# Slide 1

Good evening. Our idea is to simulate a Virtual FPGA so that the corresponding SW components can work without physical existence of FPGAs.

# Slide 2

In IxOS, a card consists of several FPGAs.

FPGA-team provides a user-document. Depending on those documents SW team configures FPGAs to get predictable outputs. For an example, if we want to send 1000 packets, depending on line rate, SW may write 1000 at one FPGA address or 100 at multiple addresses of a specific FPGA. Now, point is – SW, which writes values in FPGA, has its own calculation to convert a configuration in such a way so that outcome of FPGA’s behavior matches with user’s expectations.

Currently, when FPGA arrives, then only we can **precisely** test what SW is going to write against one specific configuration, done in IxExplorer. Sometime pushing correct configuration from SW to FPGA takes time because of SW bugs. Due to that SW-FPGA integration phase takes time.

In this hackathon, we tried to find out a solution, where we can test SW code without having any physical FPGAs.

Then obvious question is How?

Suppose, if - depending on the line-rate - SW correctly writes 1000 at one location and 100 at multiple location, we can say that internal logic of SW is correct. Therefore, I think, we can reduce SW-FPGA integration phase. Most importantly SW can start its internal-logic related testing before SW-FPGA integration phase. As SW testing starts early, SW-team can handle their bugs more quickly.

Ultimate goal of this project is to start SW code testing before arriving any physical FPGAs.

One additional benefit of this implementation can be that it will also help us to detect collaterals. If there is a stable build, run it against Virtual FPGA. Collect the log – golden test file – and compare with another build and decide which components’ behavior is changed – if any. Suppose comparison shows that there is **NO** difference, we can easily say that there may be some bugs in FPGA. Similarly, if any **difference is noticed**, SW team should look at the latest implementation.

# Slide 3

In this PoC we consider two such components, named Pine-Flag and TEC HW manager, which configure FPGAs very frequently. Pine flag uses TEC as a library and as we know pine-flag runs inside ixia-ports.

Therefore, let me give you a small description that how configuration pushed from IxServer to FPGA through Pine-Flag.

We have IxServer.

On the other hand, we have ixia-ports.

IxServer has a component, called PF-Manager. PF manager communicates with Pine flag. This communication between PF-Manager and Pine-flag happens through XML-RPC.

From K400 onwards pine flag receipts TX and RX configurations as input and pushes these configurations to TEC HW manager. Then TEC configures FPGAs through several UDP message exchanges.

As I said, Pine Flag and TEC manager run inside port, in this PoC we tried to run PineFlag and TEC in Linux VM.

# Slide 4

We already had a solution of Demo-IxServer. This server boots up with data-structure of card’s hardware abstraction layer. So, here we planned to active PF manager in Demo-IxServer and sent out the XML in such a way that Pine Flag, which ran inside Linux VM, could receive it.

Communication between IxExplorer/TCL scripts and Demo-IxServer was already present. But communication with Pine Flag and Demo-IxServer was the challenge we resolved in this PoC.

# Slide 5

So, now, we will be moving to give a Demo.

# Slide 6

Now we are again moving to Slides.

So, one can easily point out that there are very minimum differences between actual-run and simulated run. But within those two executions, I like to highlight one section, which is shown in the slide.

From this difference, we can guess that there might be some boundary check missing in SW code. Because in simulated environment read-FPGA operation returned zero. And as a result of subtracting some number from zero lead to a huge number. So, SW should have a check here.

# Slide 8

Corresponding code is shown here.

I must say that mentioned code is written depending on the FPGA-document. And as I know, this value should return always non-zero values.

# Slide 9

We also listed down pending important items of this PoC.

Ability to build with Demo-PF support. And it will not be part of production code.

Handle multiple ports. We only tried with single port.

Get to the exact same workflow as it is for normal mode.

Simulate FPGA read-operation. That will make this solution more realistic.

Think about how other SW components, like – PGID\_TEC, CMCC, Fpgadownloader, PortL1Service, which can be executed without having physical FPGAs.

If possible integrate with Idea-2062.