AMchip testing using IPbus

Students: Édouard Benoit, Yurii Piadyk

Supervisor: Francesco Crescioli

Why we used IPbus

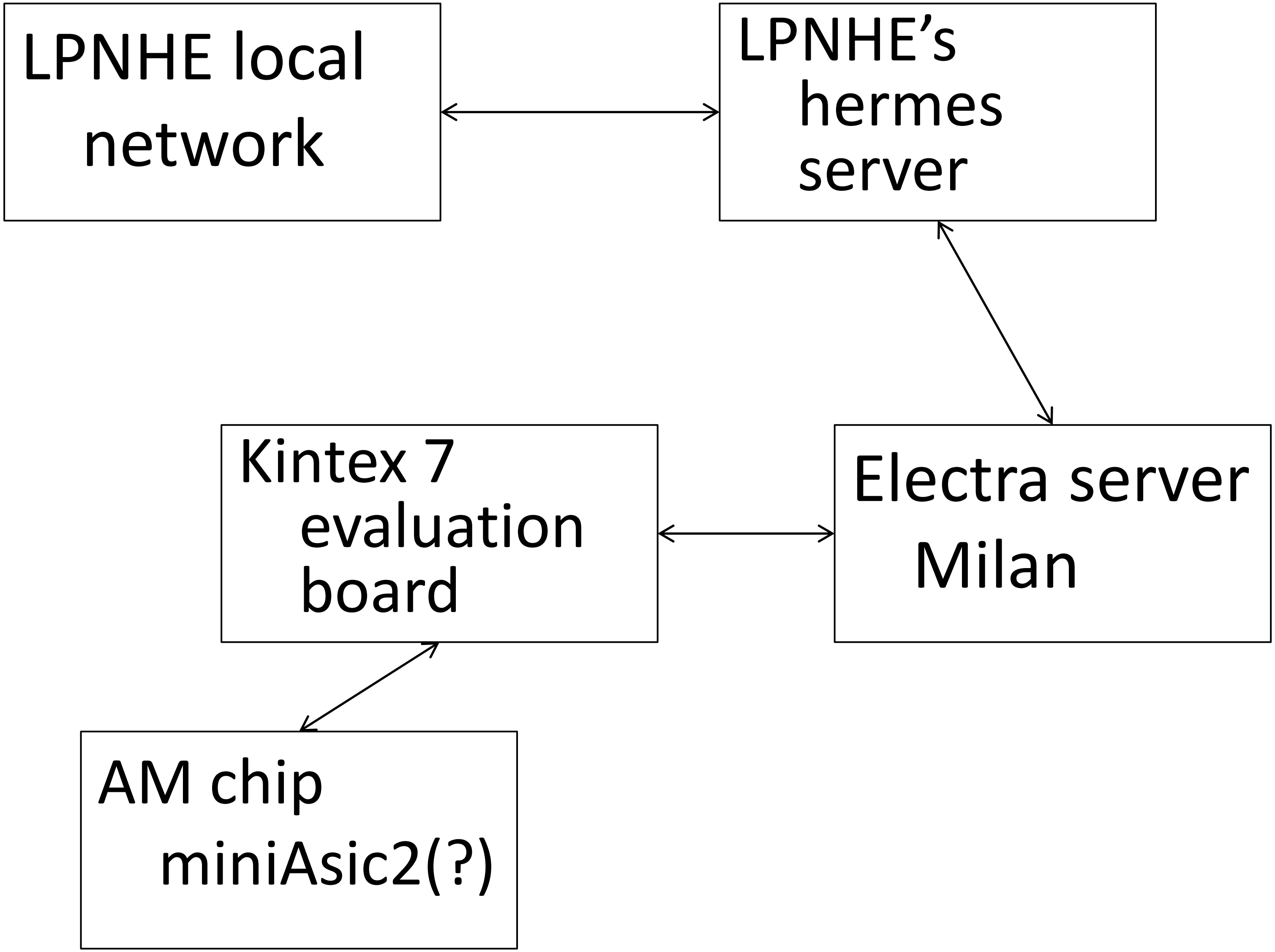
Why IPbus:

we wanted better speed than uart

What we have now:

possibility to remotely control AM chip’s JTAG

how remote?



LPNHE local network

LPNHE's hermes server

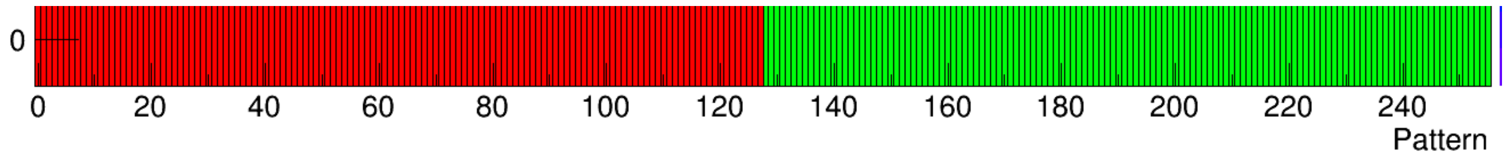
Kintex 7 evaluation board

Electra server Milan

AM chip miniAsic2(?)

What we were enabled to do

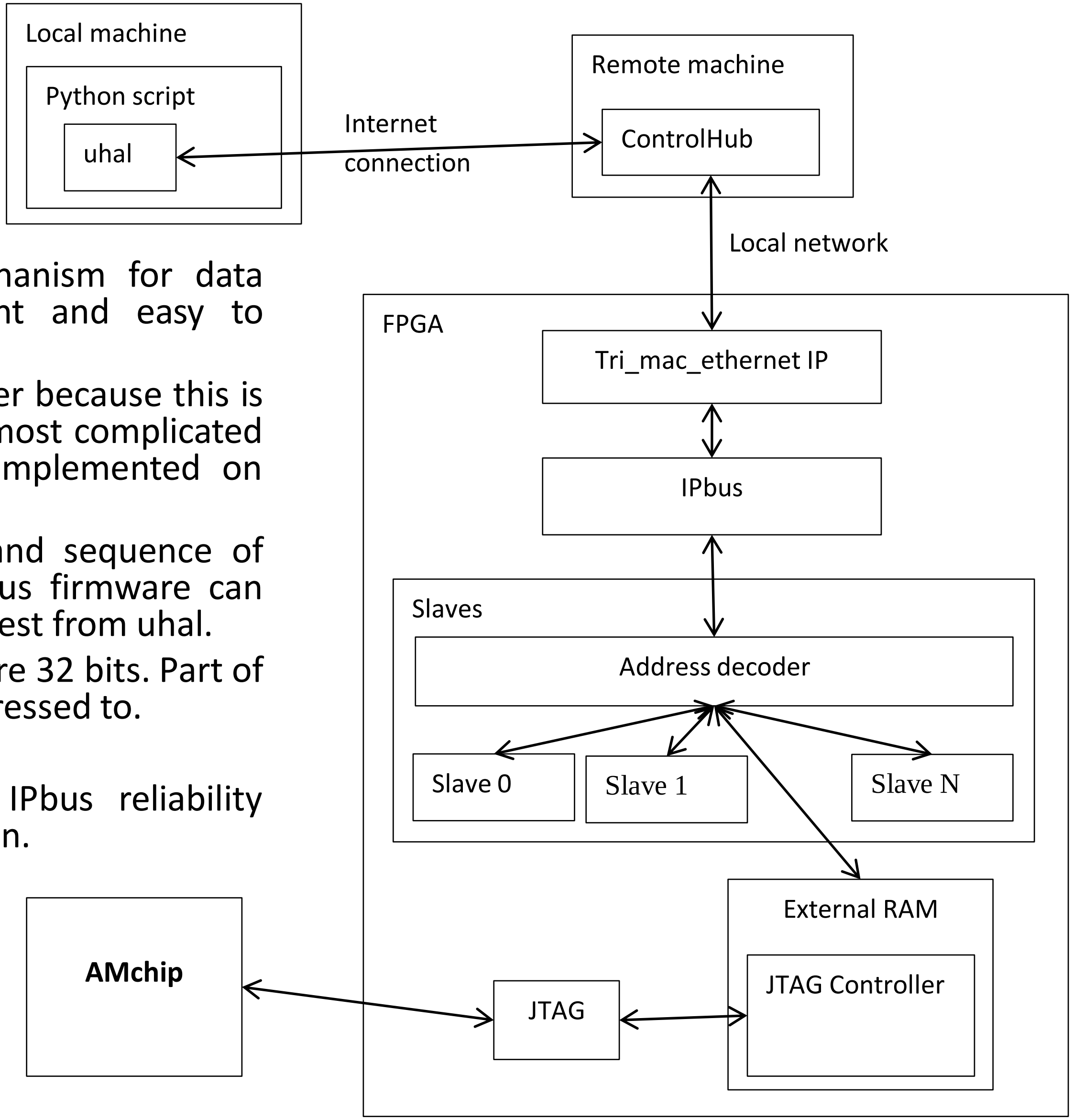
basic test, this plot shows which patterns missbehaved



→ we are able to see 2 very distincts zones, which fits with a already investigated feature of the chip: init in the first 128 patterns is known not to propagate fast enough

→ we can now assume we have a reliable way to communicate distantly with the chip

Firmware side (IPbus slaves)



IPbus implements the reliability mechanism for data transfer via unreliable but lightweight and easy to implement UDP protocol.

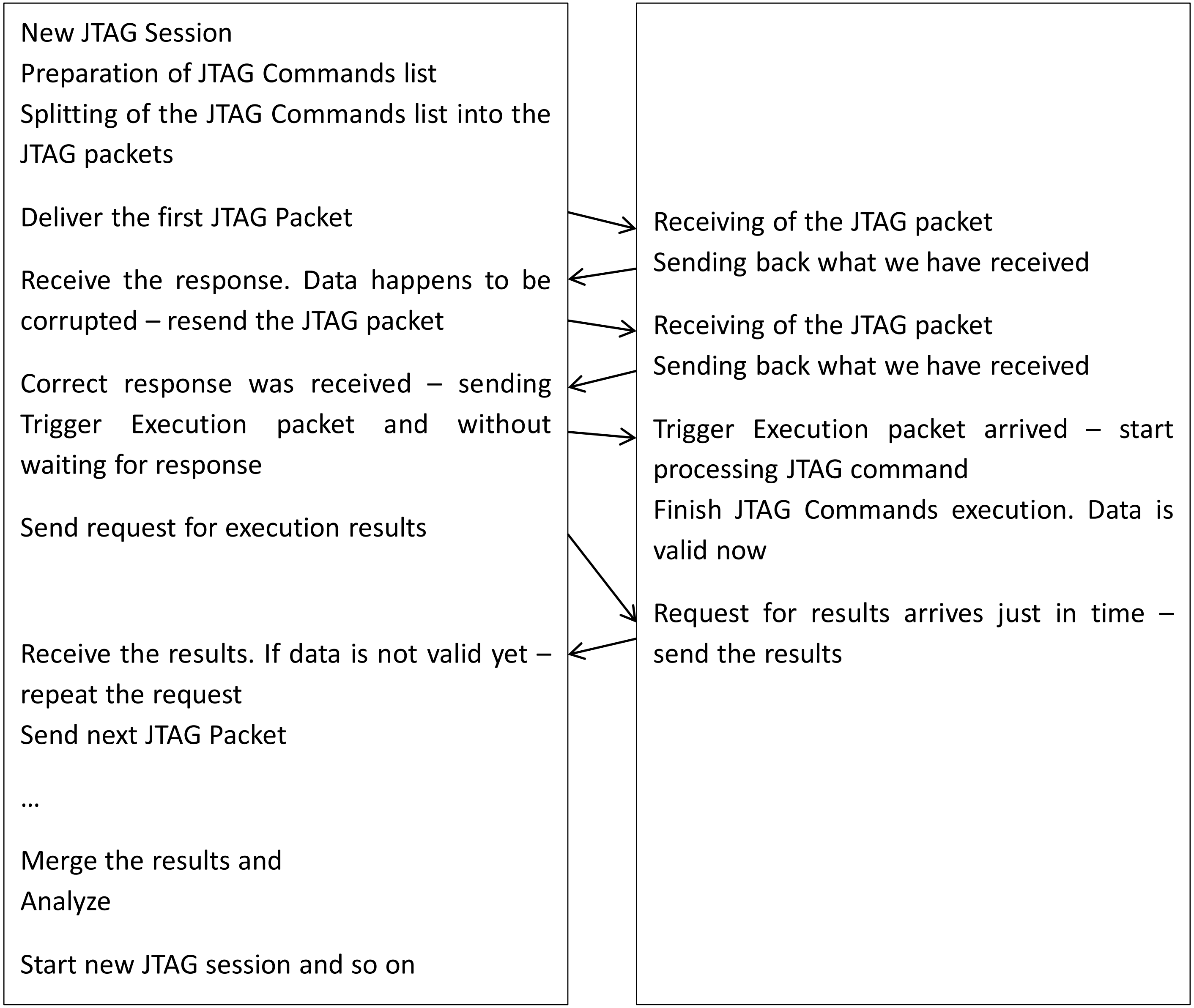
IPbus firmware can’t initiate data transfer because this is a privilege of the uhal software as the most complicated part of the reliability mechanism is implemented on software side.

IPbus packet consists of the header and sequence of read/write(with data) requests. So IPbus firmware can send data only as response on read request from uhal.

Data and address widths used in IPbus are 32 bits. Part of the address codes a slave request is addressed to.

There were problems with build in IPbus reliability mechanism so we needed to add our own.

Communication between the Software and Firmware



New JTAG Session

Preparation of JTAG Commands list

Splitting of the JTAG Commands list into the JTAG packets

Deliver the first JTAG Packet

Receive the response. Data happens to be corrupted – resend the JTAG packet

Correct response was received – sending Trigger Execution packet and without waiting for response

Send request for execution results

Receive the results. If data is not valid yet – repeat the request

Send next JTAG Packet

…

Merge the results and
Analyze

Start new JTAG session and so on

Receiving of the JTAG packet

Sending back what we have received

Receiving of the JTAG packet

Sending back what we have received

Trigger Execution packet arrived – start processing JTAG command

Finish JTAG Commands execution. Data is valid now

Request for results arrives just in time – send the results

Software side (python script)

```
import uhal, jtag # uhal is software provided with ipbus. Jtag is developed by us
d = uhal.getDevice("fpga", "chtcp-2.0://electra.fisica.unimi.it:10203?target=192.168.0.8:50001", "file://addresses.xml") # note that we connect to the FPGA not directly but via ControlHub (chtcp-2.0 instead of ipbusudp-2.0) because FPGA is connected locally to the remote machine
j = JTAG(d, buf_size=400) # New Session is stared automatically. Buf_size can be up to 2800 Jtag commands (350 words) – limited by the ipbus packet size
j.ResetAMchip() # some Jtag commands to reset the AMchip
id_info = j.GetIDCODE() # id_info contains the information about how many JTAG commands and where were added to total list in order to access register with IDCODE. id_info is needed to retrieve register value from the results
# alternative: id_info = j.access_register(IR=0x1, DR=0x0)
j.access_long_register(…)
…
j.Dispatch() # during dispatch commands are splitted to packets, sent, executed and then results are merged
j.PrintResults() # prints results of the execution of all commands
print “IDCODE = ”, j.retreive_register(id_info) # or we can retrieve interesting for us information
j.NewSession()
…
```

Outlook

- Optimize JTAG commands delivery in terms of speed and reliability
- Add another IPbus slave for fast serial link connection
- Do the tests

Acknowledgements

We should thank to Francesco for his help and advices