EE533 Spring 2026
Lab 4 Report
NetFPGA Bitfile Generation and Using NetFPGAE

Group: 14

Group member: **Zixuan Gong** **8529287764**

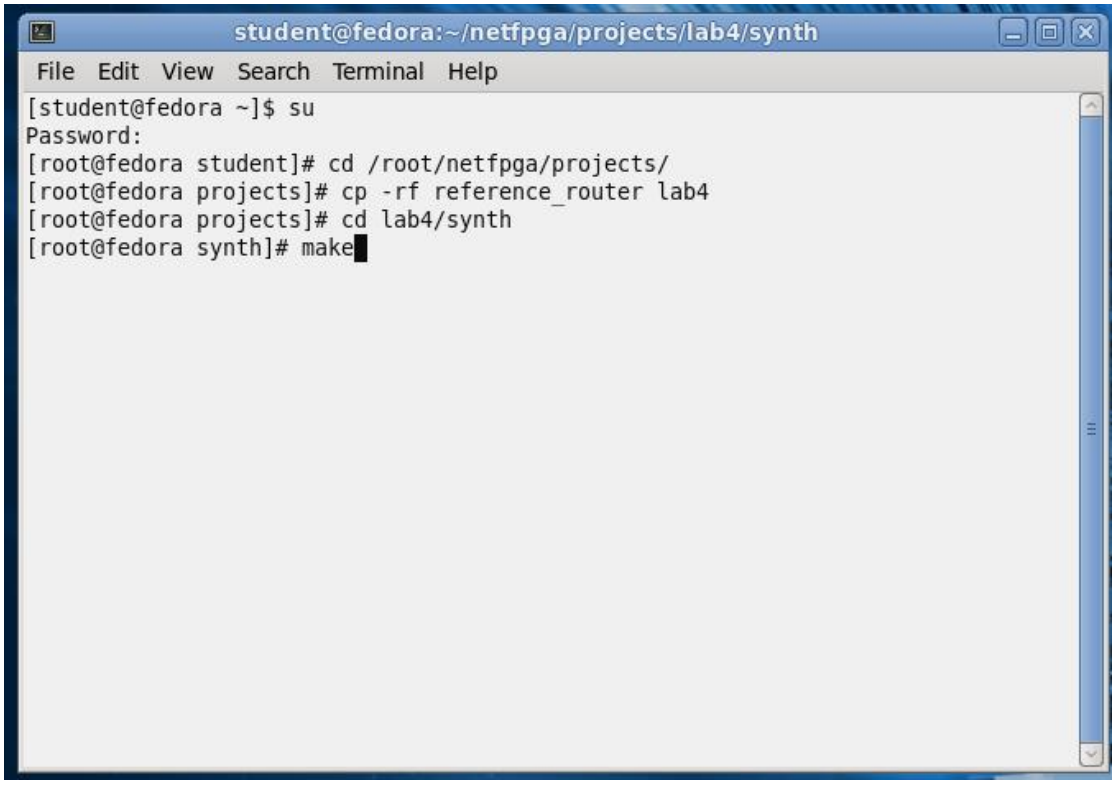
Yuhan Liang **9363204003**

Zhongqi Zhao **7864803206**

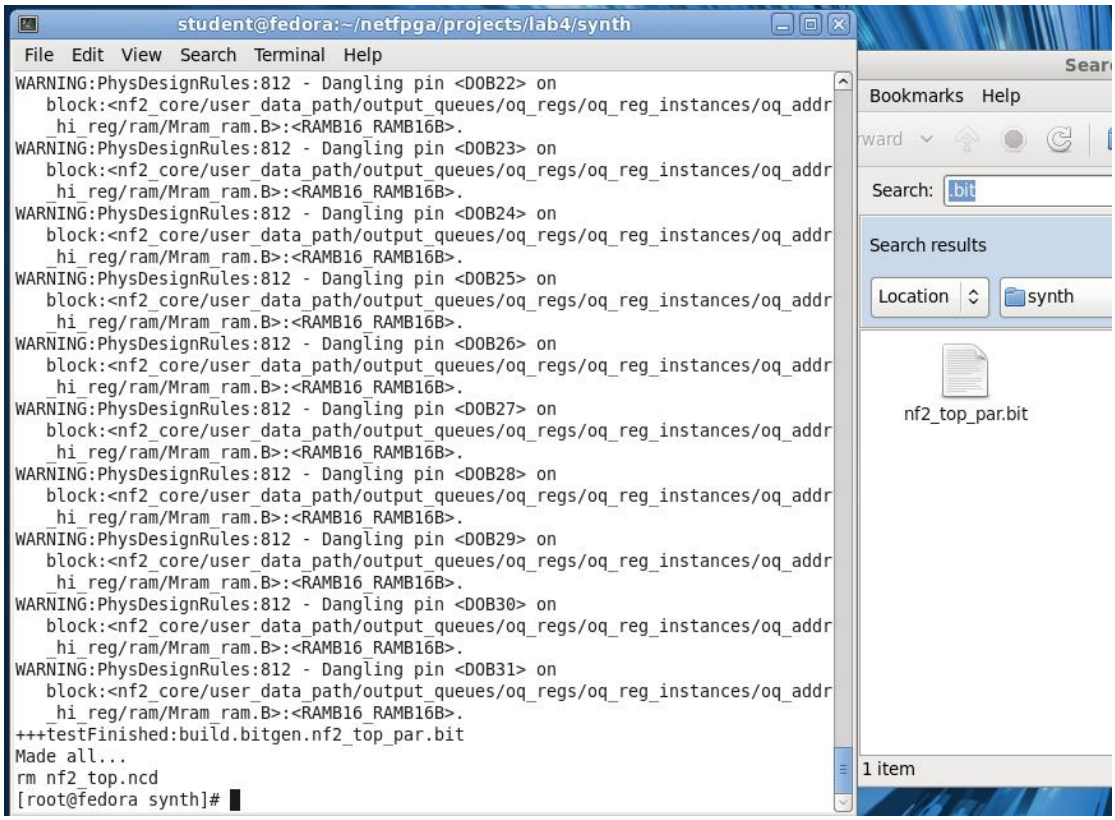
Github Link:

https://github.com/yuhanlia1/EE533-lab4-team13.git
(All files from our group members were organized and uploaded by Yuhan.)

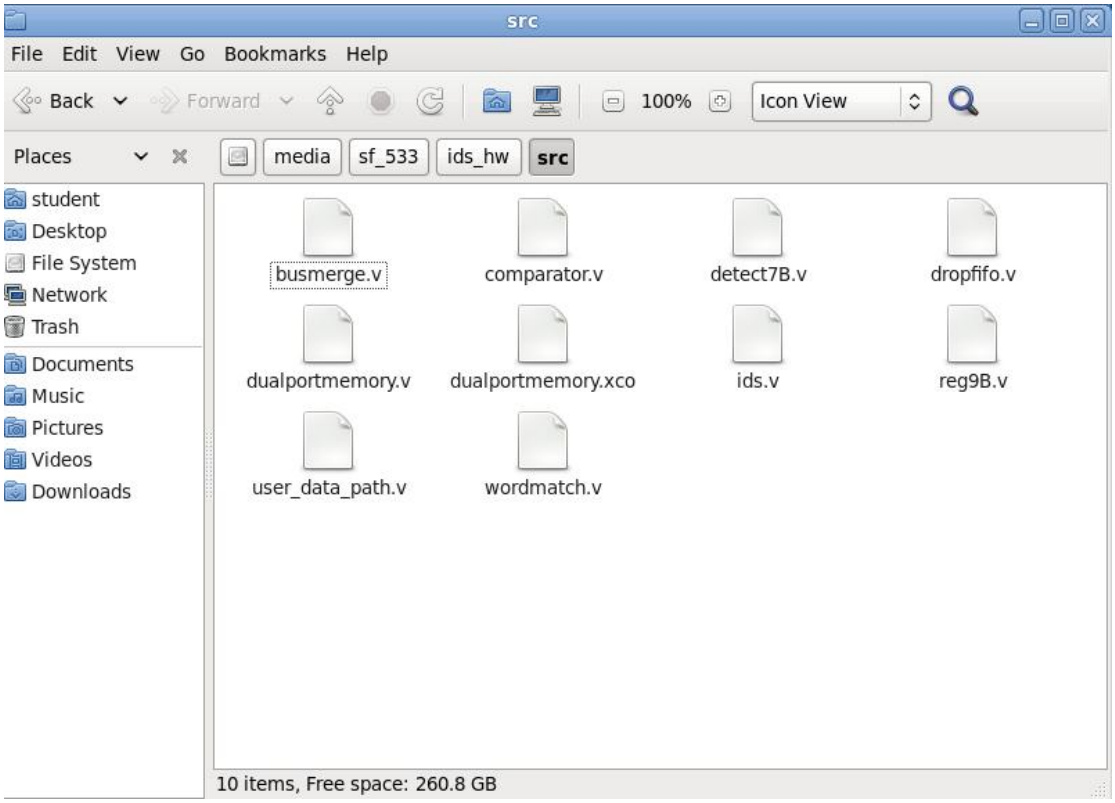
Part 1: Download and Set up NetFPGA Tool Virtual Machine



After ‘make’, we got the bitfille of refence_router.



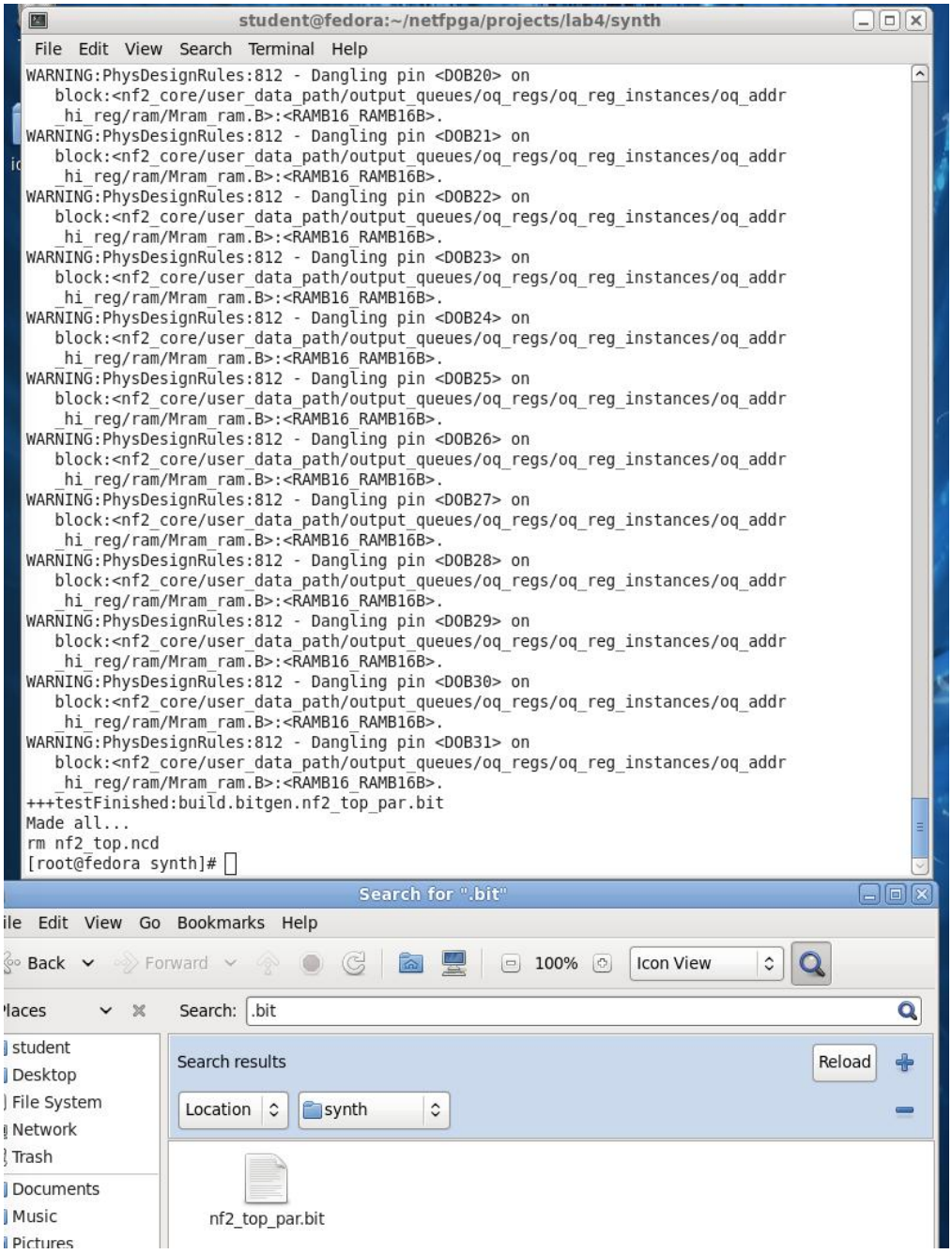
Part 2: Compile and generate a design bitfile for NetFPGA



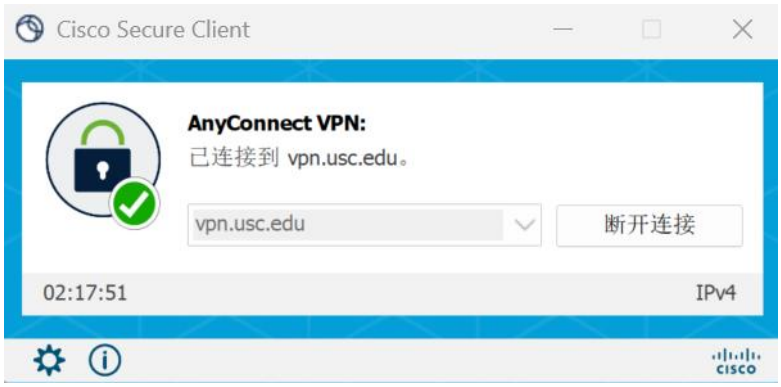
We solved errors by editing 'project.xml & ids.xml' and copying netlist 'dualmemory.ngc' to synth. Finally get the bitstream of Router with IDS (show the

details code in the part 7)

```
Processing /root/netfpga/lib/verilog/core/io/mdio/xml/mdio.xml...
WARNING: No module specific XML found for module 'core/cpci_bus'
Processing /root/netfpga/lib/verilog/core/dma/xml/dma.xml...
WARNING: No module specific XML found for module 'core/user_data_path/udp_reg_ma
ster'
WARNING: No module specific XML found for module 'core/io_queues/add_rm_hdr'
Processing /root/netfpga/lib/verilog/core/strip_headers/keep_length/xml/strip_he
aders.xml...
Processing /root/netfpga/lib/verilog/core/utils/xml/device_id_reg.xml...
WARNING: No module specific XML found for module 'core/utils/generic_regs'
Processing /root/netfpga/projects/lab4/include/ids.xml...
Definition for module 'nf2_mac_grp' not loaded at /root/netfpga/bin/nf_register_
gen.pl line 59
make: *** [registers] Error 255
[root@fedora synth]#
```

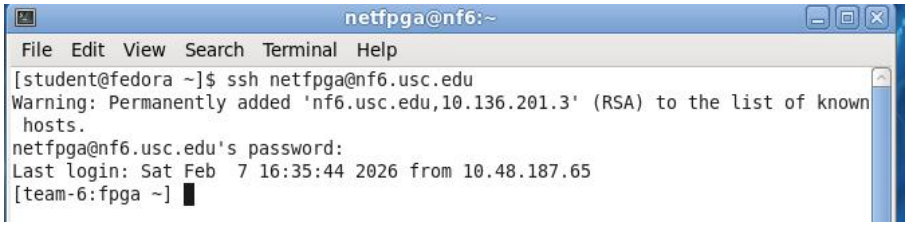


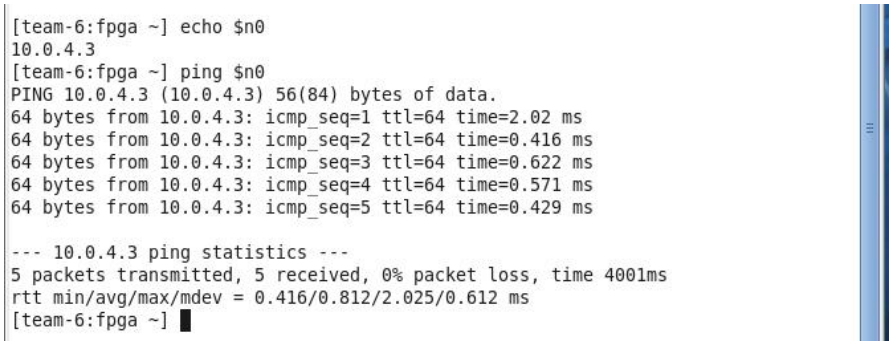
Part3: Set Up VPN to USC



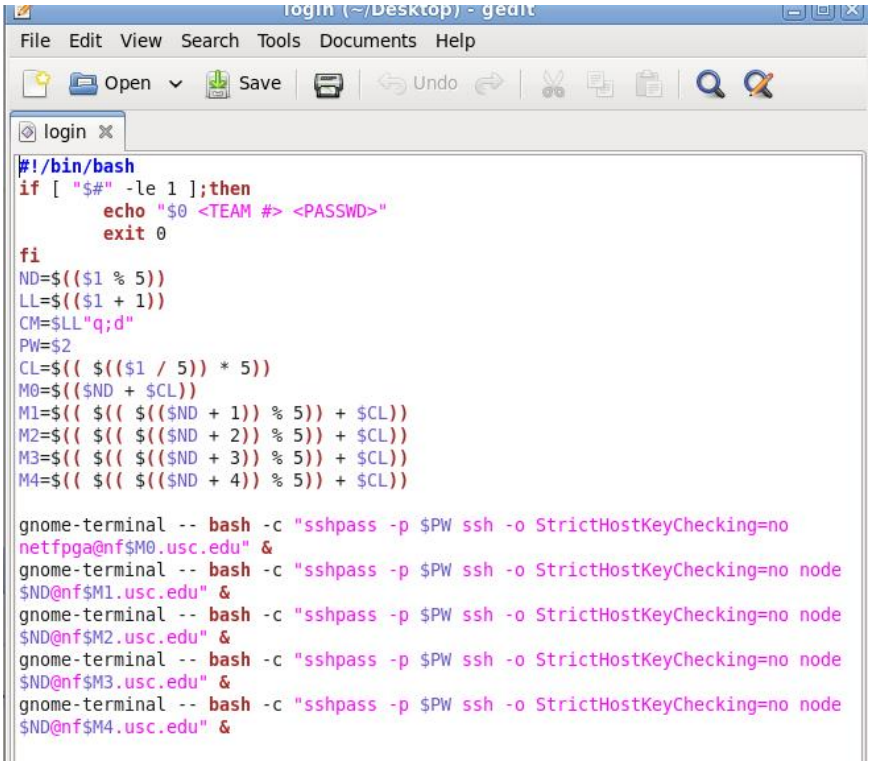
Part 4: NetFPGA Environment

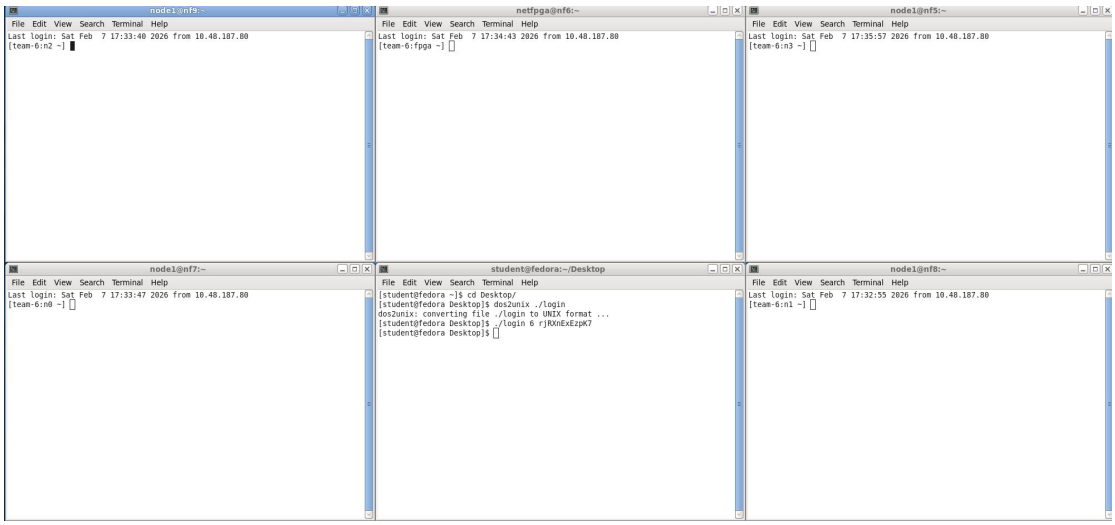
Connect by manual:



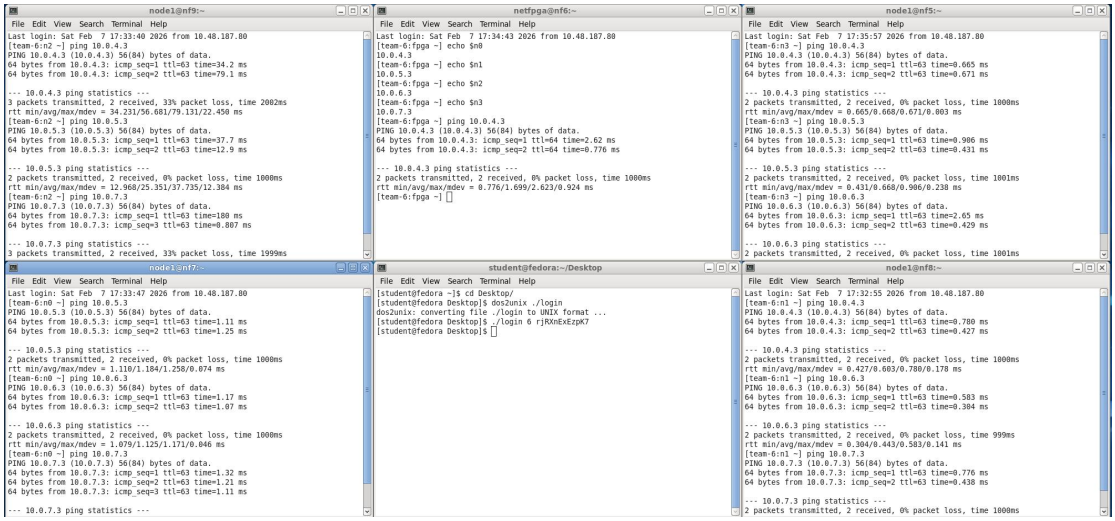


Then we use a script to open all the nodes.


login (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
login
#!/bin/bash
if ["$#" -le 1];then
echo "$0 <TEAM #> <PASSWD>"
exit 0
fi
ND=$(($1 % 5))
LL=$(($1 + 1))
CM=$LL"q;d"
PW=$2
CL=$(($(($1 / 5)) * 5))
M0=$(($ND + $CL))
M1=$(($(($(($ND + 1)) % 5)) + $CL))
M2=$(($(($(($ND + 2)) % 5)) + $CL))
M3=$(($(($(($ND + 3)) % 5)) + $CL))
M4=$(($(($(($ND + 4)) % 5)) + $CL))
gnome-terminal -- bash -c "sshpass -p $PW ssh -o StrictHostKeyChecking=no
netfpga@nf$M0.usc.edu" &
gnome-terminal -- bash -c "sshpass -p $PW ssh -o StrictHostKeyChecking=no node
$ND@nf$M1.usc.edu" &
gnome-terminal -- bash -c "sshpass -p $PW ssh -o StrictHostKeyChecking=no node
$ND@nf$M2.usc.edu" &
gnome-terminal -- bash -c "sshpass -p $PW ssh -o StrictHostKeyChecking=no node
$ND@nf$M3.usc.edu" &
gnome-terminal -- bash -c "sshpass -p $PW ssh -o StrictHostKeyChecking=no node
$ND@nf$M4.usc.edu" &



For Connectivity Test:



Part5：（All scripts and logs are submitted on Github）

NiC TCP

Server: n0	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n1	16.3	12.3	0%
n2	8.52	6.80	0%
n3	24.2	22.0	0%

Server: n1	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	13.0	9.13	0%
n2	29.2	25.5	0%
n3	14.5	13.3	0%

Server: n2	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate

n0	11.0	4.88	0%
n1	10.5	6.22	0%
n3	7.72	7.32	0%

Server: n3	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	18.9	14.1	0%
n1	6.88	5.44	0%
n2	11.1	9.83	0%

NIC UDP

Server: n0	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n1	259	31.7	66%
n2	137000	23.5	32%
n3	239	22.6	89%

Server: n1	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	247	8.9	96%
n2	246	14.6	94%
n3	238	10.3	95%

Server: n2	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	247	6.48	97%
n1	259	7.32	97%
n3	257	6.71	98%

Server: n3	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	247	11.2	95%
n1	191	18.0	91%
n2	248	13.6	95%

Part 6:

(All scripts and logs are submitted on Github)

rkd

```
[team-6:fpga ~] rkd &
[2] 29815
[team-6:fpga ~] killall rkd
[1]-  Terminated              rkd
[2]+  Terminated              rkd
```

Router TCP

Server: n0	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n1	255	183	0%
n2	357	352	0%
n3	166	165	0%

Server: n1	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	474	474	0%
n2	334	332	0%
n3	264	259	0%

Server: n2	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	307	307	0%
n1	78.5	78.3	0%
n3	290	135	0%

Server: n3	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	37.7	37.7	0%
n1	319	318	0%
n2	122	122	0%

Router UDP

Server: n0	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n1	254	1.91	65%
n2	244	16.1	89%
n3	17400	28.1	90%

Server: n1	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	246	97.4	60%
n2	243	42.7	82%
n3	245	124	49%

Server: n2	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	247	15	94%
n1	259	19.9	92%
n3	247	9.69	96%

Server: n3	Client bandwidth (Mb/s)	Server bandwidth (Mb/s)	Packet loss rate
n0	247	15.5	94%
n1	259	19.0	93%

n2	243	8.04	97%

Experiment Configuration (UDP)

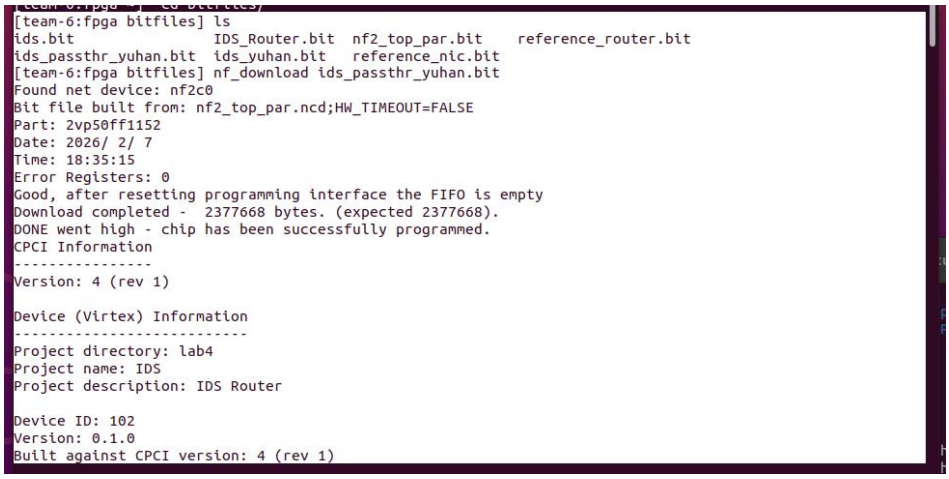
In this experiment, iperf is used in UDP mode to evaluate the forwarding performance of the NetFPGA. An iperf server running in UDP mode is launched on each node to receive traffic from other nodes. According to the experiment requirements, the packet size used by the iperf clients is fixed at 512 bytes. Using small packets significantly increases the packet processing rate, which places greater stress on the NetFPGA data path and forwarding logic.

For traffic generation, each NetFPGA port is configured to carry approximately 1 Gbps of traffic, with simultaneous traffic in both directions. Each iperf test runs for at least 30 seconds to ensure stable and reliable measurements. The same test is repeated multiple times, and the final performance results are obtained by averaging the measurements across all runs.

Part 7: Initial Attempt in Synthesizing and Testing Your Design

(All scripts are submitted on Github)

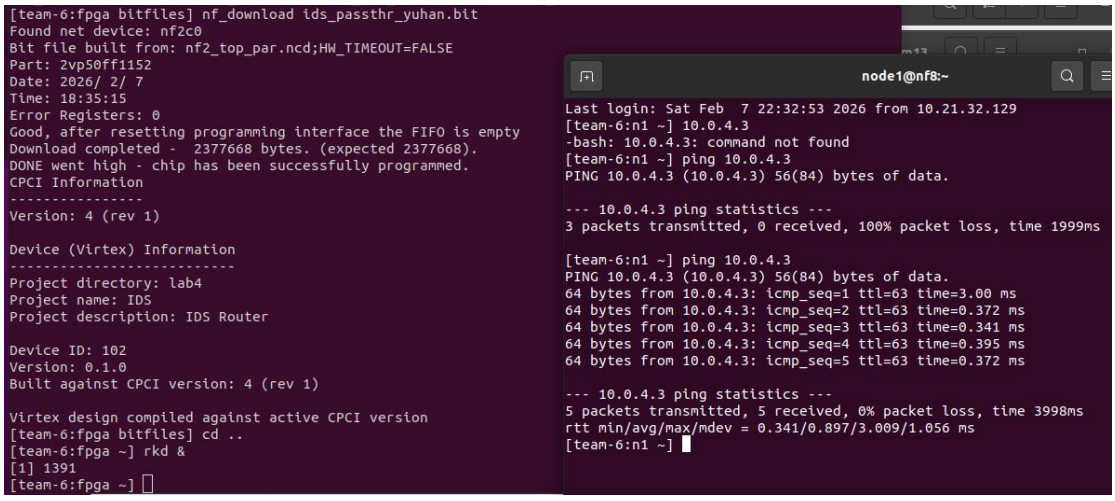
In Part 7, we completed both the passthrough version of the IDS and the normal version of the IDS. We also implemented an IDS with logic analysis. This figure shows the programming screenshot for the passthrough version of the IDS.



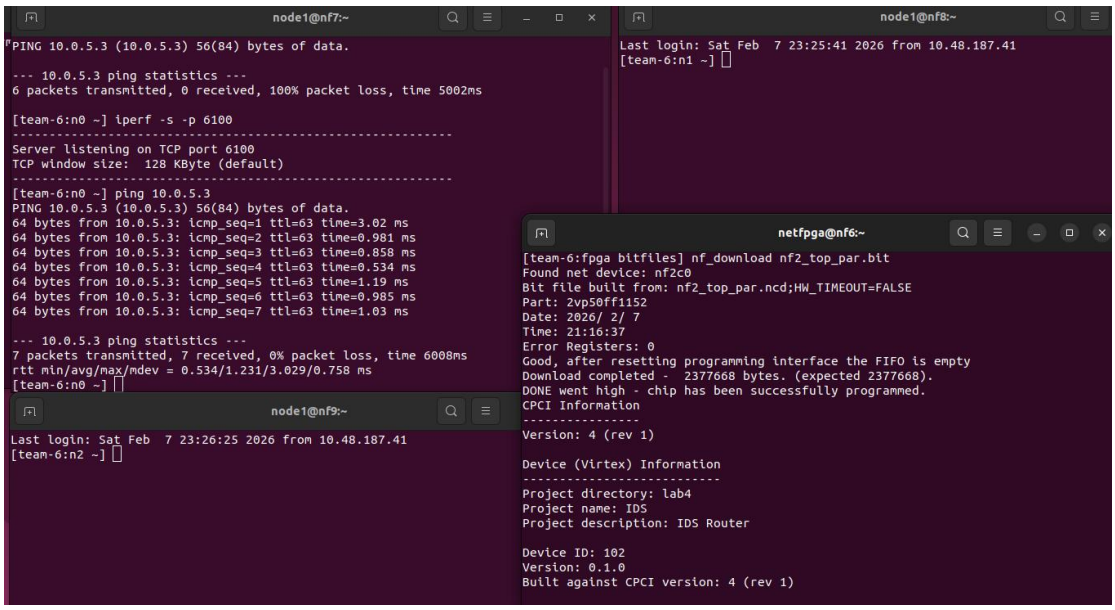
This figure shows the programming screenshot for the normal version of the IDS.

From this figure, we can see that the IDS software registers can correctly store the target pattern and the related data.

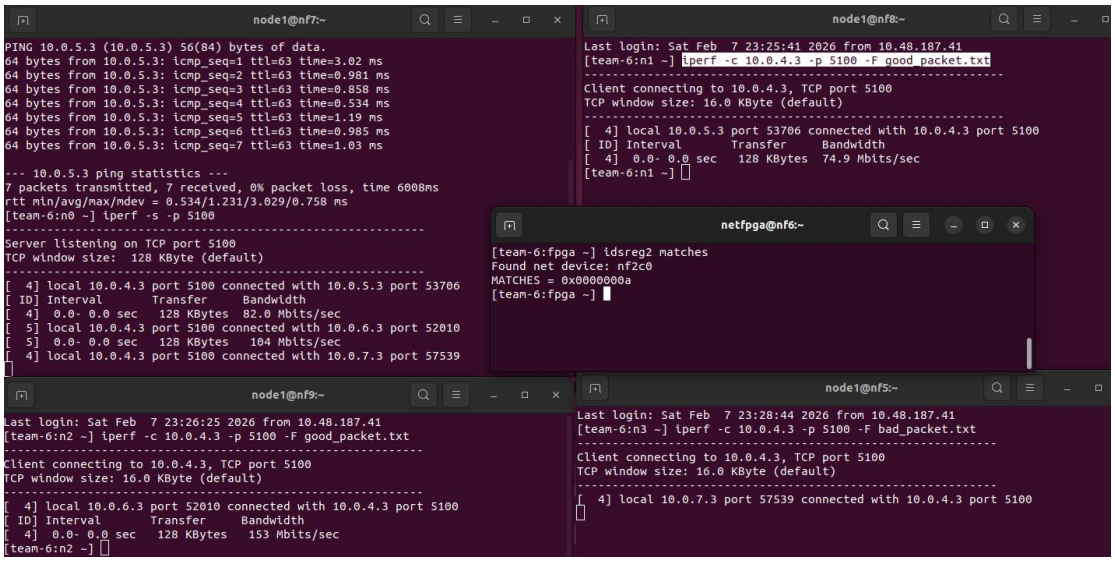
The passthrough IDS can successfully forward packets.



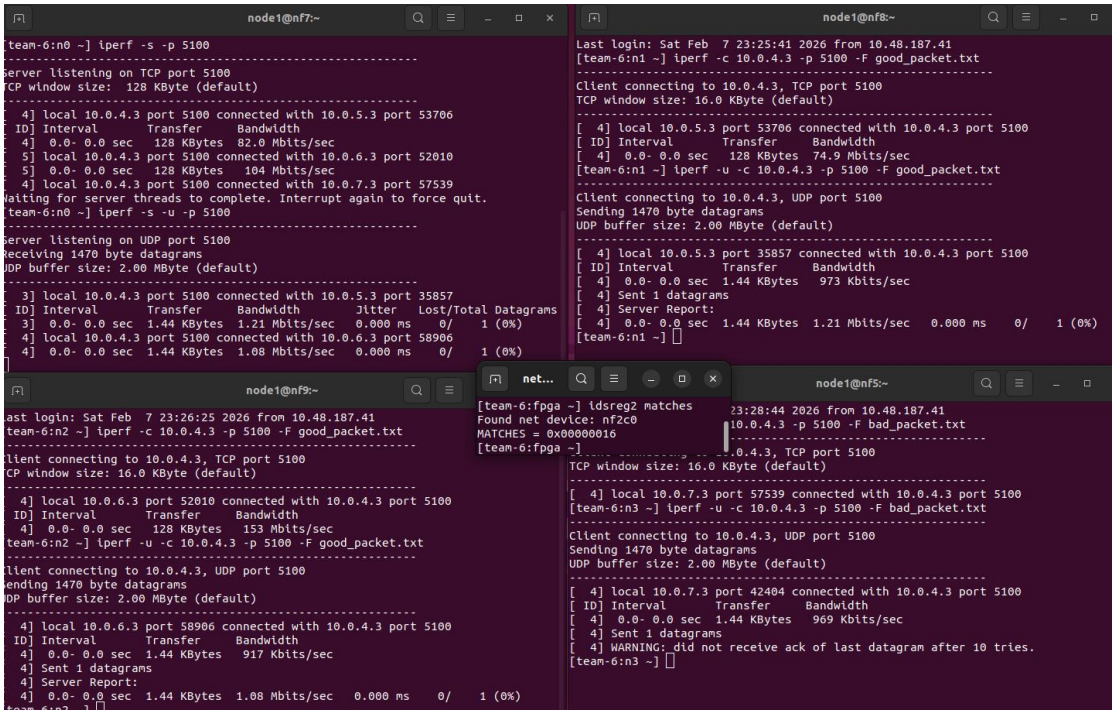
The same normal IDS module can also perform forwarding functions.



Similarly, we observed an increase in the matches register, indicating that TCP format forwarding is also possible, even with a corrupted packet.



The following are the results for UDP.


node1@nf7:~
team-6:n0 ~] iperf -s -p 5100
Server listening on TCP port 5100
TCP window size: 128 KByte (default)
4] local 10.0.4.3 port 5100 connected with 10.0.5.3 port 53706
ID] Interval Transfer Bandwidth
4] 0.0- 0.0 sec 128 KBytes 82.0 Mbits/sec
5] local 10.0.4.3 port 5100 connected with 10.0.6.3 port 52010
5] 0.0- 0.0 sec 128 KBytes 104 Mbits/sec
4] local 10.0.4.3 port 5100 connected with 10.0.7.3 port 57539
Waiting for server threads to complete. Interrupt again to force quit.
team-6:n0 ~] iperf -s -u -p 5100
Server listening on UDP port 5100
Receiving 1470 byte datagrams
UDP buffer size: 2.00 MByte (default)
3] local 10.0.4.3 port 5100 connected with 10.0.5.3 port 35857
ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
3] 0.0- 0.0 sec 1.44 KBytes 1.21 Mbits/sec 0.000 ms 0/ 1 (0%)
4] local 10.0.4.3 port 5100 connected with 10.0.6.3 port 58906
4] 0.0- 0.0 sec 1.44 KBytes 1.08 Mbits/sec 0.000 ms 0/ 1 (0%)
node1@nf8:~
Last login: Sat Feb 7 23:25:41 2026 from 10.48.187.41
[team-6:n1 ~] iperf -c 10.0.4.3 -p 5100 -F good_packet.txt
Client connecting to 10.0.4.3, TCP port 5100
TCP window size: 16.0 KByte (default)
[4] local 10.0.5.3 port 53706 connected with 10.0.4.3 port 5100
[ID] Interval Transfer Bandwidth
[4] 0.0- 0.0 sec 128 KBytes 74.9 Mbits/sec
[team-6:n1 ~] iperf -u -c 10.0.4.3 -p 5100 -F good_packet.txt
Client connecting to 10.0.4.3, UDP port 5100
Sending 1470 byte datagrams
UDP buffer size: 2.00 MByte (default)
[4] local 10.0.5.3 port 35857 connected with 10.0.4.3 port 5100
[ID] Interval Transfer Bandwidth
[4] 0.0- 0.0 sec 1.44 KBytes 973 Kbits/sec
[4] Sent 1 datagrams
[4] Server Report:
[4] 0.0- 0.0 sec 1.44 KBytes 1.21 Mbits/sec 0.000 ms 0/ 1 (0%)
[team-6:n1 ~]
node1@nf9:~
ast login: Sat Feb 7 23:26:25 2026 from 10.48.187.41
team-6:n2 ~] iperf -c 10.0.4.3 -p 5100 -F good_packet.txt
lient connecting to 10.0.4.3, TCP port 5100
CP window size: 16.0 KByte (default)
4] local 10.0.6.3 port 52010 connected with 10.0.4.3 port 5100
ID] Interval Transfer Bandwidth
4] 0.0- 0.0 sec 128 KBytes 153 Mbits/sec
team-6:n2 ~] iperf -u -c 10.0.4.3 -p 5100 -F good_packet.txt
lient connecting to 10.0.4.3, UDP port 5100
ending 1470 byte datagrams
DP buffer size: 2.00 MByte (default)
4] local 10.0.6.3 port 58906 connected with 10.0.4.3 port 5100
ID] Interval Transfer Bandwidth
4] 0.0- 0.0 sec 1.44 KBytes 917 Kbits/sec
4] Sent 1 datagrams
4] Server Report:
4] 0.0- 0.0 sec 1.44 KBytes 1.08 Mbits/sec 0.000 ms 0/ 1 (0%)
net...
[team-6:fpga ~] idsreg2 matches
Found net device: nf2c0
MATCHES = 0x00000016
[team-6:fpga ~]
node1@nf5:~
23:28:44 2026 from 10.48.187.41
10.0.4.3 -p 5100 -F bad_packet.txt
0.4.3, TCP port 5100
TCP window size: 16.0 KByte (default)
[4] local 10.0.7.3 port 57539 connected with 10.0.4.3 port 5100
[team-6:n3 ~] iperf -u -c 10.0.4.3 -p 5100 -F bad_packet.txt
Client connecting to 10.0.4.3, UDP port 5100
Sending 1470 byte datagrams
UDP buffer size: 2.00 MByte (default)
[4] local 10.0.7.3 port 42404 connected with 10.0.4.3 port 5100
[ID] Interval Transfer Bandwidth
[4] 0.0- 0.0 sec 1.44 KBytes 969 Kbits/sec
[4] Sent 1 datagrams
[4] WARNING: did not receive ack of last datagram after 10 tries.
[team-6:n3 ~]