

QUIC1GC-IP Demo Instruction

1	Environment setup	3
2	PC setup	4
2.1	IP setting	4
2.2	Speed and duplex settings	5
3	aiouic server.....	6
3.1	Run aiouic.....	6
3.2	Run Chrome web browser	8
3.3	Download with method get	9
3.4	Upload with method POST	10
4	MsQuic server.....	11
5	QUIC1GC demo setup.....	12
6	Nios command shell.....	13
7	Command detail.....	14
7.1	Set Gateway IP address	14
7.2	Set FPGA IP address.....	14
7.3	Set FPGA MAC address	14
7.4	Load FPGA network parameters	14
7.5	Set FPGA port number	14
7.6	Enable showkey mode	14
7.7	Enable showcert mode	15
7.8	Enable showsessionparams mode.....	17
7.9	GET method	18
7.10	POST method	21
7.11	PERF method	22
8	Graphic user interface	23
8.1	Connect with FPGA device.....	24
8.2	Set demo parameters	25
8.3	Test with myget command	26
8.4	Test with myperf command.....	27
9	Revision History	28

QUIC1GC-IP Demo Instruction

Rev1.01 24-Feb-2026

This document provides detailed instructions to demonstrate the use of the QUIC Client 1Gbps IP core (QUIC1GC-IP) for 1 Gigabit Ethernet, referred to “QUIC1GC-IP Reference Design”, using the FPGA Evaluation Board. The QUIC1GC-IP is used as a medium to transfer data within a secure connection following the QUIC transport protocol version 1 standard (RFC9000). This process involves handling the TLS 1.3 handshake and dealing with data encryption and decryption.

The reference design uses the QUIC1GC-IP and manages the application layer of the IP. It is tailored to test the IP functionality, help users understand how to use the IP, and to offer flexibility for users in case they need to modify the design. There are two main applications demonstrated in this reference design: one is HTTP/3, as the application layer to streamline the HTTP data, and the other is a unique application protocol designed by an organization to use with their application.

This instruction will explain step-by-step how users can utilize the QUIC1GC-IP for uploading and downloading data from two examples. aioquic is the first example, used to perform as a server using the HTTP/3, and the results are similar to those achieved by a web browser. Also, MsQuic is employed as a server to show the transfer performance using the unique application protocol.

This document is structured into the following key topics to guide you through the demo:

1. Environment & PC Setup (Sections 1-2)

Covers hardware requirements and necessary network configurations on the host PC. Covers hardware requirements and necessary network configurations on the host PC.

2. Server Configuration (Sections 3-4)

Provides detailed setup for the aioquic (HTTP/3) and MsQuic (Performance) servers used in the demo.

3. FPGA, Command Operations and Interface (Sections 5-8)

Explains how to program the FPGA and how to interact with the IP core. This includes using the Nios Command Shell for manual command entry and the Graphic User Interface (GUI) for a visual experience.

1 Environment setup

To run the QUIC1GC-IP demo, please prepare following test environment.

- 1) FPGA development board (Arria10 SoC, Sulfur Agilex 5 E-Series board).
- 2) Test PC with 1 Gigabit Ethernet port or connecting with 1 Gigabit Ethernet card.
- 3) 1 Gb Ethernet cable: Cat5e or Cat6 cable to connect the FPGA to the PC.
- 4) Micro USB cable for JTAG connection connecting between FPGA board and Test PC.
- 5) Quartus Programmer and Nios Command Shell, installed on PC.
- 6) Demo configuration file (To download these files, please visit our website at www.design-gateway.com).

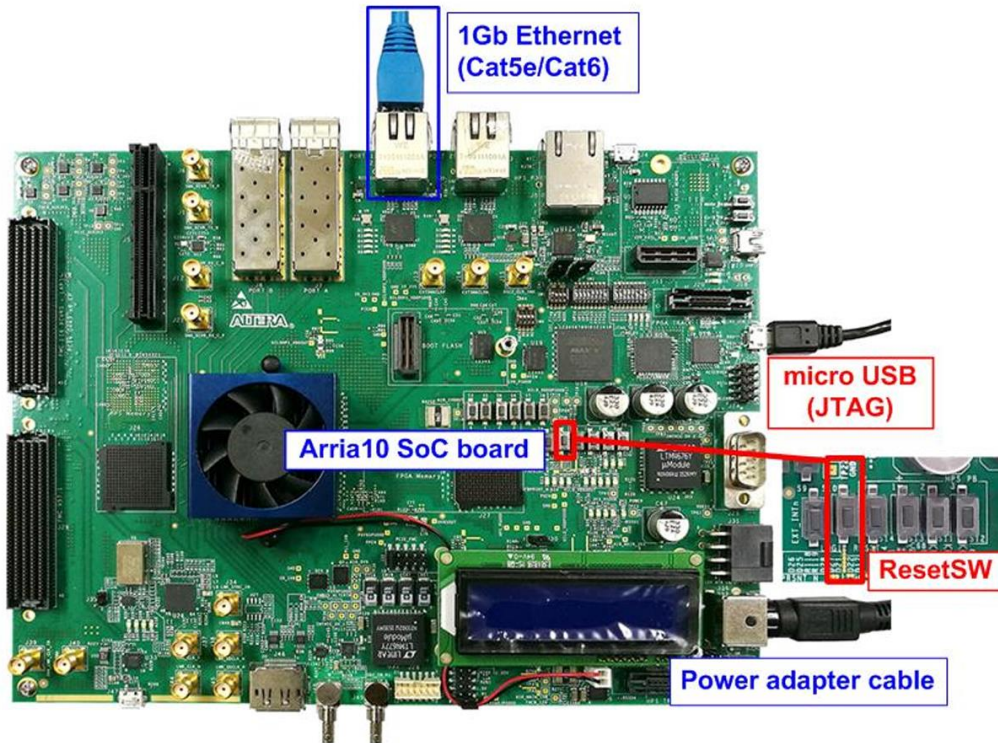


Figure 1 QUIC1GCIP demo environment on A10SoC board

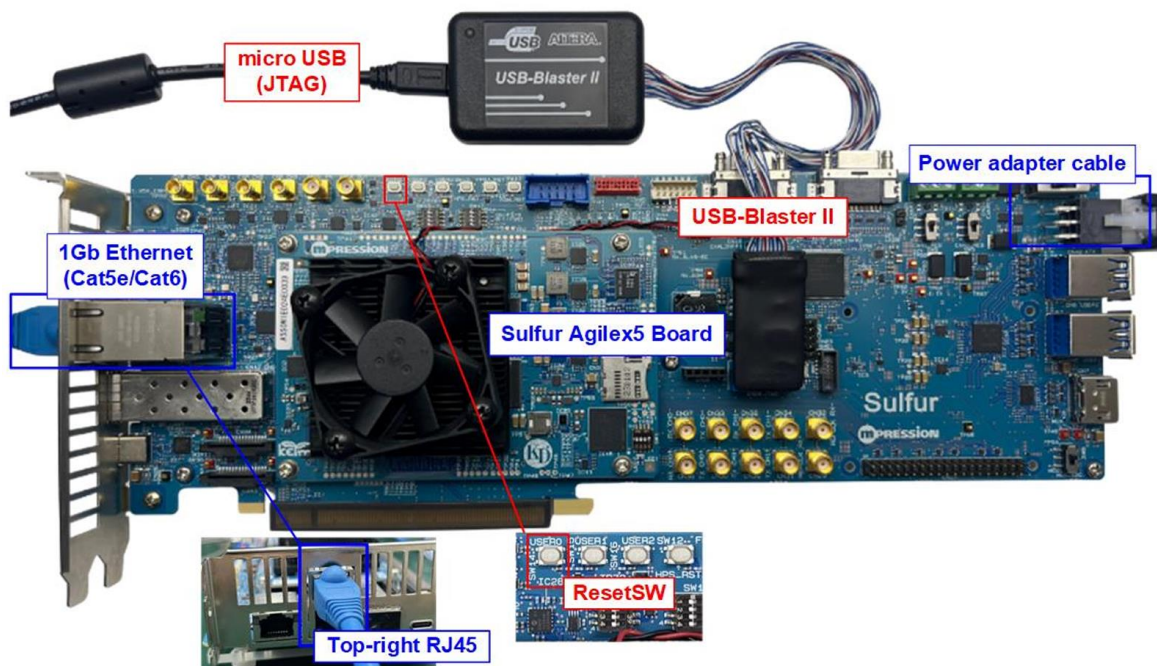


Figure 2 QUIC1GCIP demo environment on Sulfur Agilex 5 E-Series board

2 PC setup

Before running demo, please check the network setting on PC. The example of setting 1 Gb Ethernet connection is described as follows.

2.1 IP setting

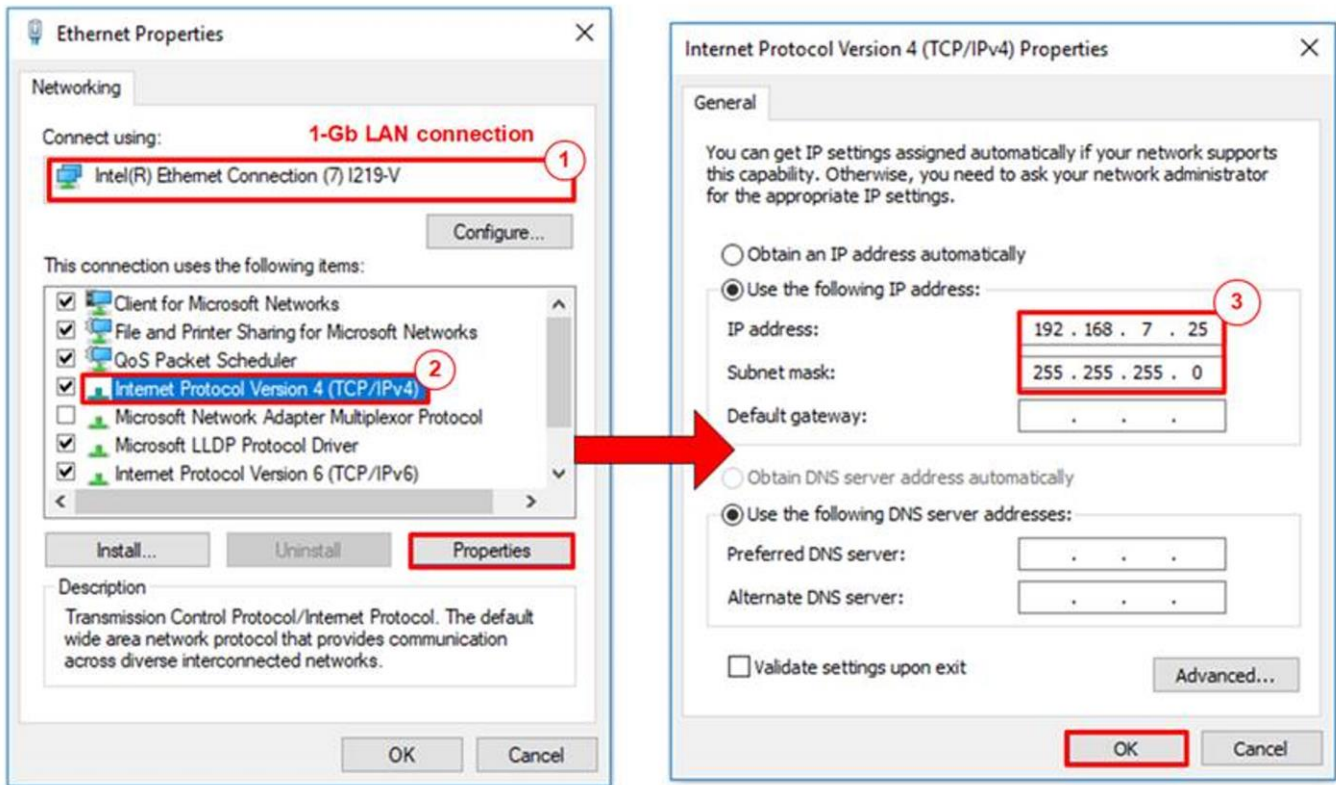


Figure 3 Setting IP address for PC

- 1) Open Local Area Connection Properties of 1 Gb connection, as shown in the left window of Figure 3.
- 2) Select "TCP/IPv4" and then click Properties.
- 3) Set IP address = 192.168.7.25 and Subnet mask = 255.255.255.0, as shown in the right window of Figure 3.

2.2 Speed and duplex settings

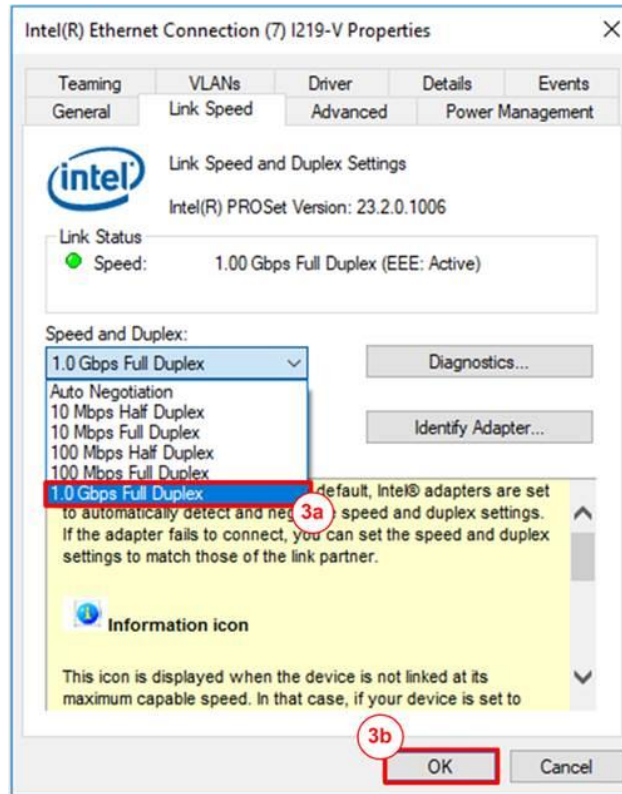


Figure 4 Set Link Speed = 1 Gbps

- 1) On Local Area Connection Properties window, click “Configure”, as shown in Figure 4.
- 2) On Advanced Tab, select “Speed and Duplex”. Set the value to “1.0 Gbps Full Duplex” for running 1 Gigabit transfer test, as shown in Figure 4.

3 aioquic server

aioquic is an open-source implementation of QUIC and HTTP/3 in Python. This demonstration is based on aioquic version 1.0.0*, which is available in our forked repository: github.com/design-gateway/aioquic.

Since QUIC1GC-IP supports two signature algorithms, `rsa_pss_rsae_sha256` and `ecdsa_secp256r1_sha256`, the key and certificate must correspond to either

- an RSA key, or
- an ECDSA key using the `secp256r1` curve.

In this demonstration, two example key and certificate pairs are provided. By default, the example files use RSA (`tls_key.pem` and `tls_cert.pem`).

For testing with ECDSA, alternative files are also provided as `ecdsa_key.pem` and `ecdsa_cert.pem`. To switch the aioquic server to use ECDSA, replace:

- `tls_key.pem` with `ecdsa_key.pem`
- `tls_cert.pem` with `ecdsa_cert.pem`

Note: The original aioquic can be found at: github.com/aioquic/aioquic.

To run this demonstration, the following modifications have been made to our fork:

- 1) Replaced 'examples/templates/index.html' (the default aioquic index page) with Design Gateway's html file.
- 2) Added sample key and certificate files (`tls_key.pem` and `tls_cert.pem`) in the 'tests/' directory.
- 3) Added an html file, 'tests/httpEcho.html', to demonstrate one functionality of aioquic.

If you have any questions regarding the aioquic core implementation, please contact the aioquic development team directly.

3.1 Run aioquic

'examples/http3_server.py' is used as an example to run a QUIC demo server. To run this server, certain parameters are required, all of which you can find using the help command, as shown in Figure 5. This information shows the available options and usage instructions for the aioquic HTTP/3 server, allowing users to configure parameters such as the TLS certificate, private key, host address, port, and other settings relevant to the QUIC server operation.

```
D:\Chromium\ aioquic>py examples/http3_server.py -h
usage: http3_server.py [-h] -c CERTIFICATE [--congestion-control-algorithm CONGESTION_CONTROL_ALGORITHM] [--host HOST]
                    [--port PORT] [-k PRIVATE_KEY] [-l SECRETS_LOG] [--max-data MAX_DATA]
                    [--max-stream-data MAX_STREAM_DATA] [--max-datagram-size MAX_DATAGRAM_SIZE] [-q QUIC_LOG]
                    [--retry] [-v]
                    [app]

QUIC server

positional arguments:
  app                  the ASGI application as <module>:<attribute>

options:
  -h, --help            show this help message and exit
  -c CERTIFICATE, --certificate CERTIFICATE
                        load the TLS certificate from the specified file
  --congestion-control-algorithm CONGESTION_CONTROL_ALGORITHM
                        use the specified congestion control algorithm
  --host HOST           listen on the specified address (defaults to :)
  --port PORT          listen on the specified port (defaults to 4433)
  -k PRIVATE_KEY, --private-key PRIVATE_KEY
                        load the TLS private key from the specified file
  -l SECRETS_LOG, --secrets-log SECRETS_LOG
                        log secrets to a file, for use with Wireshark
  --max-data MAX_DATA  connection-wide flow control limit (default: 1048576)
  --max-stream-data MAX_STREAM_DATA
                        per-stream flow control limit (default: 1048576)
  --max-datagram-size MAX_DATAGRAM_SIZE
                        maximum datagram size to send, excluding UDP or IP overhead
  -q QUIC_LOG, --quic-log QUIC_LOG
                        log QUIC events to QLOG files in the specified directory
  --retry              send a retry for new connections
  -v, --verbose        increase logging verbosity
```

Figure 5 aioquic console with the help command

As depicted in, an aioquic server is operated by running the python file on the terminal with certain options. This example uses the provided key and certificate files, binds to an existed address on the machine, and logs the secrets in a text file.

```

D:\Chromium\aiouic>py examples/http3_server.py --host 192.168.7.25 --certificate tests/tls_cert.pem --private-key tests
/tls_key.pem -l C:\Users\tan\sslkey\sslkeylog.txt
2024-05-29 10:31:06,329 INFO quic [37cc129c68e544ac] Duplicate CRYPTO data received for epoch Epoch. INITIAL
2024-05-29 10:31:06,330 INFO quic [37cc129c68e544ac] Duplicate CRYPTO data received for epoch Epoch. INITIAL
2024-05-29 10:31:06,330 INFO quic [37cc129c68e544ac] Duplicate CRYPTO data received for epoch Epoch. INITIAL
2024-05-29 10:31:06,331 INFO quic [37cc129c68e544ac] Duplicate CRYPTO data received for epoch Epoch. INITIAL
2024-05-29 10:31:06,331 INFO quic [37cc129c68e544ac] Duplicate CRYPTO data received for epoch Epoch. INITIAL
2024-05-29 10:31:06,344 INFO quic [37cc129c68e544ac] ALPN negotiated protocol h3
2024-05-29 10:31:06,350 INFO quic [37cc129c68e544ac] HTTP request GET /
2024-05-29 10:31:06,396 INFO quic [37cc129c68e544ac] HTTP request GET /style.css
2024-05-29 10:31:06,523 INFO quic [37cc129c68e544ac] HTTP request GET /favicon.ico

```

Figure 6 Example of running the aioquic HTTP/3 server

3.2 Run Chrome web browser

A typical web browser can be used to communicate with the aioquic server. The process involves using the GET method to download data from the server and using the POST method to upload data to the server. To access the demo server running on the local machine, launch Chromium or Chrome with the following command:

```
<path to chrome.exe> --enable-experimental-web-platform-features \
--ignore-certificate-errors-spki-list=<SPKI> \
--origin-to-force-quic-on=<server ip:server port> \
url path
```

```
C:\Users\tan>"c:\Program Files\Google\Chrome\Application\chrome.exe" --enable-experimental-web-platform-features --ignore-certificate-errors-spki-list=2kaKWhS2v9++0KSEdx5JfwzGs6QM3cBv0R9OZTM/GeI= --origin-to-force-quic-on=192.168.7.25:4433 https://192.168.7.25:4433/
```

Figure 7 Example command line to run Chrome with aioquic server

The certificate used in this demonstration is self-signed, meaning it was not issued by a certification authority (CA). When attempting to access the server with a self-signed certificate, the web browser may generate a certificate unknown error and terminate the connection. To bypass certificate errors, users can run the Chrome browser from the command prompt with the '--ignore-certificate-errors-spki-list' flag, specifying the certificate's SPKI (Subject Public Key Info). This allows Chrome/Chromium to accept the self-signed certificate as valid. Users can generate the SPKI with the following command:

```
openssl x509 -noout -pubkey -in tls_cert.pem |^
openssl pkey -pubin -outform der |^
openssl dgst -sha256 -binary |^
openssl base64
```

```
D:\Chromium\ aioquic\tests>openssl x509 -noout -pubkey -in tls_cert.pem | openssl pkey -pubin -outform der | openssl dgst -sha256 -binary | openssl base64
2kaKWhS2v9++0KSEdx5JfwzGs6QM3cBv0R9OZTM/GeI=
```

Figure 8 Example command line to calculate public key hash from certificate

When launching Chrome with the '--ignore-certificate-errors-spki-list' flag, users may encounter a message indicating that Chrome is running with an unsupported command-line flag. The flag '--ignore-certificate-errors-spki-list' is used to bypass SSL/TLS certificate errors by specifying a list of SPKI hashes. This can be useful for testing purposes but is not recommended for regular use due to potential security and stability risks.



Figure 9 Example of encountering the --ignore-certificate-errors-spki-list flag

Remark: Our tested web browser is Google Chrome version 125.0.6422.113.

3.3 Download with method get

The aioquic demo example supports HTTP/3, allowing users to download data from the aioquic server using a web browser via the GET method. To download data, the URL required as an input must follow this format:

https://ip:port/length

- Where ip represents server's IP address in dot-decimal notation.
- port represents server's port number.
- length represents data length in byte (or leave blank to get the homepage).

For example, if the server's IP address is 192.168.7.25 and the port number is 4433, the URL must be https://192.168.7.25:4433/ to establish a secure connection and display the aioquic homepage in the web browser, as illustrated in Figure 11.

```
2024-05-27 16:00:03,396 INFO quic [95110ffd4223fd08] ALPN negotiated protocol h3
2024-05-27 16:00:03,410 INFO quic [95110ffd4223fd08] HTTP request GET /
2024-05-27 16:00:03,865 INFO quic [95110ffd4223fd08] HTTP request GET /favicon.ico
```

Figure 10 aioquic console when the client downloads the homepage



Figure 11 Download the index html using web browser

To download data with a specific length, user adds a size in byte unit at the end of the URL. For example, https://192.168.7.25:4433/500 is used as the URL to download 500-byte data, which will be displayed in the web browser, as shown in Figure 12.

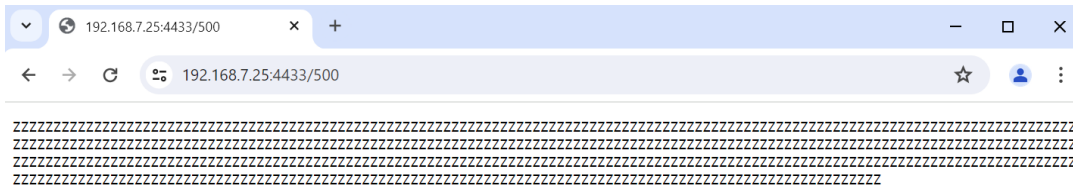


Figure 12 Download data pattern shown in the web browser

```
2024-05-27 16:01:28,719 INFO quic [05b3bbfb94816be4] HTTP request GET /500
2024-05-27 16:01:28,721 INFO quic [05b3bbfb94816be4] ALPN negotiated protocol h3
```

Figure 13 aioquic console when the client downloads data pattern

3.4 Upload with method POST

αιοquic also offers a method to upload data to an aιοquic server. By using a web browser, an application named 'httpEcho.html' is provided for uploading data in a secure connection with an aιοquic server.

Users can open the webpage of this application by simply dragging and dropping the html file into a browser. At this point, a user interface appears which allows users to set parameters, such as server's IP address and port number, and to input the data message to be uploaded to the server, as shown in Figure 14. After setting the inputs, users press the "Send" button in order to send a POST command to the aιοquic server with the URL endpoint "/echo".

Although the aιοquic doesn't support a direct POST method by showing the data message at their side, it returns the data message back to the sender, the web browser in this case. As a result, the echoed data is displayed in the web browser, as depicted in Figure 15.

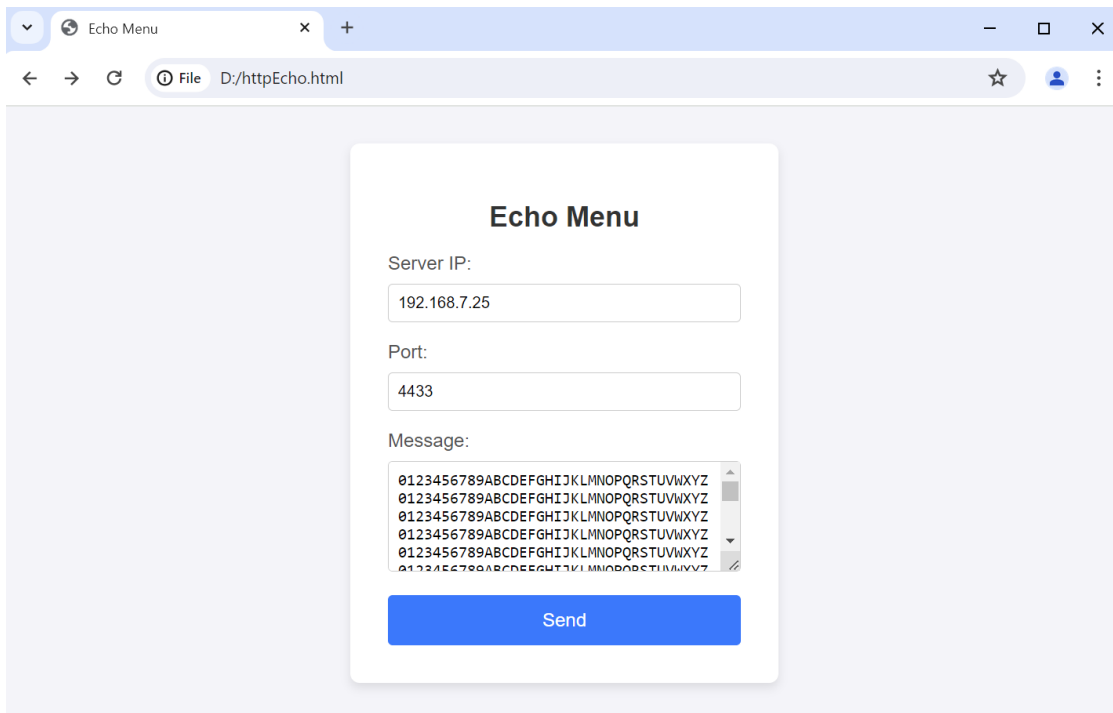


Figure 14 Upload a message via a web browser



Figure 15 Echo Data from the aιοquic server shown in the web browser

```
2024-05-27 16:15:26,232 INFO quic [a9d17b5558b939aa] ALPN negotiated protocol h3
2024-05-27 16:15:26,239 INFO quic [a9d17b5558b939aa] HTTP request POST /echo
```

Figure 16 aιοquic console when the client uploads data

4 MsQuic server

The second QUIC implementation used in this reference design is MsQuic, an open-source project developed by Microsoft, written in C and available at github.com/microsoft/msquic. This demo utilizes MsQuic version 2.3.5. We would like to express our gratitude to Microsoft and the MsQuic team for their contributions to the QUIC ecosystem.

While no modifications to the original MsQuic code are required to run this demo, we provide a fork of the repository as a reference branch here: github.com/design-gateway/msquic. Please note that technical inquiries regarding the MsQuic core should be directed to the official MsQuic development team.

The secretperf Application

Among the various examples offered by MsQuic, this demo uses the 'secretperf' application. It is specifically optimized for high-performance data transfer and utilizes its own dedicated application protocol rather than HTTP/3.

To set up and run the MsQuic server using secretperf, follow these steps:

1. Execution: Run secretperf.exe using three required parameters:
 - IP Address: The address the server will bind to.
 - Port Number: The port the server will listen on.
 - Profile Setting: Configured for maximum performance in this demo.
2. Verification: Once executed, the console will display the message "Started!". No further messages will appear unless an error occurs.

The example of command execution and successful start of the MsQuic server are illustrated in Figure 17.

```
PS D:\37.QUIC10GC\msquic> ./artifacts/bin/windows/x64_Debug_openssl/secretperf.exe -exec:maxtput -ip:192.168.7.25 -port:4432
Started!
```

Figure 17 MsQuic server application console

Once the server is started, a client running the secretperf application can be connected. In this example, the client is executed using five primary options:

- -target: Specifies the IP address of the server to connect to.
- -port: Specifies the Port number of the server to connect to.
- -exec: Must match the server's execution profile setting.
- -upload / -download: Specifies the data length for the upload or download test.
- -ptput: Enables the printing of throughput information.

The console output for an MsQuic client uploading data to the server is illustrated in Figure 18, while the output for downloading data from the server is shown in Figure 19.

```
PS D:\37.QUIC10GC\msquic> ./artifacts/bin/windows/x64_Debug_openssl/secretperf.exe -exec:maxtput -target:192.168.7.25
-port:4432 -up:1gb -ptput:1
Started!

Result: Upload 1000000000 bytes @ 3173511 kbps (2520.867 ms).
```

Figure 18 MsQuic client application console uploading data

```
PS D:\37.QUIC10GC\msquic> ./artifacts/bin/windows/x64_Debug_openssl/secretperf.exe -exec:maxtput -target:192.168.7.25
-port:4432 -down:1gb -ptput:1
Started!

Result: Download 1000000000 bytes @ 3543217 kbps (2257.835 ms).
```

Figure 19 MsQuic client application console downloading data

5 QUIC1GC demo setup

- 1) Make sure power switch is off and connect power supply to FPGA development board.
- 2) Connect USB cables between FPGA board and PC via micro-USB ports.
- 3) Turn on power switch for FPGA board.
- 4) Open Quartus Programmer to program FPGA through USB-1 by following step.
 - i. Click “Hardware Setup...” to select
 - USB-BlasterII [USB-1] for Arria10 SoC
 - ii. Click “Auto Detect” and select FPGA number.
 - iii. Select FPGA device icon.
 - iv. Click “Change File” button, select SOF file in pop-up window and click “open” button.
 - v. Check “program”.
 - vi. Click “Start” button to program FPGA.
 - vii. Wait until Progress status is equal to 100%.

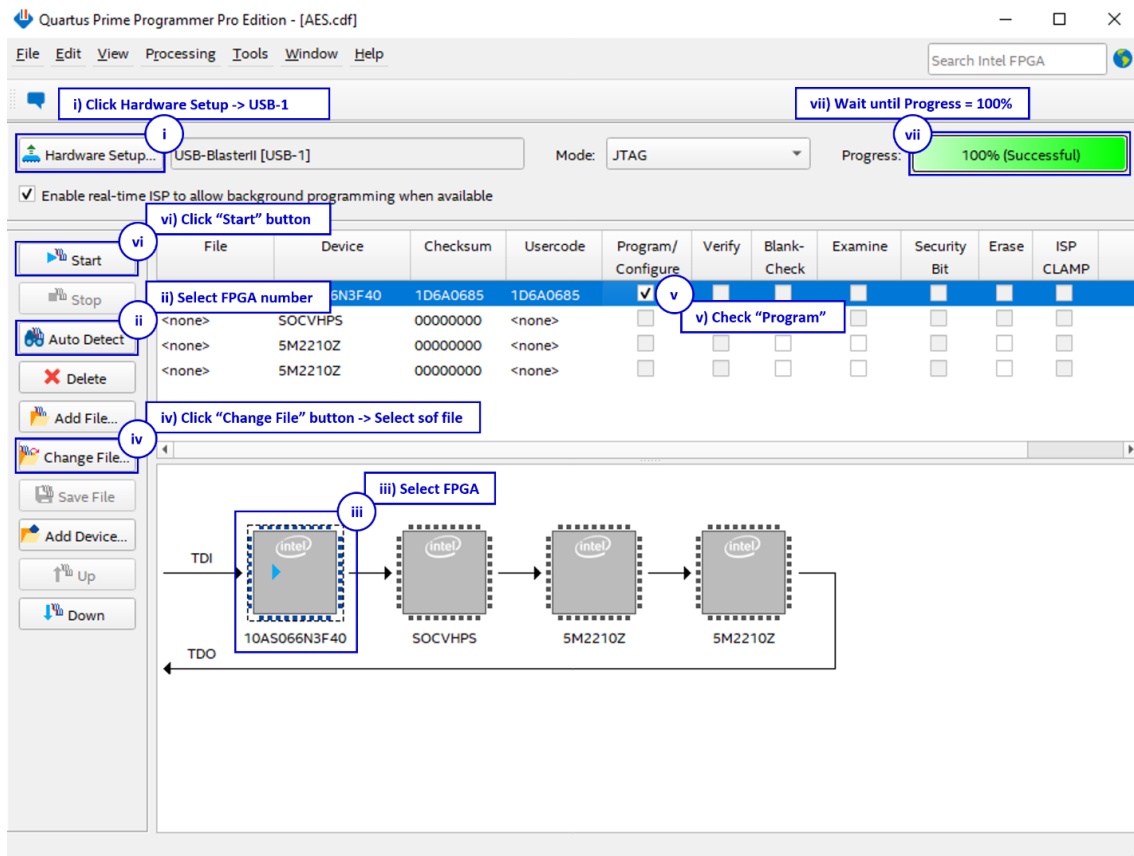


Figure 20 Program Device

After program SOF file complete, Quartus Prime will show popup message of Intel FPGA IP Evaluation Mode Status as shown in Figure 2-3. Please do not press cancel button.

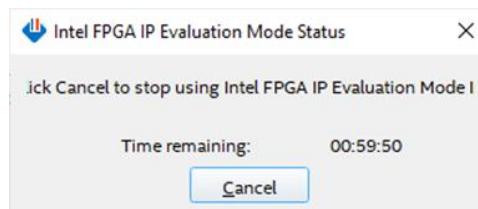


Figure 21 Intel FPGA IP Evaluation Mode Status

6 Nios command shell

To launch the FPGA console using the command shell. Run the following command:

For Nios II command shell >> nios2-terminal.exe

For Nios V command shell >> juart-terminal.exe

Then users can set the parameters or run the download and upload applications by using the following commands. The QUIC1GC demo commands and their usages will be displayed, as shown in Figure 22. Detailed information about each command is described in topic 7.

```

:/mnt/c/intelFPGA_pro/20.4$ nios2-terminal.exe
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-BlasterII [USB-1]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

=====
                QUIC1GC (version 0x80013440)
=====

Usage:
[0] setip ddd.ddd.ddd.ddd[/nn]
    Set FPGA's IP address in dotted-decimal format with optional CIDR notation.
[1] setgatewayip ddd.ddd.ddd.ddd
    Set Gateway IP address in dotted-decimal format.
[2] setport ddddd
    Set FPGA's port number in decimal format or type dynamic/d/-d to set dynamic port number.
[3] setmac hh-hh-hh-hh-hh-hh
    Set FPGA's MAC address in hexadecimal format.
[4] loadnetworkparameters
    Set to load all the network parameters necessary for the IP to initialize and connect to a network system.
[5] showkey <1: enable, 0: disable>
    Enable showkey mode for showing TLS traffic ticket, session key and iv for encryption/decryption.
[6] showsessionparams <1: enable, 0: disable>
    Enable showsessionparams mode for showing session parameters.
[7] showcert <1: enable, 0: disable>
    Enable showcert mode for showing certificate information.
[8] myGET https://host:port/path
    Send GET command for downloading data from server.
[9] myECHO https://host:port/echo size
    Send ECHO command for uploading and downloading pattern data with server.
[10] myPERF host:port uploadsize downloadsize
    Send PERF command for both downloading and uploading pattern data with SecNetPerf of msquic.

Type 'help' to view all available commands and usage.
Type '<command> help' to view usage details for a specific command.

>>

```

Figure 22 Nios II Command Shell

7 Command detail

7.1 Set Gateway IP address

```
command> setgatewayip <ddd.ddd.ddd.ddd>
```

This command is used to set the Gateway IP address in dotted-decimal format. The default Gateway IP address is 0.0.0.0, which indicates to the IP that there is no valid Gateway IP address. Users can input the setgatewayip command followed by a valid IP address, as shown in Figure 22.

7.2 Set FPGA IP address

```
command> setip <ddd.ddd.ddd.ddd>[/nn]
```

This command is used to set the FPGA's IP address in dotted-decimal format, with an optional subnet mask in CIDR notation. The default FPGA IP address is 192.168.7.42/24. Users can input the setip command followed by a valid IP address (e.g., 192.168.7.42) or an IP address with a CIDR suffix (e.g., 192.168.7.42/24), as shown in Figure 22. If the CIDR value is omitted, the default subnet mask of /24 is applied.

7.3 Set FPGA MAC address

```
command> setmac <hh-hh-hh-hh-hh-hh>
```

This command is used to set the FPGA's MAC address in hexadecimal format. The default FPGA's MAC address is 80-11-22-33-44-55, which is a unicast MAC address.

7.4 Load FPGA network parameters

```
command> loadnetworkparameters
```

This command is used to load all the network parameters necessary for the IP to initialize and connect to a network system. These include the Gateway IP address, FPGA's IP address, and FPGA's MAC address. The QUIC1GC-IP must have all network parameters loaded at least once after a power-on reset or when the IP core system reset is active.

7.5 Set FPGA port number

```
command> setport <dxxxx>
```

This command is used to set the static port number of FPGA in decimal format. By default, the FPGA's port number is set to be dynamic. Dynamic ports range from 49152 to 65535. Users can enable dynamic port again after specifying a port number by using "setport dynamic" command, as shown in Figure 22. It is worth noting that this command is not listed in the necessary network parameters, and therefore, it can be used to change the port number at any time before opening connection.

7.6 Enable showkey mode

```
command> showkey <1: enable, 0: disable>
```

This command is used to enable the showkey mode. When the showkey mode is enabled, the TLS traffic ticket for encryption/decryption is displayed on the serial console, as shown in Figure 23. Users can utilize the TLS traffic ticket in the (Pre)-Master-Secret log file for Wireshark*, enabling them to decrypt transferred data between the client and server.

*Wireshark, a network packet analyzer tool used for network troubleshooting, analysis, and security purposes.

```
>> myGET https://google.com
This connections is try 0-RTT
=====
Start IP initialization process
Handshake done
Traffic Secret
CLIENT_EARLY_TRAFFIC_SECRET 3112D215D317D456D456D5F2D5F2D6A0D6A0D789D789A7889F978D958A94C092 C5C3A20AE5B7A56B1FC0624DE442E7064B41278AE0C5092B15EEFA3802DCADFE
CLIENT_HANDSHAKE_TRAFFIC_SECRET 3112D215D317D456D456D5F2D5F2D6A0D6A0D789D789A7889F978D958A94C092 98A7ABB066AD47AA48E72B97815557946AD82CDC5FAA0ECF05940B77A2D10F35
SERVER_HANDSHAKE_TRAFFIC_SECRET 3112D215D317D456D456D5F2D5F2D6A0D6A0D789D789A7889F978D958A94C092 484194C28718EAA81A4010CAA7A59074BF601001D0230C39B4EA3DB680517DAB
CLIENT_TRAFFIC_SECRET_0 3112D215D317D456D456D5F2D5F2D6A0D6A0D789D789A7889F978D958A94C092 38D675BA3C5648D7105C6CE519C803DD725250FA88B781B66B716DA75D8C43EC
SERVER_TRAFFIC_SECRET_0 3112D215D317D456D456D5F2D5F2D6A0D6A0D789D789A7889F978D958A94C092 1BB6C0A31B68C85FA9C800C35B0EA3CE8F7401D01B192D47878DEFB274133B8
Loading...done
=====
```

Figure 23 Serial console when the showkey mode is enabled

7.7 Enable showcrt mode

command> showcrt <1: enable, 0: disable>

This command is used to enable the showcrt mode. When the showcrt mode is enabled, the server's certificate stored in RAM called CertRam is displayed on the serial console, as shown in Figure 24. The certificate information is displayed in hexadecimal format, which corresponds to the result obtained by using openssl command: openssl x509 -in tls_cert.pem -outform der | hexdump -C, as shown in Figure 25.

```

Certificate information
00000000 0B 00 01 EE 00 00 01 EA 00 01 E5 30 82 01 E1 30
00000010 82 01 87 A0 03 02 01 02 02 14 65 EC 9A A5 3E E2
00000020 BF EA 5B 87 7D 89 DB D8 16 A5 66 7E A8 8F 30 0A
00000030 06 08 2A 86 48 CE 3D 04 03 02 30 46 31 0B 30 09
00000040 06 03 55 04 06 13 02 54 48 31 10 30 0E 06 03 55
00000050 04 08 0C 07 62 61 6E 67 6B 6F 6B 31 0B 30 09 06
00000060 03 55 04 0A 0C 02 44 47 31 0B 30 09 06 03 55 04
00000070 0B 0C 02 44 47 31 0B 30 09 06 03 55 04 03 0C 02
00000080 53 49 30 1E 17 0D 32 34 30 39 32 34 30 32 31 36
00000090 30 31 5A 17 0D 32 35 30 39 32 34 30 32 31 36 30
000000A0 31 5A 30 46 31 0B 30 09 06 03 55 04 06 13 02 54
000000B0 48 31 10 30 0E 06 03 55 04 08 0C 07 62 61 6E 67
000000C0 6B 6F 6B 31 0B 30 09 06 03 55 04 0A 0C 02 44 47
000000D0 31 0B 30 09 06 03 55 04 0B 0C 02 44 47 31 0B 30
000000E0 09 06 03 55 04 03 0C 02 53 49 30 59 30 13 06 07
000000F0 2A 86 48 CE 3D 02 01 06 08 2A 86 48 CE 3D 03 01
00000100 07 03 42 00 04 47 16 DF DB 8E DF F5 D5 66 D7 97
00000110 C9 35 EF 58 88 EC 76 86 7B 70 C0 B6 99 67 9C 49
00000120 2A 9F 08 3F 6E E0 5C D6 AD 1D 13 11 B3 7E B8 52
00000130 78 08 DD 5A 0D 19 F0 10 1D DB 22 F0 95 99 CA FF
00000140 DC FE 55 E2 83 A3 53 30 51 30 1D 06 03 55 1D 0E
00000150 04 16 04 14 0C 88 88 3A 3D A7 AD 68 66 9E 93 75
00000160 6A 83 A1 72 71 65 5C 4F 30 1F 06 03 55 1D 23 04
00000170 18 30 16 80 14 0C 88 88 3A 3D A7 AD 68 66 9E 93
00000180 75 6A 83 A1 72 71 65 5C 4F 30 0F 06 03 55 1D 13
00000190 01 01 FF 04 05 30 03 01 01 FF 30 0A 06 08 2A 86
000001A0 48 CE 3D 04 03 02 03 48 00 30 45 02 21 00 89 61
000001B0 3B 8A DB A8 A5 C6 B6 56 B2 CA 2C E7 79 E5 B4 97
000001C0 96 92 47 62 16 C4 F9 A3 E2 FA 0C 31 AB E8 02 20
000001D0 3E AC 67 58 72 41 42 F7 52 1B 92 86 4C EC E3 A9
000001E0 06 1A CC 7F 02 B1 56 1A 1B 5B 83 7A F1 8B 3F C2
000001F0 00 00
  
```

Figure 24 Serial console when the showcrt mode is enabled

```

D:\>openssl x509 -in ecdsa_cert.pem -outform DER | hexdump -C
000000 30 82 01 e1 30 82 01 87 a0 03 02 01 02 02 14 65 0...0.....e
000010 ec 9a a5 3e e2 bf ea 5b 87 7d 89 db d8 16 a5 66 ...>...[.}.....f
000020 7e a8 8f 30 0a 06 08 2a 86 48 ce 3d 04 03 02 30 ~.0...*.H.=...0
000030 46 31 0b 30 09 06 03 55 04 06 13 02 54 48 31 10 F1.0...U...TH1.
000040 30 0e 06 03 55 04 08 0c 07 62 61 6e 67 6b 6f 6b 0...U...bangkok
000050 31 0b 30 09 06 03 55 04 0a 0c 02 44 47 31 0b 30 1.0...U...DG1.0
000060 09 06 03 55 04 0b 0c 02 44 47 31 0b 30 09 06 03 ...U...DG1.0...
000070 55 04 03 0c 02 53 49 30 1e 17 0d 32 34 30 39 32 U...SI0...24092
000080 34 30 32 31 36 30 31 5a 17 0d 32 35 30 39 32 34 4021601Z..250924
000090 30 32 31 36 30 31 5a 30 46 31 0b 30 09 06 03 55 021601Z0F1.0...U
0000a0 04 06 13 02 54 48 31 10 30 0e 06 03 55 04 08 0c ...TH1.0...U...
0000b0 07 62 61 6e 67 6b 6f 6b 31 0b 30 09 06 03 55 04 .bangkok1.0...U.
0000c0 0a 0c 02 44 47 31 0b 30 09 06 03 55 04 0b 0c 02 ...DG1.0...U...
0000d0 44 47 31 0b 30 09 06 03 55 04 03 0c 02 53 49 30 DG1.0...U...SI0
0000e0 59 30 13 06 07 2a 86 48 ce 3d 02 01 06 08 2a 86 Y0...*.H.=...*.
0000f0 48 ce 3d 03 01 07 03 42 00 04 47 16 df db 8e df H.=...B..G....
000100 f5 d5 66 d7 97 c9 35 ef 58 88 ec 76 86 7b 70 c0 ..f...5.X..v.{p.
000110 b6 99 67 9c 49 2a 9f 08 3f 6e e0 5c d6 ad 1d 13 ..g.I*...?n.\....
000120 11 b3 7e b8 52 78 08 dd 5a 0d 19 f0 10 1d db 22 ..~.Rx..Z....."
000130 f0 95 99 ca ff dc fe 55 e2 83 a3 53 30 51 30 1d .....U...S0Q0.
000140 06 03 55 1d 0e 04 16 04 14 0c 88 88 3a 3d a7 ad ..U.....:=..
000150 68 66 9e 93 75 6a 83 a1 72 71 65 5c 4f 30 1f 06 hf..uj..rqe\00..
000160 03 55 1d 23 04 18 30 16 80 14 0c 88 88 3a 3d a7 .U.#..0.....:=.
000170 ad 68 66 9e 93 75 6a 83 a1 72 71 65 5c 4f 30 0f .hf..uj..rqe\00.
000180 06 03 55 1d 13 01 01 ff 04 05 30 03 01 01 ff 30 ..U.....0....0
000190 0a 06 08 2a 86 48 ce 3d 04 03 02 03 48 00 30 45 ...*.H.=...H.0E
0001a0 02 21 00 89 61 3b 8a db a8 a5 c6 b6 56 b2 ca 2c .!.a;.....V.,,
0001b0 e7 79 e5 b4 97 96 92 47 62 16 c4 f9 a3 e2 fa 0c .y.....Gb.....
0001c0 31 ab e8 02 20 3e ac 67 58 72 41 42 f7 52 1b 92 1... >.gXrAB.R..
0001d0 86 4c ec e3 a9 06 1a cc 7f 02 b1 56 1a 1b 5b 83 .L.....V...[.
0001e0 7a f1 8b 3f c2 z...?.
    
```

Figure 25 Certificate information from the openssl command

7.8 Enable showsessionparams mode

command> showsessionparams <1: enable, 0: disable>

This command is used to enable the session parameter display mode. When the showsessionparams mode is enabled, the negotiated QUIC transport parameters and session resumption ticket are displayed on the serial console after a successful handshake, as shown in Figure 26. Users can utilize this information to verify the 0-RTT resumption parameters stored for future connections and debug QUIC transport layer configuration.

```

=====
Session Parameters for Server: google.com
InitMaxData: 196608
InitMaxStreamDataBiLocal: 131072
InitMaxStreamDataBiRemote: 131072
InitMaxStreamDataUni: 131072
InitMaxStreamBi: 100
InitMaxStreamUni: 103
PreSharedKey: C2A93976F0D7E7F597E2C93E80D08F263D01AE51B721346CACDE1635E1981B7B
TicketLifetime: 172799 seconds
TicketAgeAdd: 3526891597
TicketLength: 265
TicketValue: 0214E176AEE9AEA1362F233808613B717EF687A5DCA7467A5582441C3B8985DF...
TimeStamp: 131
=====

```

Figure 26 Serial console when the showsessionparams mode is enabled

For testing with general server, QUIC1GC demo will display the received data on the serial console, as shown in Figure 29.

```

>> myGET https://example.com
=====
Start IP initialization process
Handshake done
Loading...done
=====
http3 encoding header content has 158 bytes

Address  0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000 00 00 D9 E0 5F 1D 87 49 7C A5 89 D3 4D 1F 57 A6
00000010 FE 5E 68 4C BD 24 A4 78 0F 88 AD C8 F9 01 20 C8
00000020 F1 65 7D 96 5D 08 C0 ED C0 BA CB 8E BE F8 1E 02
00000030 E1 10 42 CB 5F CF 5A 96 D0 7A BE 94 0B 2A 65 1D
00000040 4A 08 02 6D 41 02 E0 43 70 40 53 16 8D FF 54 83
00000050 08 9B 73 5F 15 8A A4 7E 56 1C C5 81 E7 00 00 7F
00000060 56 96 E4 59 3E 94 03 CA 6A 22 54 10 04 DA 80 7E
00000070 E0 9F B8 DB EA 62 D1 BF 2F 02 ED 69 88 B7 72 D8
00000080 83 1E AF 87 07 90 00 00 00 00 1F 5F 44 90 9D 98
00000090 3F 9B 8D 34 CF F3 F6 A5 23 81 F6 5C 00 3F 00 10
=====
http3 data content has 1256 bytes

<!doctype html>
<html>
<head>
  <title>Example Domain</title>

  <meta charset="utf-8" />
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <style type="text/css">
    body {
      background-color: #f0f0f2;
      margin: 0;
      padding: 0;
      font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
    }
    div {
      width: 600px;
      margin: 5em auto;
      padding: 2em;
      background-color: #fdfdff;
      border-radius: 0.5em;
      box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
      color: #38488f;
      text-decoration: none;
    }
    @media (max-width: 700px) {
      div {
        margin: 0 auto;
        width: auto;
      }
    }
  </style>
</head>

<body>
<div>
  <h1>Example Domain</h1>
  <p>This domain is for use in illustrative examples in documents. You may use this
  domain in literature without prior coordination or asking for permission.</p>
  <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>

=====
Handshake Time: 249 ms.
Firstbyte Time: 491 ms.
=====
Total transfer size = 1256 Byte(s)
Download Speed 44.460 Mbps

>>

```

Figure 29 Serial console when downloading webpage

7.10 POST method

command> myECHO https://<ip:port>/echo <length>

This command simulates the POST method of HTTP/3 to upload data to the aioquic server. The URL structure for running the POST method is similar to the GET method with the exception that the transfer length is replaced by a string of “echo”. Users can instead specify the length of the uploading data, which is an 8-bit counting pattern, in another option. After the upload is completed, the aioquic server will return what it has received. This allows the users to verify the received data from the aioquic server. By enabling the verification feature, it monitors whether the received data matches the expected pattern or not, and after verifying it, the data content, transfer length, and transfer speeds are displayed, as shown in Figure 30 and Figure 31.

```
>> myECHO https://192.168.7.25:4433/echo 123
=====
Start IP initialization process
Handshake done
Uploading...done
Downloading...done
=====
http3 encoding header content has 43 bytes

Address  0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000 00 00 D9 5F 4D 89 19 8F DA D3 11 80 AE 25 C1 56
00000010 96 C3 61 BE 94 10 14 C2 58 D4 10 04 E2 80 7A E0
00000020 9F B8 16 94 C5 A3 7F 54 82 08 99 30 2E 30 20 28

=====
Echoed data has been verified, and
Showing Rx data content with the first data offset at 0x000055B0

Address  0 1 2 3 4 5 6 7 8 9 a b c d e f
000055B0 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
000055C0 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
000055D0 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
000055E0 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
000055F0 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
00005600 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
00005610 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
00005620 70 71 72 73 74 75 76 77 78 79 7A 2B 2C 2D 2E 2F

=====
Handshake Time: 95 ms.
Firstbyte Time: 100 ms.
=====
Total transfer size = 123 Byte(s)
Upload Speed 197.709 kbps
Total transfer size = 123 Byte(s)
Download Speed 1.159 Mbps
```

Figure 30 Serial console when uploading small data

```
>> myECHO https://192.168.7.25:4433/echo 50000000
This connections is try 0-RTT
=====
Start IP initialization process
Handshake done
Uploading...done
Downloading...done
=====
http3 encoding header content has 47 bytes

Address  0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000 00 00 D9 5F 4D 89 19 8F DA D3 11 80 AE 25 C1 56
00000010 96 C3 61 BE 94 10 14 C2 58 D4 10 04 E2 80 7A E3
00000020 2E DC 6D E5 31 68 DF 54 86 6C 00 00 00 00 7F 28

=====
Echoed data has been verified, and
Rx data content is too large so only the transfer speed is displayed
=====
Handshake Time: 41 ms.
Firstbyte Time: 6744 ms.
=====
Total transfer size = 50000000 Byte(s)
Upload Speed 178.714 Mbps
Total transfer size = 50000000 Byte(s)
Download Speed 90.458 Mbps
```

Figure 31 Serial console when uploading large data

7.11 PERF method

command> myPERF <ip:port> <uploadlength> <downloadlength>

This command is designed to run with the “secnetperf” example of an MsQuic server. There are three parameters required to run this command – the first option is the server’s IP address and server’s port number separated by a colon, the second is the length of the upload data, and the last one is the length of the download data.

Specifically, this application protocol is designed to have the client transferring the upload data firstly, after which the client receives the download data from the server. Similar to the POST method, the verification feature is used to monitor the received data, and the results, such as the download content, the transfer length, and the transfer speeds, are presented, as shown in Figure 32. It is also important to note that the performance of this operation depends on the network system and the resources available on the test machine.

```
>> myPERF 192.168.7.25:4432 0 123
=====
Start IP initialization process
Handshake done
Running...
TX: Start
TX: Done
RX: Start
RX: Done
Completed
=====
Pattern data has been verified, and
Showing Rx data content with the first data offset at 0x00049C3

Address  0 1 2 3 4 5 6 7 8 9 a b c d e f
000049C0 78 79 7A 00 00 00 00 00 00 00 08 09 0A 0B 0C
000049D0 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C
000049E0 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C
000049F0 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C
00004A00 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B 4C
00004A10 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C
00004A20 5D 5E 5F 60 61 62 63 64 65 66 67 68 69 6A 6B 6C
00004A30 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A DD DE
=====
Total transfer size = 8 Byte(s)
Upload Speed 67.940 kbps
Total transfer size = 123 Byte(s)
Download Speed 2.337 Mbps
```

Figure 32 Serial console when downloading small data

Figure 32 is not a good example to represent the transfer performance because the operation time is too small for accurate calculation. Figure 33, however, can be used to show the transfer speeds for both upload and download because the transfer size settings are large enough.

```
>> myPERF 192.168.7.25:4432 600000000 550000000
=====
Start IP initialization process
Handshake done
Running...
TX: Start
TX: 100 MB
TX: 200 MB
TX: 300 MB
TX: 400 MB
TX: 500 MB
TX: Done
RX: Start
RX: 100 MB
RX: 200 MB
RX: 300 MB
RX: 400 MB
RX: 500 MB
RX: Done
Completed
=====
Pattern data has been verified, and
Data content is too large so only the transfer speed is displayed
=====
Total transfer size = 600000000 Byte(s)
Upload Speed 912.017 Mbps
Total transfer size = 550000000 Byte(s)
Download Speed 926.145 Mbps
```

Figure 33 Serial console when downloading large data

8 Graphic user interface

To visualize the use of the QUIC1GC-IP, QUIC1GCDemo-GUI.exe and config.json are provided. The GUI allows users to easily interact with the QUIC1GC-IP through the provided buttons and view transfer results based on issued commands and received data in the right panel.

In addition, users can interact with the QUIC1GC-IP via a command-line interface, similar to the Nios Command Shell, in the left panel as shown in Figure 34.

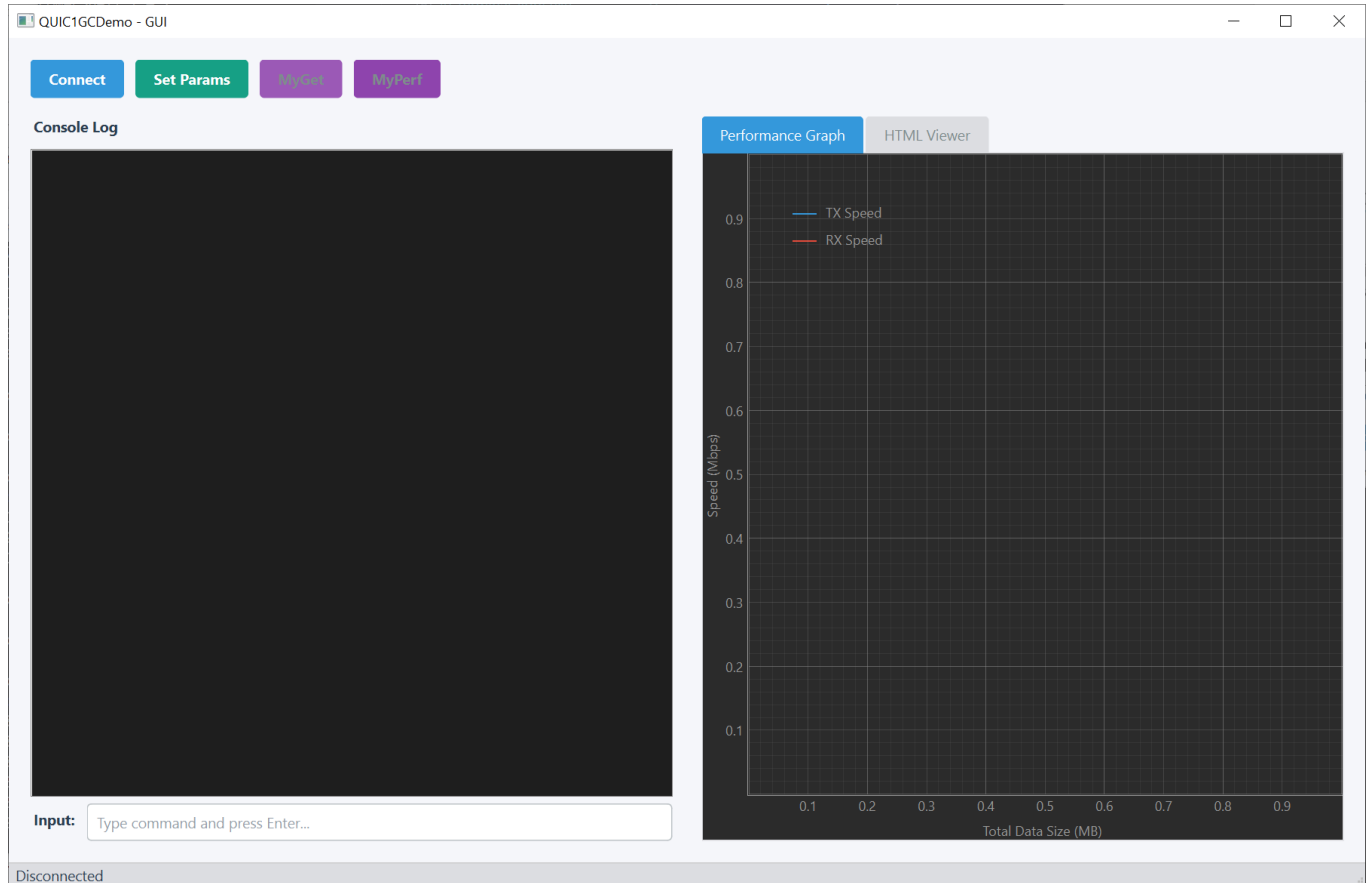


Figure 34 QUIC1GCDemo-GUI window

Note: To support custom application development, the Python source code for the GUI (QUIC1GCDemo-GUI.py) is provided as a reference example. Users may use this code to understand how to interface with the QUIC1GCdemo or modify it to create their own custom application interface. Please note that Design Gateway does not take responsibility for any issues arising from user-modified code.

8.1 Connect with FPGA device

To establish a connection with the FPGA:

- 1) Ensure the cable_setting in config.json is correctly configured (e.g., "USB-BlasterII [USB-1]" for JTAG-UART or "COMx" for Serial).
- 2) Click the "Connect" button in the top control bar.
- 3) The status bar at the bottom will display "Connected (JTAG-UART)" or "Connected (Serial COMx)" upon success.
- 4) Once connected, the button label changes to "Disconnect", and the "MyGet" and "MyPerf" buttons become enabled.

As shown in Figure 35, in this example, "cable_setting" in config.json is set to USB-BlasterII [USB-1]. The application connects to the FPGA device via a JTAG-UART connection and displays "Connected (JTAG-UART)" in the status bar at the bottom of the window.

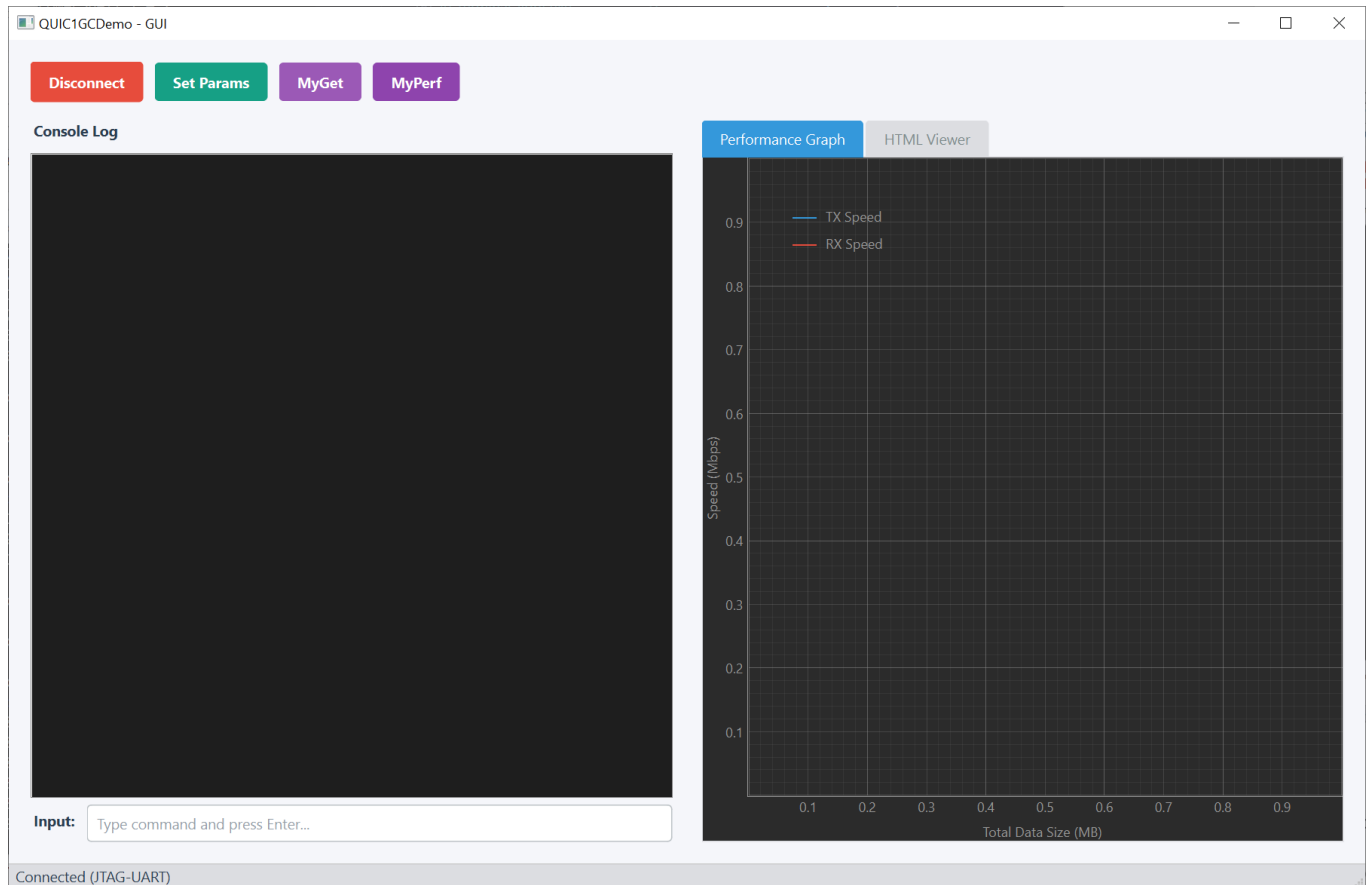


Figure 35 QUIC1GCDemo-GUI device connection

8.2 Set demo parameters

Before running tests, the network and test parameters must be initialized:

- 1) Review and edit the network settings and test parameters in config.json.
- 2) Click the “Set Params” button.
- 3) The software reloads the configuration from the file and automatically sends the following initialization commands to the FPGA: setip, setgatewayip, setport, setmac, and loadnetworkparameters.

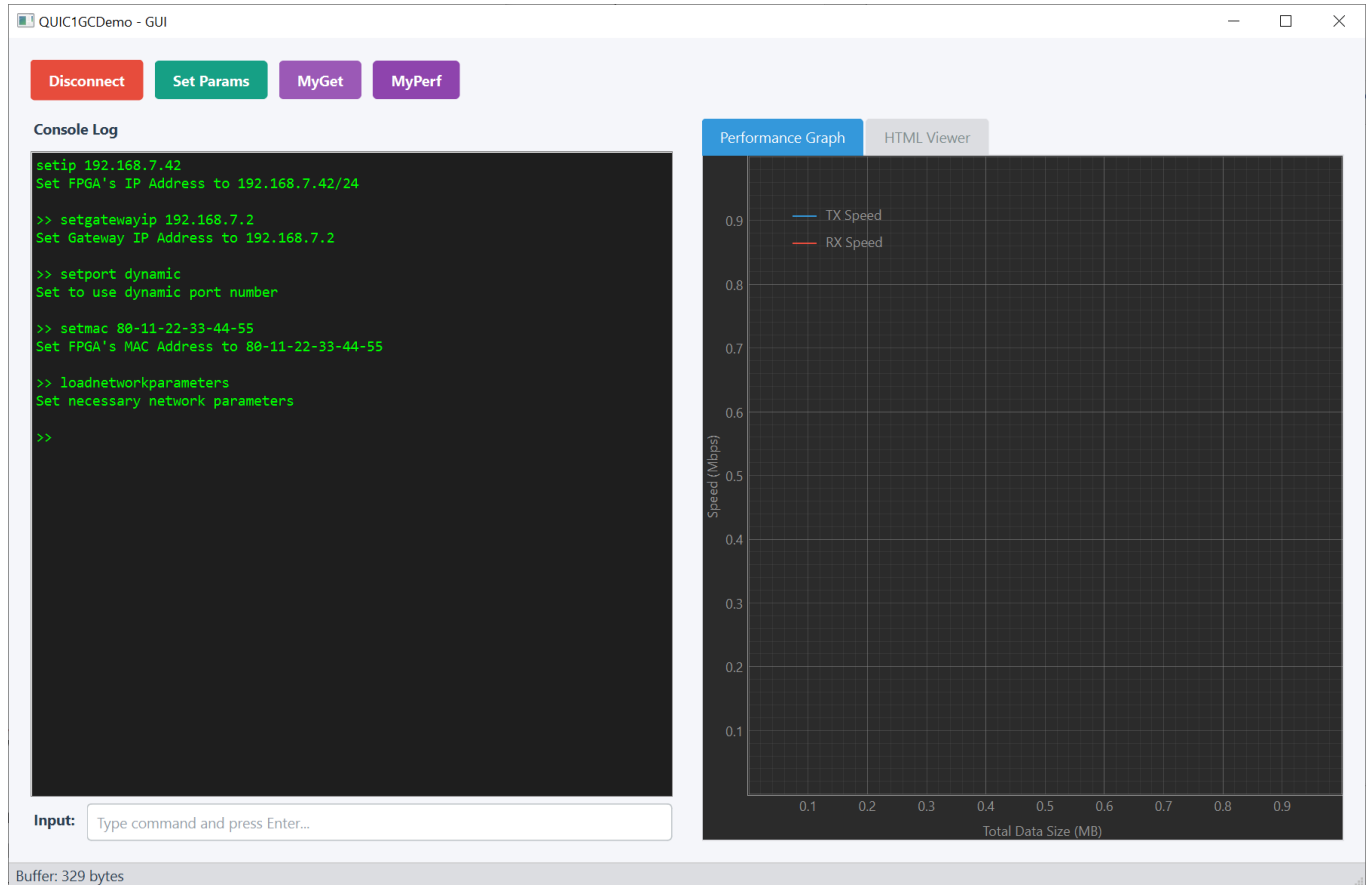


Figure 36 QUIC1GCDemo-GUI parameters setting

8.3 Test with myget command

The “MyGet” command is used to test HTTP/3 request/response functionality:

- 1) Click the “MyGet” button.
- 2) The software sends a request to the URL specified by “myget_url” in the configuration file loaded when the Set Params button is clicked.
- 3) Upon detecting the command, the GUI automatically switches the right panel to the HTML Viewer tab.
- 4) The received HTML content is extracted from the terminal stream and rendered in the HTML viewer. If the received data does not conform to HTML format, “Cannot show html page” is displayed in the right panel instead.

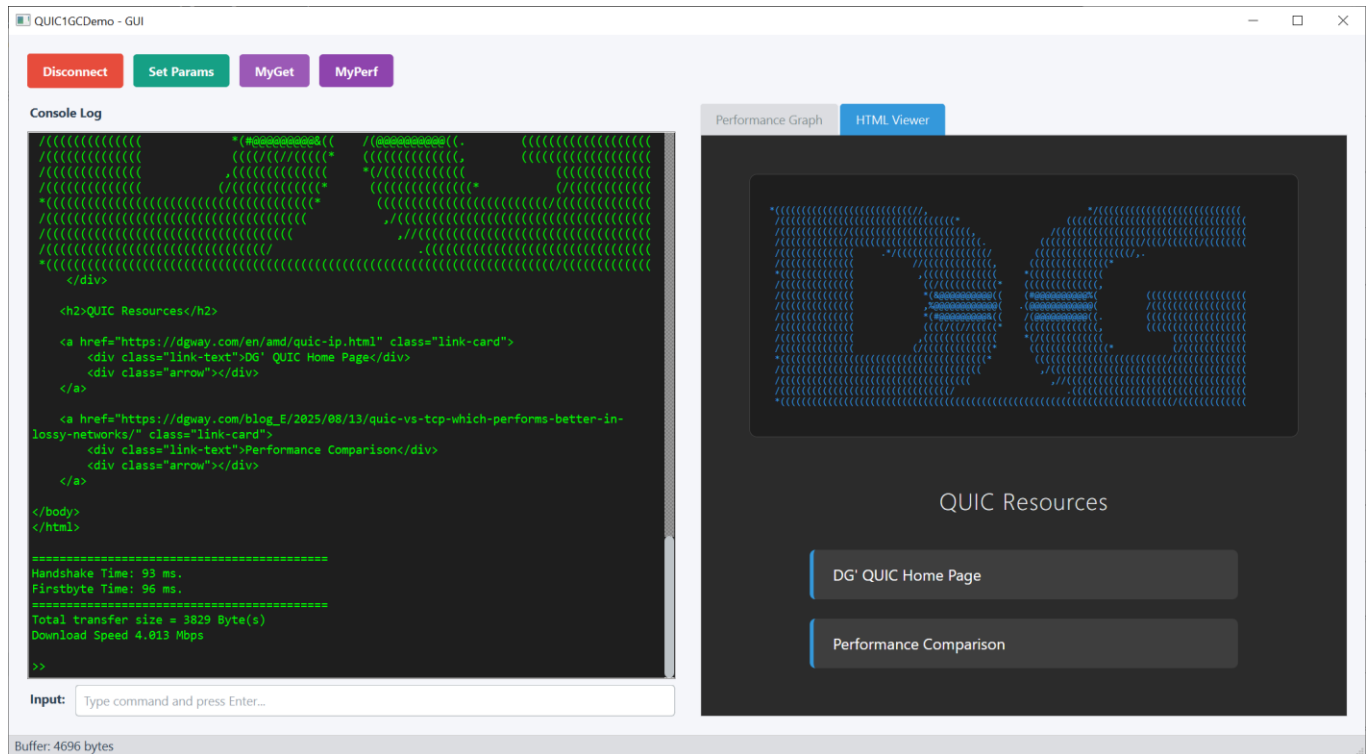


Figure 37 QUIC1GCDemo-GUI MyGet command

Note: To browse a URL that is not specified in config.json, the user can manually enter the myget command with the desired URL in the input box, similar to entering commands in the Nios Command Shell.

8.4 Test with myperf command

The “MyPerf” command is used to measure network throughput:

- 1) Click the “MyPerf” button.
- 2) The software sends a performance test command using the TX and RX data lengths specified in config.json, which is loaded when the “Set Params” button is clicked.
- 3) The GUI automatically switches the right panel to the Performance Graph tab.
- 4) As the test runs, the software parses the "TX: X MB" and "RX: X MB" progress messages in real-time.
- 5) The graph plots the average speed (Mbps) against the total data size (MB), providing a live visualization of the transfer speed.

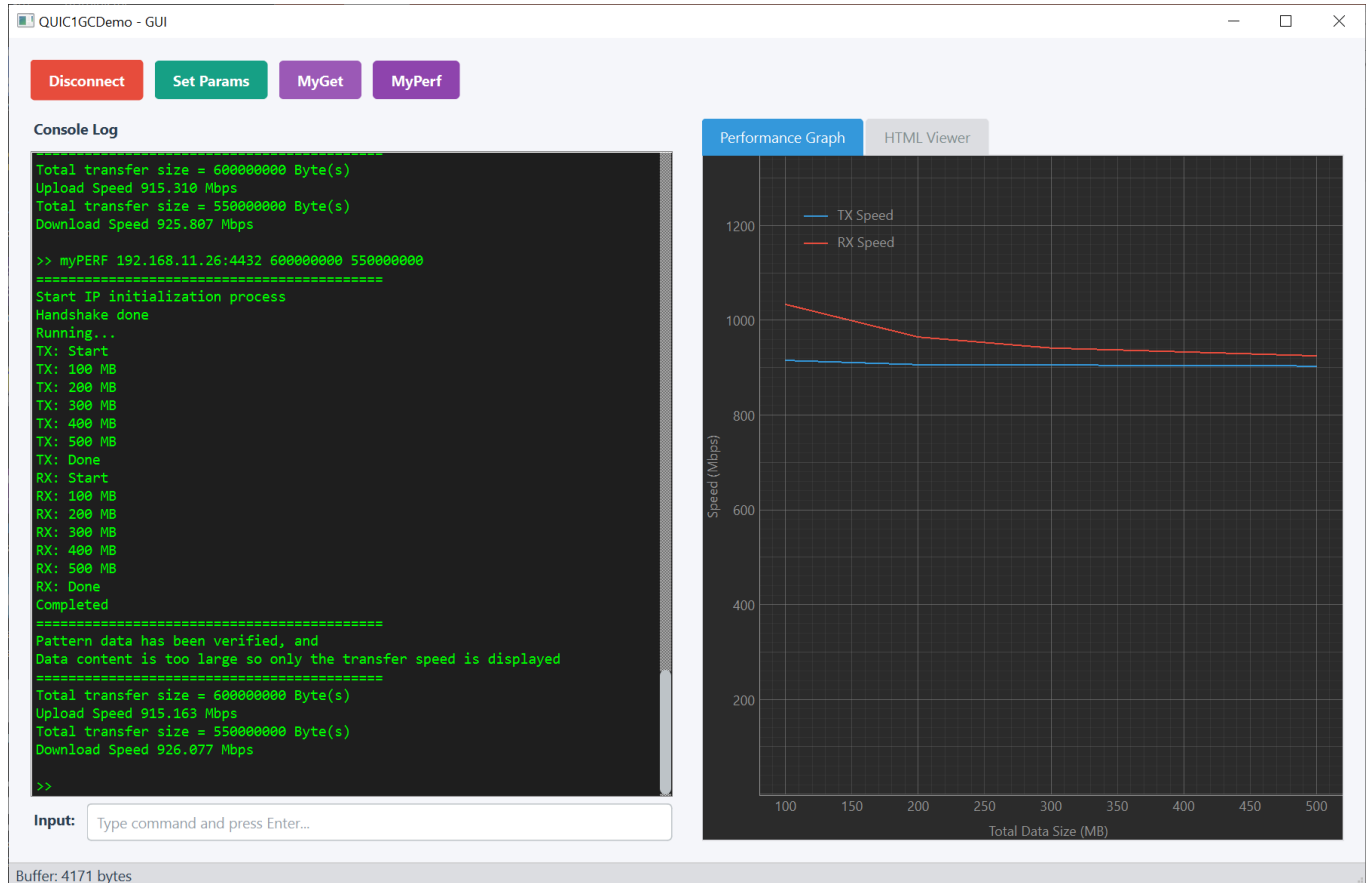


Figure 38 QUIC1GCDemo-GUI Myperf command

Note: The user can manually enter the myperf command with a different server or data size in the input box, similar to using the Nios Command Shell.

9 Revision history

Revision	Date (D-M-Y)	Description
1.01	24-Feb-26	Add Sulfur Agilex 5 E-Series board.
1.00	16-Feb-26	Initial version release