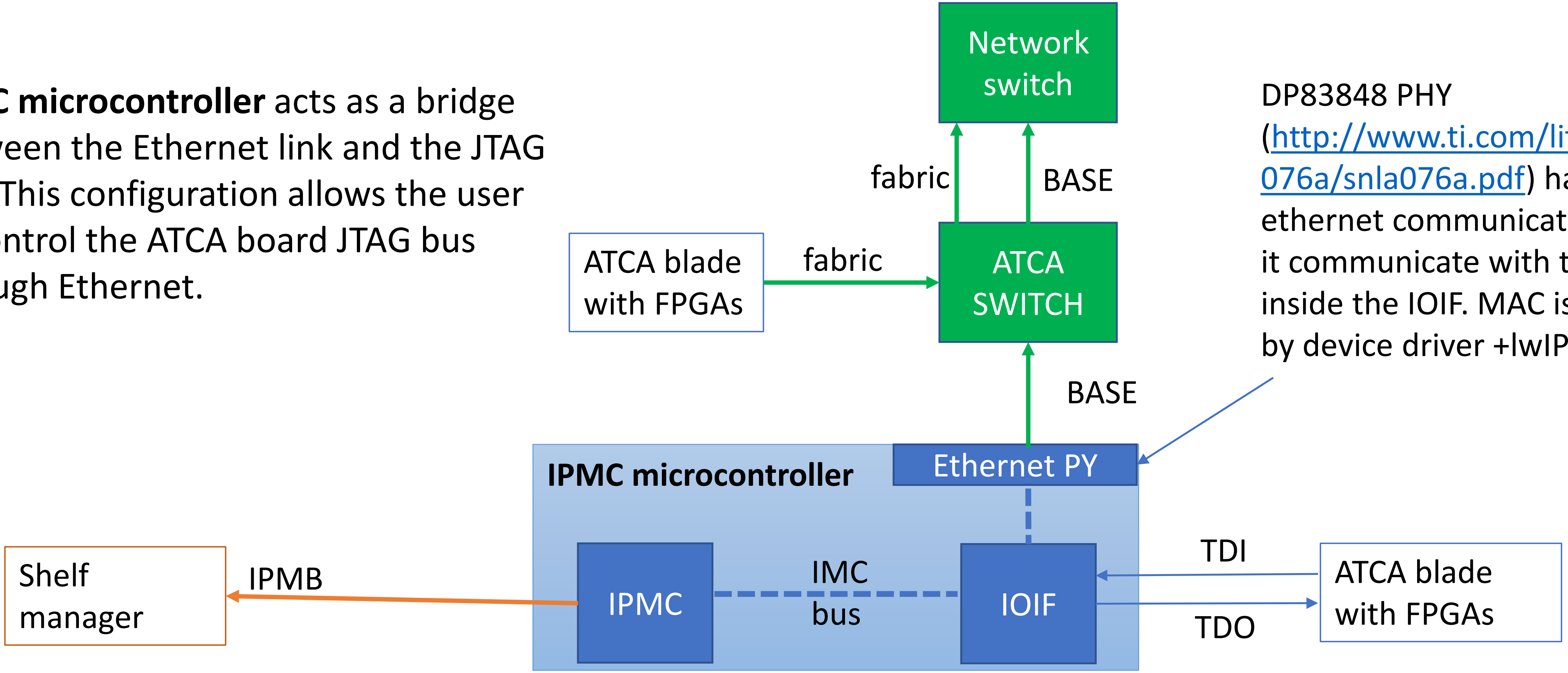
Board on

Hotswap light on

- Initially some instabilities with hotswapping DF board related to how IPMC logic copes with the fact that the DF front panel handle is loose
- Anti-bounce feature added to IPMC, IPMC now tolerates some level of handle instabilities for short periods of time
- Remaining hotswapping issues are minor and don’t significantly interfere with operation

Program DF FPGA using IPMC over ethernet

IPMC microcontroller acts as a bridge between the Ethernet link and the JTAG bus. This configuration allows the user to control the ATCA board JTAG bus through Ethernet.

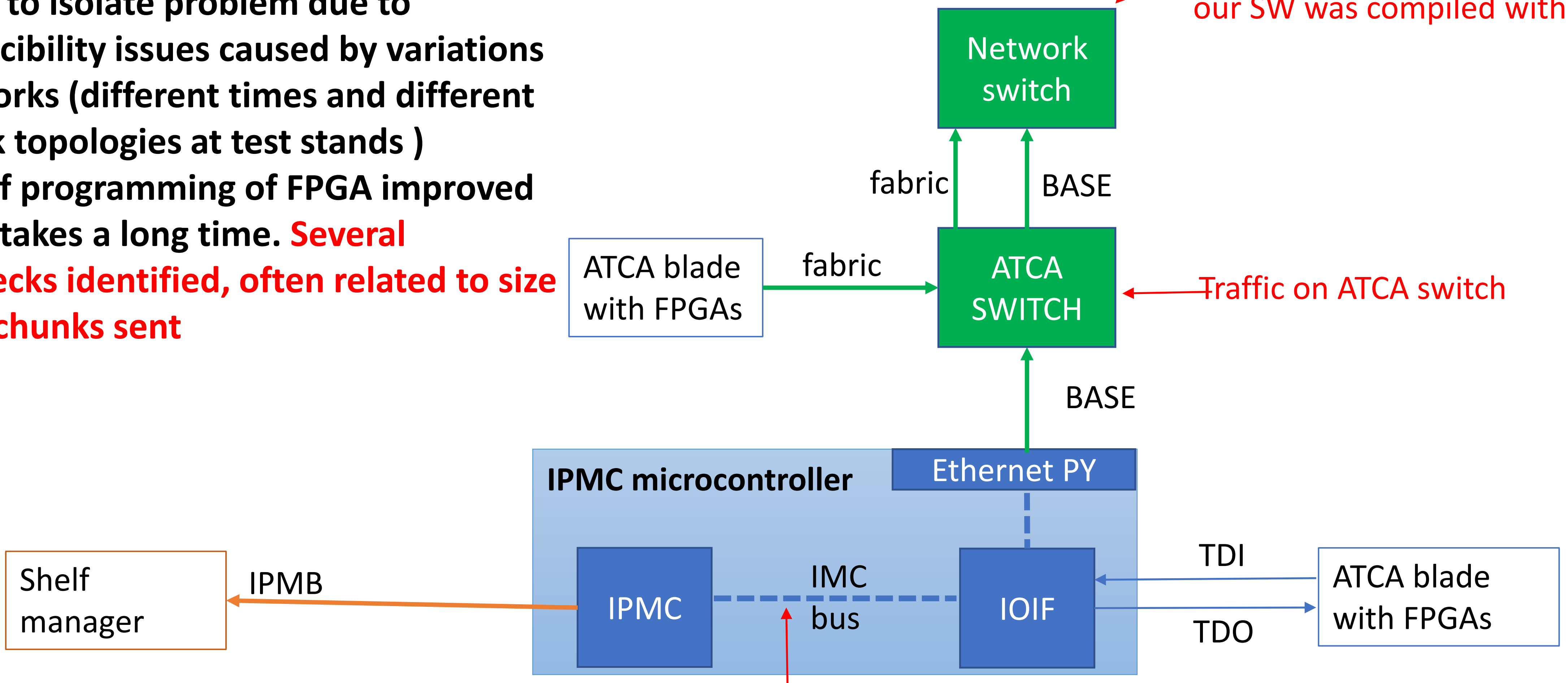


DP83848 PHY (http://www.ti.com/lit/an/snla076a/snla076a.pdf) handles ethernet communication.
it communicate with the MAC inside the IOIF. MAC is handled by device driver +lwIP

Communication of IPMC and IOIF could interfere with how IPMC communicates with shelf manager and IOIF/Ethernet PY communicates with switch.

Program DF FPGA using IPMC over ethernet

Difficult to isolate problem due to reproducibility issues caused by variations in networks (different times and different network topologies at test stands)
Speed of programming of FPGA improved but still takes a long time. Several bottlenecks identified, often related to size of data chunks sent



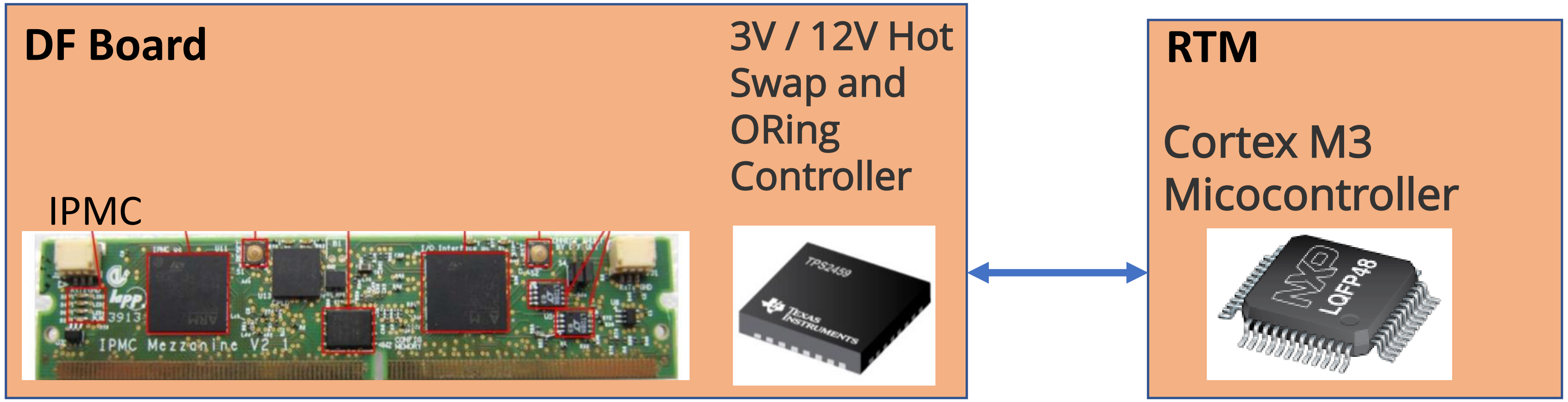
Whether our programming sw was written in C++ or python, which boost librairies our SW was compiled with.

Traffic on ATCA switch

Communication between IPMC and IOIF

Turning on RTM

- Turning on RTM is a 6-step process that follows IPMC/I2C standard. Several Bugs in process fixed, but still the microcontroller on RTM, does not respond. RTMs do not turn on 1/5 of the time



DF Board

3V / 12V Hot Swap and ORing Controller

IPMC

RTM

Cortex M3 Micocontroller

The Process:
1. When the RTM is first installed the IPMC detects the PS_N pin go low.
2a. IPMC communicates to ORing Controller over Mgt_I2C to set control bits such as output enable bits, current limits, etc
2b. The 3.3V management power is then controlled by the IPMC asserting the MP_ENABLE
3. The 3.3 V enables MMC microcontroller on the RTM
4. IPMC and MMC establish I2C communications
5. IPMC asserts the MP_ENABLE line and Oring Controllerthen turns on the main 12V power to the RTM
6. The ENn line is driven by the IPMC to the MMC micro. When this line is high the MMC micro should be in reset.

Program IPMC firmware over ethernet

- We update the firmware/software of the IPMC remotely over ethernet
- Sometimes the programming gets stuck, and the DF board has to be taken out of the shelf in order to reset the IPMC
- Have debugged and improved overall stability. However, currently programming IPMC over ethernet fails about ¼ of the time

Summary

- Close work was performed with LAPP engineers to debug issues that do not appear in LAPP test-stand where IPMC is on generic ATCA blade. Interaction of IPMC with Data Formatter and network brought up issues that couldn't be spotted in LAPP test stand
- Issues related to hotswapping were resolved
- Issues related to powering RTM have been debugged, but IPMC and RTM MMC still unable to establish communication (could be MMC issues, needs investigation)
- Remotely programming the IPMC often crashes in lab 4 and at P1 (works fine at LAPP – possibly related to CERN network)
- Monitoring sensor information not yet implemented from Data Formatter side

BACKUP

Question 1: Is there something about ATCA switch or within DF environment that makes IPMC program slower than when it is attached directly to network switch?

2 cases compared.

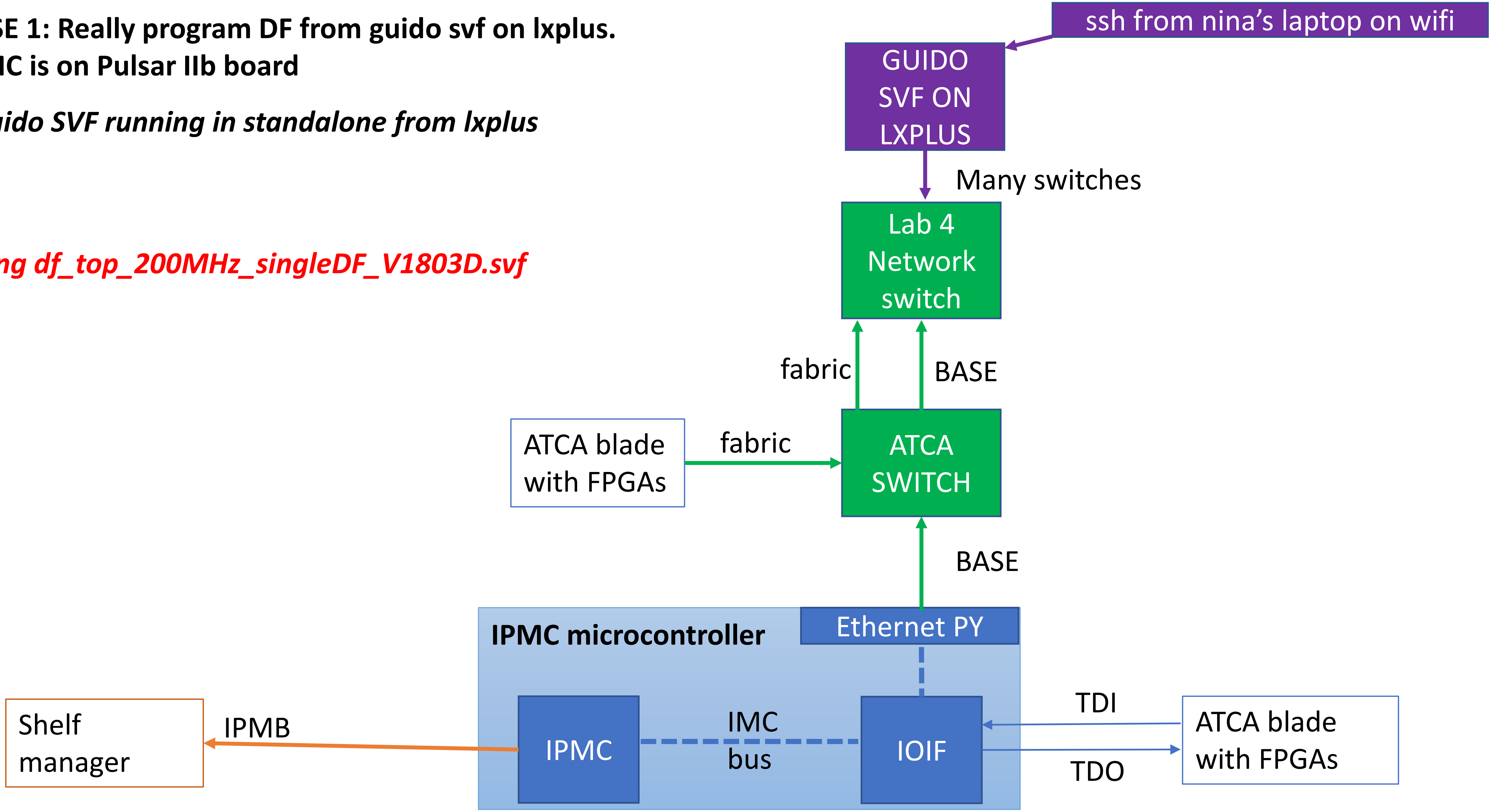
-Case 1: Really programming DF through ATCA switch when IPMC is on PULSAR IIb

-Case 2: Fake programming DF with IPMC just connected to Lab 4 network switch

CASE 1: Really program DF from guido svf on lxplus.
IPMC is on Pulsar IIb board

Guido SVF running in standalone from lxplus

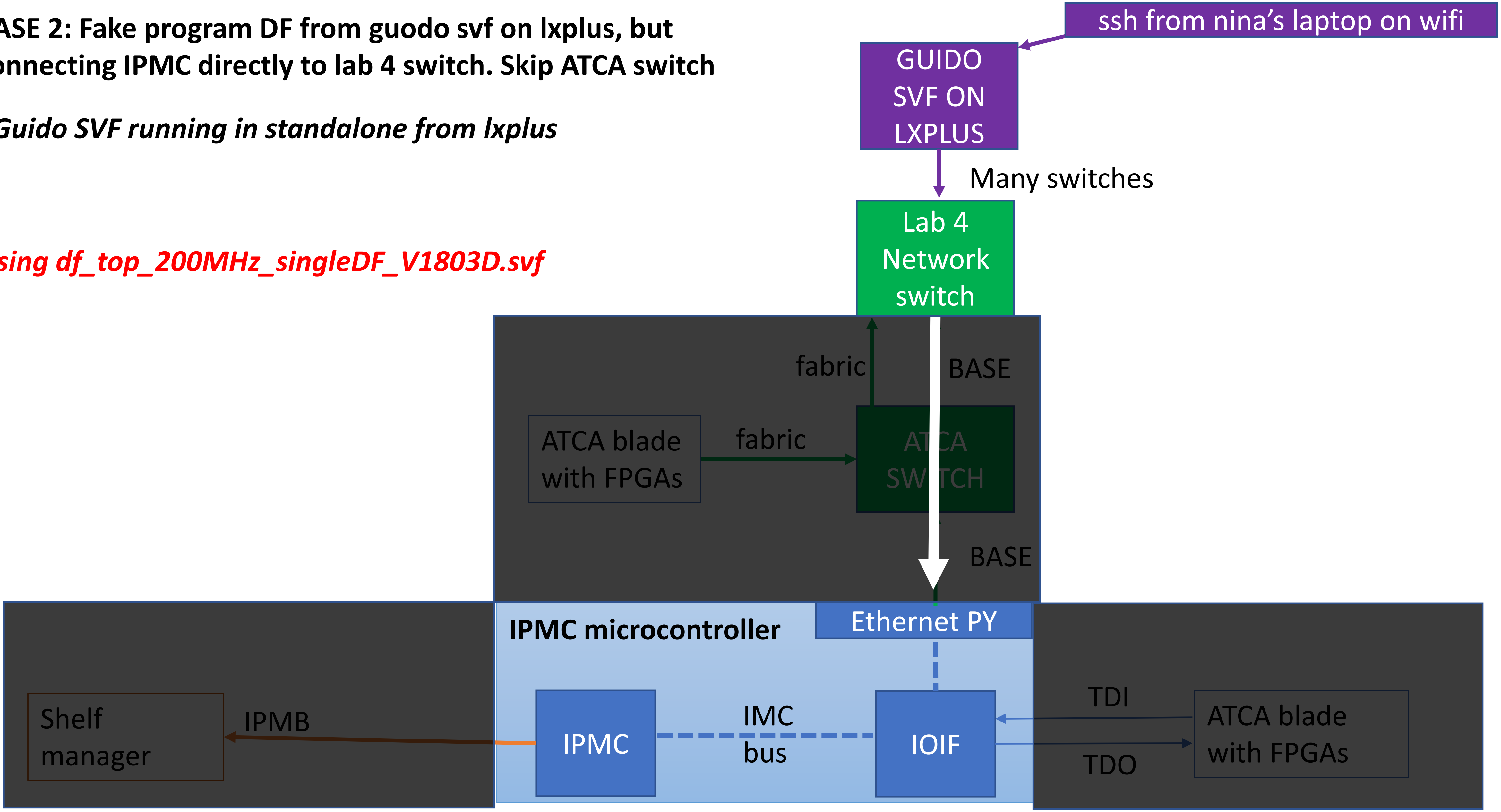
Using df_top_200MHz_singleDF_V1803D.svf



CASE 2: Fake program DF from guodo svf on lxplus, but connecting IPMC directly to lab 4 switch. Skip ATCA switch

***Guido SVF running in standalone from lxplus**

Using df_top_200MHz_singleDF_V1803D.svf



Question 1: Is there something about ATCA switch or within DF environment that makes IPMC program slower than when it is attached directly to network switch?

Answer 1: Yes: either the DF or ATCA switch is limiting IPMC pbytes received to 1.5 kbytes rather than 10 kbytes. The reason could be either something about how IOIF/ATCA switch communicate (can ATCA switch handle lwIP? What else?). Or something related to IPMC and IOIF communication, since sensor reading of IPMC happens only when IPMC is on Pulsar IIb, and this sensor info is also transferred to IOIF

Question 2: There is an extra layer of network switches between lxplus where guido svf is ran and the lab 4 network switch. Will running guido svf from a computer close to the lab 4 network switch speed programming up?

2 cases compared

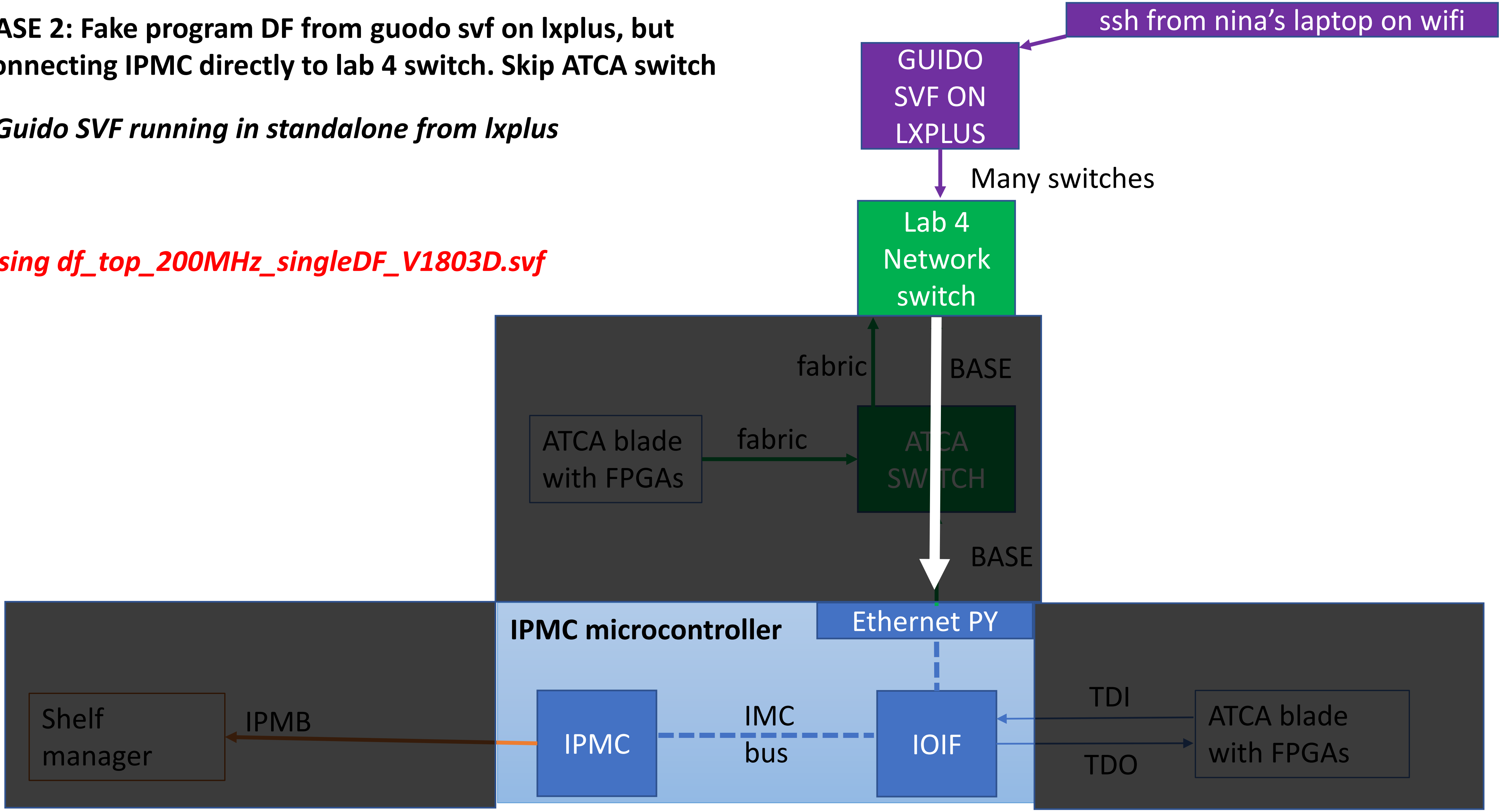
- Case 2 (previous slide): Fake program DF, running svf from lxplus
- Case 3: fake program DF, running guidosvf from computer connected to lab 4 network switch

Note: On lxplus guido svf can be ran in standalone. When running guido svf player in lab 4 it has to be compiled with TDAQ sw and Data Formatter package due to lack of access to boost libraries.

CASE 2: Fake program DF from guodo svf on lxplus, but connecting IPMC directly to lab 4 switch. Skip ATCA switch

***Guido SVF running in standalone from lxplus**

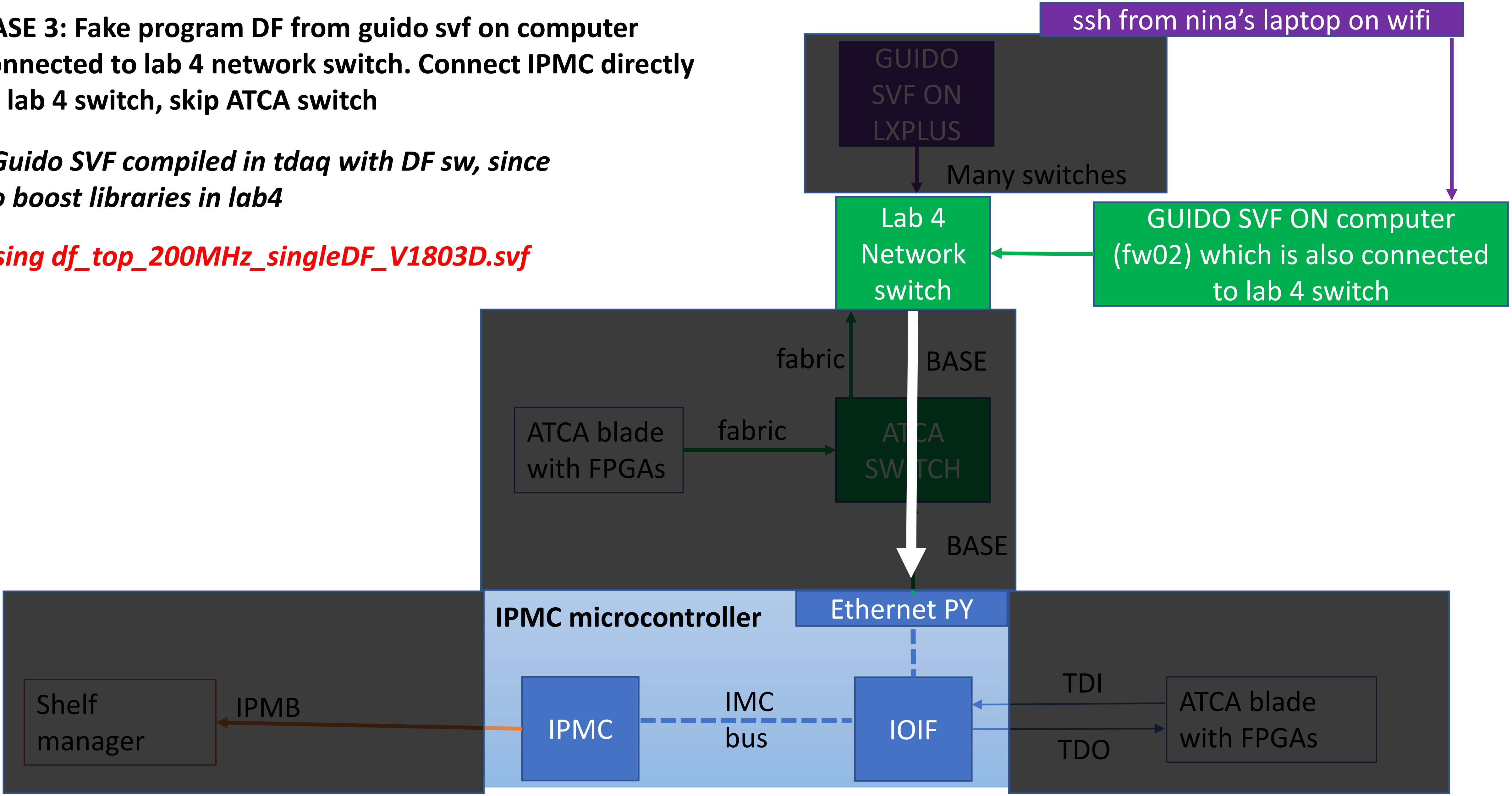
Using df_top_200MHz_singleDF_V1803D.svf



CASE 3: Fake program DF from guido svf on computer connected to lab 4 network switch. Connect IPMC directly to lab 4 switch, skip ATCA switch

***Guido SVF compiled in tdaq with DF sw, since no boost libraries in lab4**

Using df_top_200MHz_singleDF_V1803D.svf



Communication of IPMC and IOIF could interfere with how IPMC communicates with shelf manager and IOIF/Ethernet PY communicates with switch.

Question 2: There is an extra layer of network switches between lxplus where guido svf is ran and the lab 4 network switch. Will running guido svf from a computer close to the lab 4 network switch speed programming up?

Answer 2: NO. Running guido svf from computer closer to lab 4 network switch is actually SLOWER?! WHY? Could it be because guido svf is compiled/ran differently in the 2 cases (question 3)? Or could the wifi nina's laptop is on just be bad on some days (question 4)?

Note: On lxplus guido svf can be ran in standalone. When running guido svf player in lab 4 it has to be compiled with TDAW sw and Data Formatter package due to lack of access to boost libraries.

Question 3: Does running compiling guido svf with TDAQ/DF sw lead to slower programming than running guidosvf standalone?

2 cases compared

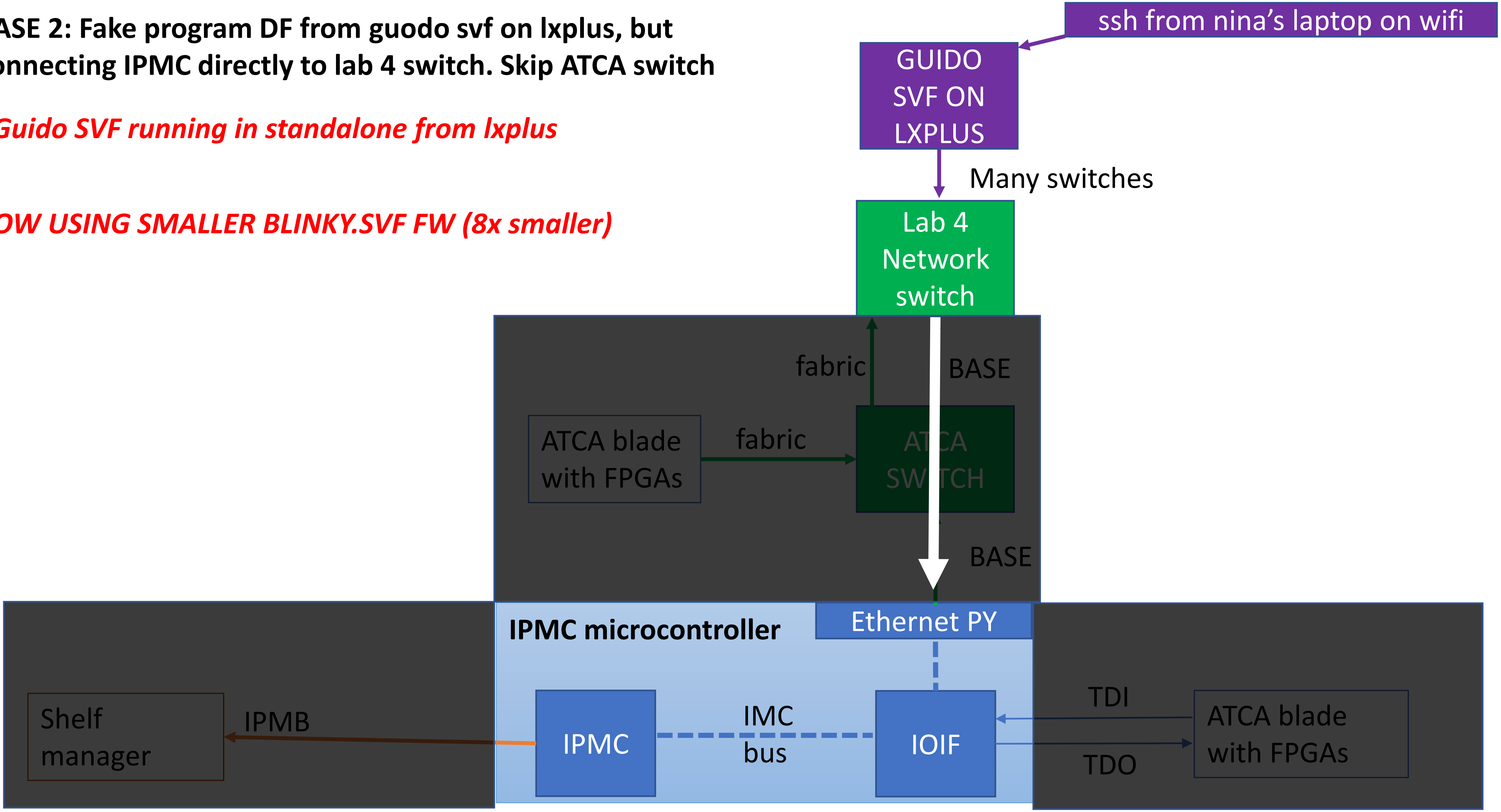
- Case 2: guido svf player compiled standalone on lxplus
- Case 4: guido svf player compiled with tdaq and DF package on lxplus

Note: to answer this question smaller fw was used: blinkey.svf

CASE 2: Fake program DF from guodo svf on lxplus, but connecting IPMC directly to lab 4 switch. Skip ATCA switch

Guido SVF running in standalone from lxplus

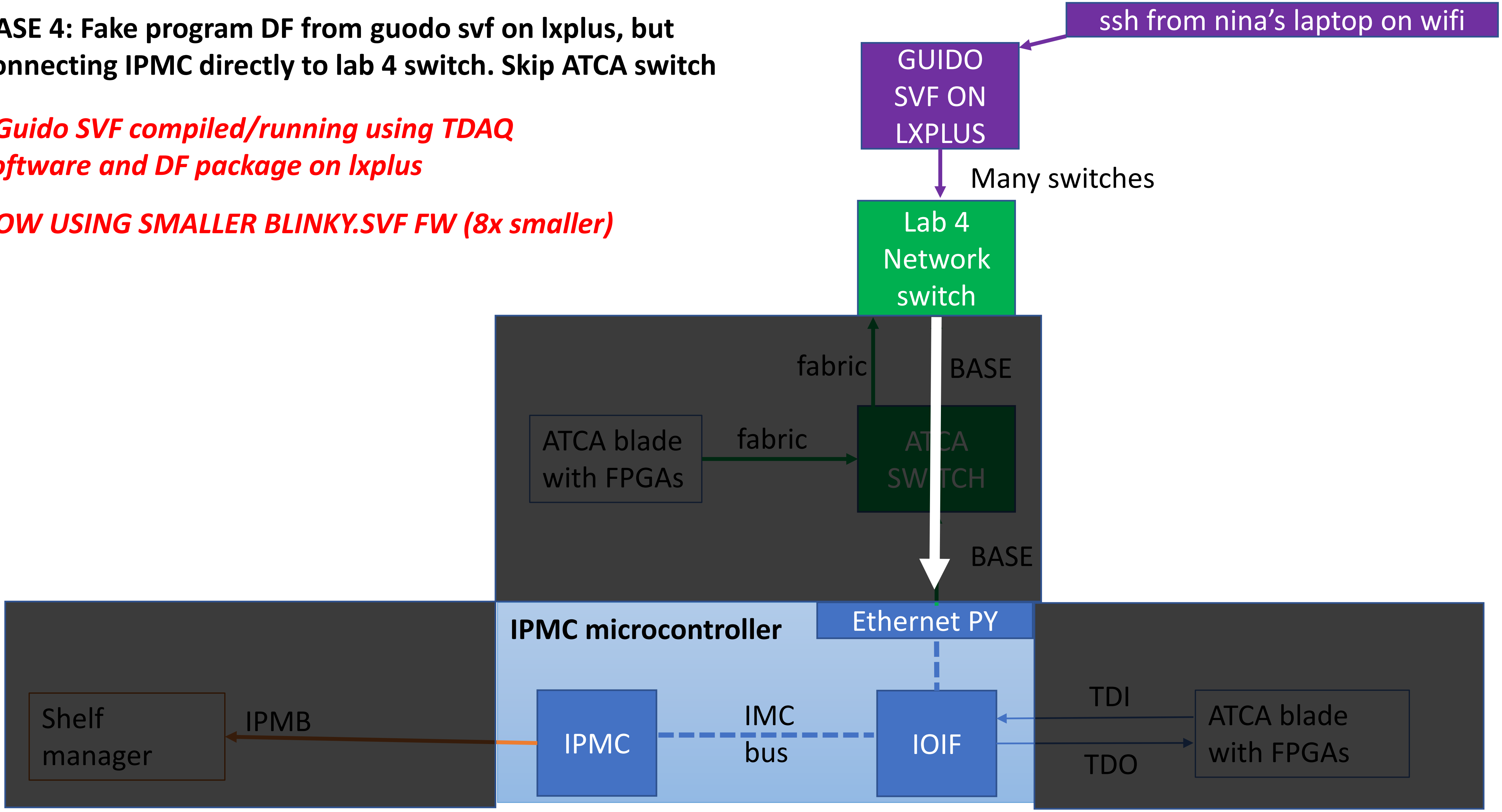
NOW USING SMALLER BLINKY.SVF FW (8x smaller)



CASE 4: Fake program DF from guodo svf on lxplus, but connecting IPMC directly to lab 4 switch. Skip ATCA switch

**Guido SVF compiled/running using TDAQ software and DF package on lxplus*

NOW USING SMALLER BLINKY.SVF FW (8x smaller)



Question 3: Does running compiling guido svf with TDAQ/DF sw lead to slower programming than running guidosvf standalone?

ANSWER 3: Yes. When guido svf player is compiled in DF sw with TDAQ (the only way to get it to work in lab 4), programming is 3x slower. Packet size received by IOIF is ~ 5 times smaller