# 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 is consists of several FPGAs. FPGA-team provides a how-to document to SW-team. In that how-to document FPGA team provide address-value pair. Using those information SW team understands that how to program FPGA to get desirable output. For an example – say, if SW writes value 100 at location 0x1000, FPGA will send 100 packets of a specific stream.

Though this can be a configuration but most of the configuration pushed from IxNetwork or IxServer are very complex, which is not straight-forward and needs serious calculations.

Point is that SW has its own logics to convert IxNetwork/IxServer information so that FPGA’s behavior and user expectation can match.

As there are address-value pairs, until a physical FPGA arrives, SW team can only analyze their codes logically. Instead of this dry run if we can provide a platform where SW can run its code and get a snapshot of what address-value pair SW is going to configure in FPGA, SW-team can handle their bugs more quickly. Most importantly SW can start testing of their code before SW-FPGA integration.

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

One additional benefit of this implementation is that it will also help us to detect collaterals. If there is a stable build, run it against Virtual FPGA. Collect the log, and compare with the suspicious build and decide which components’ behavior is changed.

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 can take a look.

# Slide 3

In this PoC we consider one such component, named Pine-Flag, which configures FPGA very often. Pine flag runs inside ixia-port.

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

So we have IxServer. IxServer has a component called PF-Manager in it. PF manager communicates with Pine flag component. This communication happens through XML. From K400 onwards pine flag receipts TX and RX configurations as input and creates address-value pair for FPGAs. These address-value pairs are pushed to TEC HW manager component and TEC sends UDP packets to related FPGAs. Using these steps all address-value pairs are configured in all FPGAs.

As I said, Pine Flag and TEC manager is running inside port, in this PoC we tried to pull out that PF from port and ran it in Linux VM. With default implementation, TEC tries to communicate with FPGA. That path had to be changed as well.

# 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 send out the XML in such a way that Pine Flag, which runs inside Linux VM, can receive it. Additionally we also bypass FPGA and dumped all FPGA addresses and corresponding values, came from TEC, to a text file.

Communication between IxExplorer/TCL scripts and Demo-IxServer was already present. But communication with Pine Flag and Demo-IxServer was the challenge we tackle 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 non-zero always.

# 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 implementation more accurately.

Think about how efficiently Demo Pine Flag can be integrated to other cards.

If possible integrate with Idea-2062.