# Readme

Unzip the project, project-develop.zip.

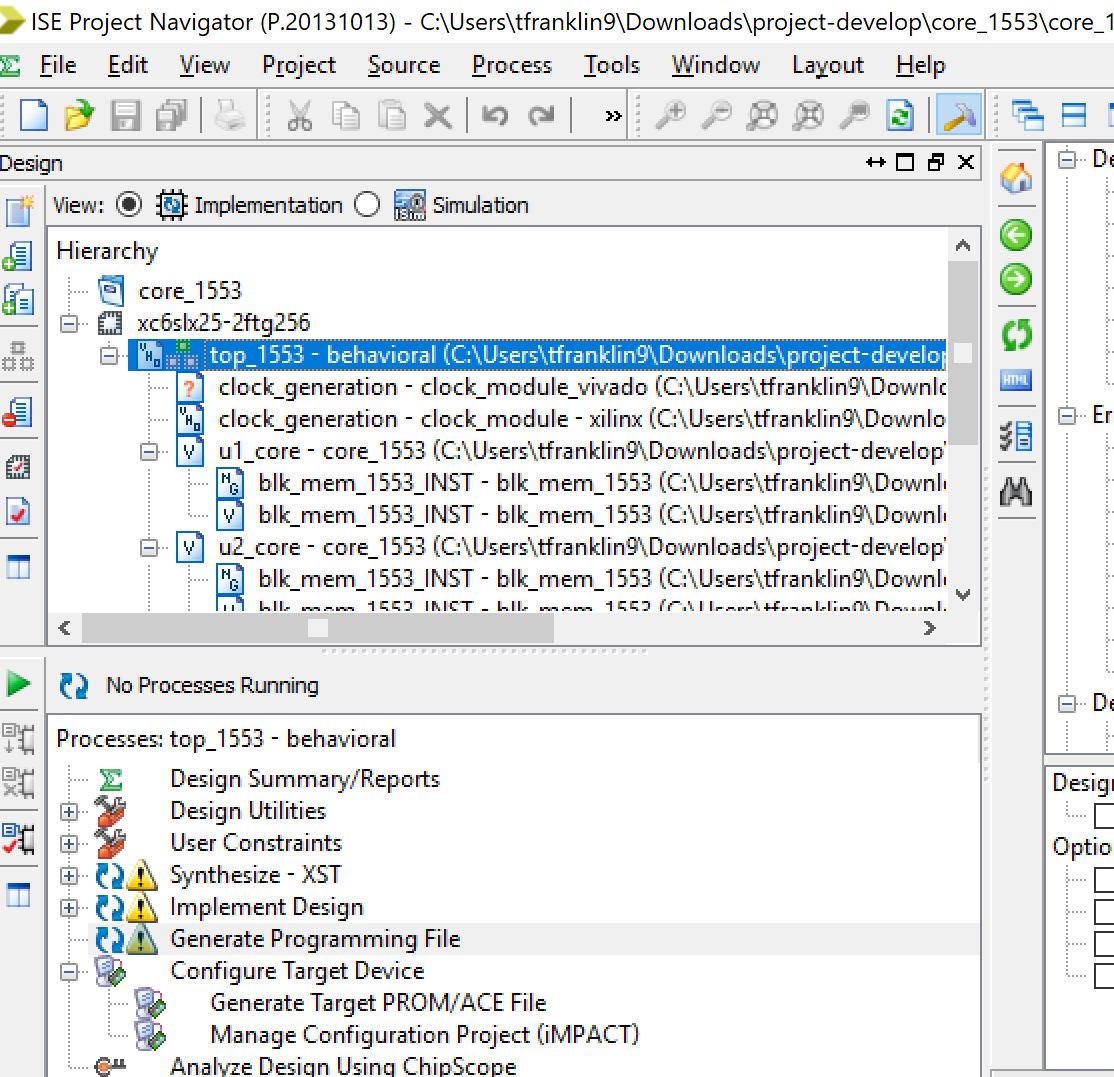
Unzip the ztex Hardware files, ztex-171119.zip.

## Building the project.

After downloading Xilinx ISE 14.7 and obtaining a WebPACK license generating a bit file should be relatively straight forward.

1. Open ISE. Click on File 🡪 Open Project and navigate to the ./project-develop/core\_1553 directory, core\_1553.xise.
2. Select Spartan6 LX25 or LX16 by right clicking on ‘top\_1553’ in the ISE Hierarchy pane and clicking *Design Properties.* Change the Device value to the required Part. The ztex boards are populated with either the XC6LX25 or XC6LX16. Note, the archived project targets the Spartan6 LX25.
3. In the Processing window pane, right click on 'Generate Programming File' and select *ReRun All*.

The following figure is a capture of the ISE window.



After synthesis and P&R the bitfile, top\_1553.bit, is located in the project’s core\_1553 directory.

## Programming the FPGA

### USB Driver

Make sure the laptop in use has a USB driver installed. If problems arise while trying to install the USB driver normally, a USB driver installation tool, Zadig.exe, is included in the ztex zip file. The Quick Start Guide for Zadig is as follows:

1. Plug-in the device. If Windows asks for a driver, cancel this.
2. Start Zadig
3. Choose the device and click install driver.

The ztex board should show up in the drop down list and has the Vendor ID, 0x221A.

1. Top of Form
2. Bottom of Form

### Loading the Bitfile

To load the bitfile into non-volatile memory open a command prompt and navigate to the unzipped ztex hardware files.

1. From the /ztex/java/DeviceServer directory run the command,

Windows: *java -cp DeviceServer.jar DeviceServer -ha <ip address> -sp 1 -v -l2 logfile.txt*

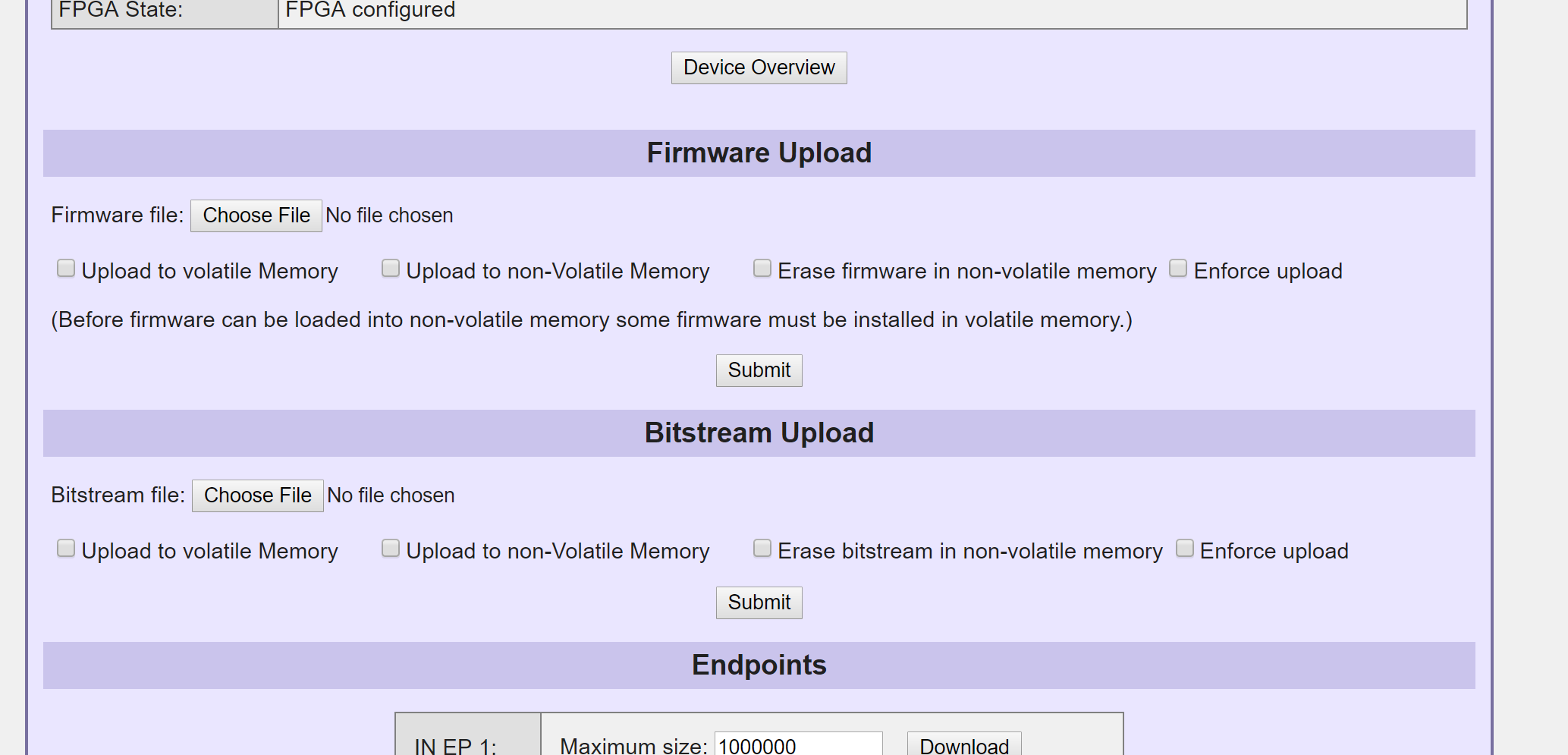
Cygwin: *./DeviceServer -ha <ip address> -sp -1 -v -l2 logfile.txt.*

1. Once the device server is running you should be able to access the ztex board through a web browser at http://<ip address>:9080

Once connected via USB you can erase the bit file in memory and replace it with a new bitfile. *Note that the Firmware Upload section looks very similar to the Bitstream upload section. Make sure you are erasing and loading the FPGA Bitstream and not the ztex board Firmware.*

1. In the Bitstream Upload section, check “*Erase bitstream in non-volatile memory*” and click the *submit* button. The device server messages section at the bottom should tell you that the erase has been completed.
2. Disconnect the usb and reconnect it. The light on the ztex board should remain on. The FPGA has not been programmed as there is no bit file in memory.
3. In the bitstream upload section, deselect the “*Erase bitstream in non-volatile memory”*, select *“Upload to non-volatile memory”*, click *browse*, select the bit file from the project’s core\_1553 directory, and click *submit*. The Device Server Messages section should report that the bitstream has been loaded to non-volatile memory.
4. Power cycle again. This time the light on the ztex board should go off indicating that the FPGA has been programmed.

The following figure shows the ztex web interface.



FPGA Pinout.

# Transmit and receive

NET "rxa\_p\_BC" LOC = "T12" | IOSTANDARD = LVCMOS33; # IO A3

NET "rxa\_n\_BC" LOC = "T14" | IOSTANDARD = LVCMOS33; # IO A4

NET "txa\_p\_BC" LOC = "T15" | IOSTANDARD = LVCMOS33 | DRIVE = 12; # IO A5

NET "txa\_n\_BC" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 12; # IO A6

# Capture signal outputs

NET "csw" LOC = "R12" | IOSTANDARD = LVCMOS33 | DRIVE = 12 ; # B3

NET "dw" LOC = "T13" | IOSTANDARD = LVCMOS33 | DRIVE = 12 ; # B4

NET "enc\_data\_en" LOC = "R14" | IOSTANDARD = LVCMOS33 | DRIVE = 12 ; # B5

NET "enc\_data" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 12 ; # B6

# Trigger signals

NET "triggerA" LOC = "P15" | IOSTANDARD = LVCMOS33 ; # B7 / P15~IO\_L48P\_HDC\_M1DQ8\_1

NET "triggerB" LOC = "N14" | IOSTANDARD = LVCMOS33 ; # B8 / N14~IO\_L45P\_A1\_M1LDQS\_1

NET "triggerC" LOC = "M15" | IOSTANDARD = LVCMOS33 ; # B9 / M15~IO\_L46P\_FCS\_B\_M1DQ2\_1

#inputs switch8 is the bypass signal. It should be pulled high. If you want to take the FPGA out of bypass mode this line needs to be tied to GND.

#For triggering leave this line alone.

NET "switch8" LOC = "L14" | IOSTANDARD = LVCMOS33 | PULLUP ; # B11 / L14~IO\_L47P\_FWE\_B\_M1DQ0\_1

## Triggering

Setting the triggers is done by setting the generics in the top level vhdl file located at ‘/project-develop/source/top\_1553.vhd’. The trigger generics are shown below.

*constant WAIT\_MAX : positive := 32 ;*

*constant DELAY\_MAX : positive := 1000 ;*

*constant HLD\_MAXA : positive := 1088 ;*

*constant DLY\_MAXA : positive := 2 ;*

*constant HLD\_MAXB : positive := 544 ;*

*constant DLY\_MAXB : positive := 2 ;*

*constant HLD\_MAXC : positive := 272 ;*

*constant DLY\_MAXC : positive := 2 ;*

*constant NO\_HLDA : std\_logic := '0';*

*constant NO\_DLYA : std\_logic := '0';*

*constant NO\_HLDB : std\_logic := '0';*

*constant NO\_DLYB : std\_logic := '0';*

*constant NO\_HLDC : std\_logic := '0';*

*constant NO\_DLYC : std\_logic := '0';*

WAIT\_MAX is the duration, in microseconds, of inactivity on the rx bus before searching for and triggering on an incoming command/status sync word. DELAY\_MAX is the duration, in microseconds, that the user expects the complete transaction to take. After DELAY\_MAX, the user can assert triggers of different delays and lengths. Currently the duration of the bus transactions and gaps have to be obtained by the User prior to build time. The FPGA can be used to view the bus transaction timing before hand by examining csw, dw, enc\_data, and enc\_data\_en on a scope or logic analyzer.

There are three triggers, triggerA, triggerB, and triggerC. Each trigger has a maximum delay, DLY\_MAX, and a maximum hold time, HLD\_MAX. DLY\_MAX is the wait between the DELAY\_MAX of the transaction and the assertion of the trigger. HLD\_MAX is the duration of the asserted trigger.

A timing diagram of the trigger delays is shown below.

*DELAY\_MAX*

*WAIT\_MAX*

*HLD\_MAXA*

*DLY\_MAXA*

*WAIT\_MAX*

*Bus*

*triggerA*

*transactions*

Use the NO\_HLD generic if there is no need for the trigger. Use the NO\_DLY generic to start the trigger immediately after DELAY\_MAX.

The current code does not support ‘0’ values and all delay and hold units are in microseconds.