TCL Flow

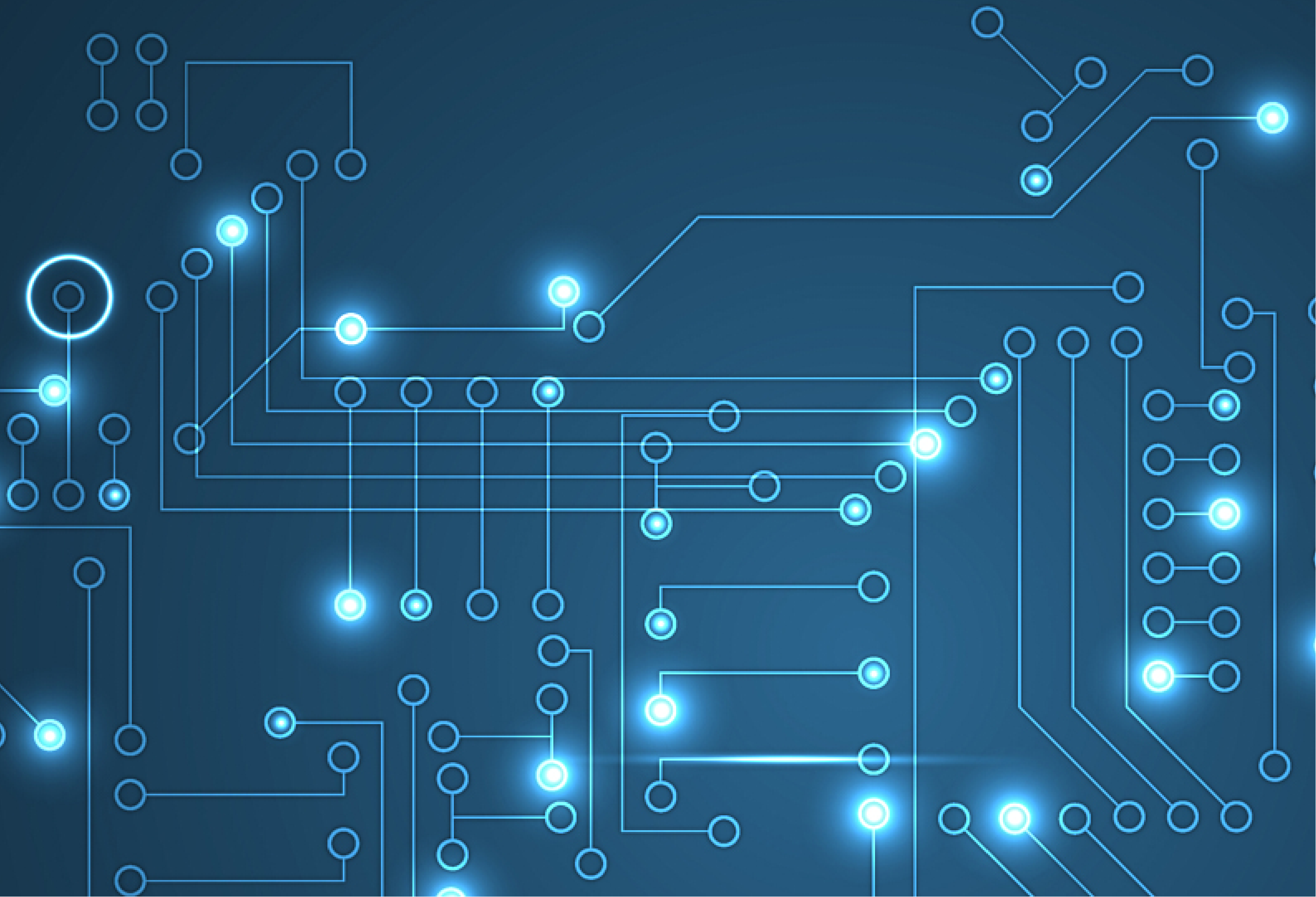
A companion guide to the text book:

A Practical Introduction to the Xilinx Zynq-7000 Adaptive SoC

Author: Derek Murray

Version: 1.0

Date: 26/9/21



Revision History

Version	Date	Comment
1.0	26/9/21	First version

Table 1. Revision History

1 Introduction

This document describes a TCL flow for building the hardware project associated with the textbook "A Practical introduction to the Xilinx Zynq-7000 Adaptive SoC- Bare-Metal Fundamentals". The following boards are currently supported:

- Digilent Zybo-Z7-20
- Digilent Zybo-Z7-10
- Digilent/Avnet ZedBoard

Note that the Digilent board files must be installed in the correct Vivado directory, as follows (assuming that C is the install directory):

- C:\Xilinx\Vivado\<ver>\data\boards\board_files\zybo-z7-20
- C:\Xilinx\Vivado\<ver>\data\boards\board_files\zybo-z7-10
- C:\Xilinx\Vivado\<ver>\data\boards\board_files\zedboard

(In the case of the ZedBoard, another set of board files called "*zed*" might be found in the *board_files* directory; these are not compatible with the TCL script, and the "*zedboard*" files supplied by Digilent must be used.)

The TCL script files carry out the following steps:

- The block diagram is created.
- OOC synthesis is performed.
- A top-level wrapper file is created.
- Constraint files are loaded for the design.
- The design is synthesised and implemented, and the bit-file is created.
- Vivado is launched.

At this point, the user can export the design and run the associated software IDE (SDK for Vivado <= 2019.1 or Vitis IDE (for Vivado >= 2019.2). The TCL script has been tested and found to work on **Vivado 2017.4, 2018.3, 2019.1, and 2020.2**. It may also work on other versions (although an issue has been found in 2021.1, see next).

In Vivado 2021.1, the steps described above are carried out and Vivado will launch, but the source files (e.g. block diagram and top-level wrapper file) cannot be opened until Vivado is restarted. However, it is still possible to export the design and build the software projects in Vitis IDE 2021.1 without restarting Vivado. If a fix is found for 2021.1, it will be implemented (or another script will be made available).

Once the software IDE is opened, the reader can continue with the design flow described in the related step-by-step companion documents. Two such documents are available, one for SDK and one for Vitis:

- Vivado and SDK Development Flow (*Vivado-SDK Flow.pdf*)

- Vivado and Vitis Development Flow (*Vivado-Vitis IDE Flow.pdf*)

In both cases, the user should go to Section 2.5, "*Hardware Hand-off*", to continue with the design flow.

1.1 Pre-requisites

The reader should have the following ready before proceeding:

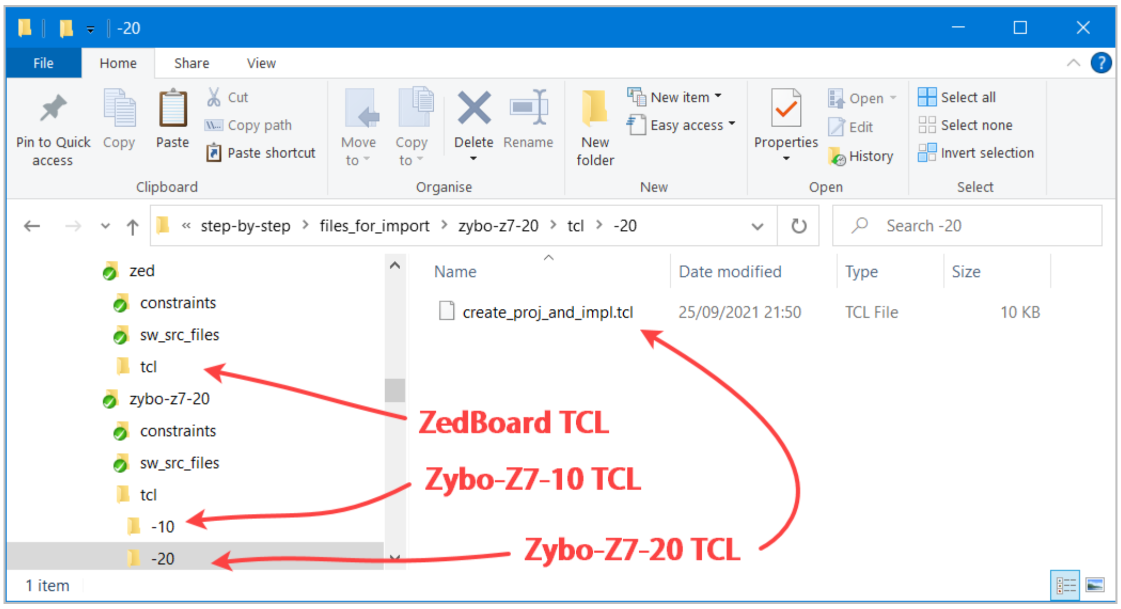
- Board files at correct location in the Vivado installation, as mentioned above.
- Step-by-step document(s) for HW hand-off and SW design, also mentioned above.
- FPGA constraint files for the required board, available on this GitHub repo (e.g. <repo>\book1-zynq-intro\step-by-step\files_for_import\<board>\constraints\main_constraints.xdc)
- Software files for the required board, also available on this GitHub repo (e.g. <repo>\book1-zynq-intro\step-by-step\files_for_import\<board>\sw_src_files\sw_projX\.*)

Note that the constraint and software files for the Zybo-Z7-20 are compatible with the Zybo-Z7-10.

The TCL files are named "*create_proj_and_impl.tcl*", and they are, of course, also available on GitHub (see also Figure 1):

- <repo>\book1-zynq-intro\step-by-step\files_for_import\<board>\tcl\*\.*

In this case, different files are available for the Zybo-Z7-20 and Zybo-Z7-10 (i.e. the TCL file for the Zybo-Z7-20 cannot be used for the Zybo-Z7-10). The TCL file for the Zybo-Z7-20 is found in the "*-20*" directory, and the file for the Zybo-Z7-10 is found in the "*-10*" directory.

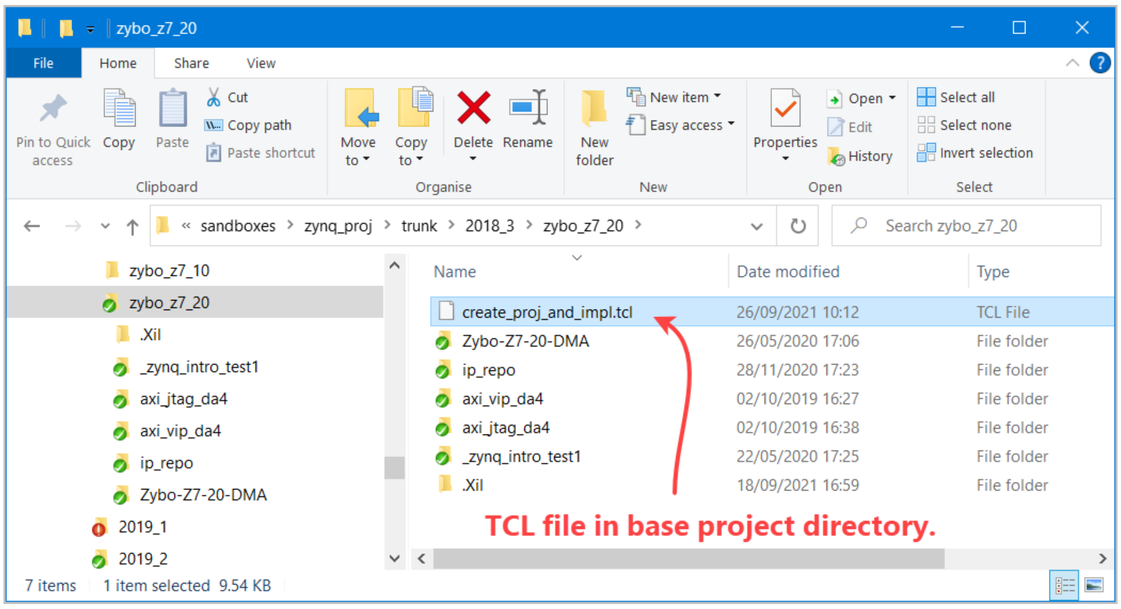


Name	Date modified	Type	Size
create_proj_and_impl.tcl	25/09/2021 21:50	TCL File	10 KB

Figure 1. The TCL files can be found in *<repo>\book1-zynq-intro\step-by-step\files_for_import\<board>\tcl\*

2 Procedure

To start, the user should copy the relevant TCL file to the directory where they want to build the project (Figure 2). (Note that when the TCL file is executed, a sub-directory containing the project will be created.)

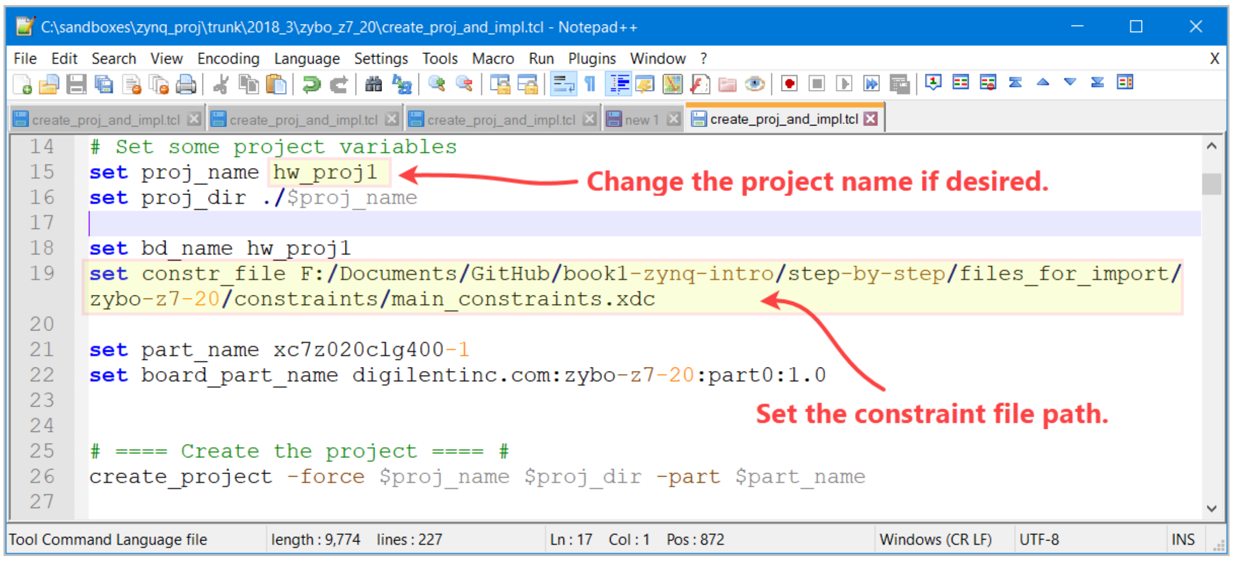


Name	Date modified	Type
create_proj_and_impl.tcl	26/09/2021 10:12	TCL File
Zybo-Z7-20-DMA	26/05/2020 17:06	File folder
ip_repo	28/11/2020 17:23	File folder
axi_vip_da4	02/10/2019 16:27	File folder
axi_jtag_da4	02/10/2019 16:38	File folder
_zynq_intro_test1	22/05/2020 17:25	File folder
.Xil	18/09/2021 16:59	File folder

Figure 2. Copy the TCL file to the project directory.

Next, open the TCL file and make two changes (Figure 3) (although the first change is optional):

1. The project name can be set to the desired name ("hw_proj1" is used in the example). (Note: DO NOT change the *bd_name* from "hw_proj1"!!!)
2. The path to the constraint file should be set to wherever the user has cloned or downloaded the file, as shown in the figure.



C:\sandboxes\zynq_proj\trunk\2018_3\zybo_z7_20\create_proj_and_impl.tcl - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
create_proj_and_impl.tcl
create_proj_and_impl.tcl
create_proj_and_impl.tcl
new 1
create_proj_and_impl.tcl
Change the project name if desired.
Set the constraint file path.
Tool Command Language file
length : 9,774 lines : 227
Ln : 17 Col : 1 Pos : 872
Windows (CR LF)
UTF-8
INS

```
# Set some project variables
set proj_name hw_proj1
set proj_dir ./$proj_name

set bd_name hw_proj1
set constr_file F:/Documents/GitHub/book1-zynq-intro/step-by-step/files_for_import/
zybo-z7-20/constraints/main_constraints.xdc

set part_name xc7z020clg400-1
set board_part_name digilentinc.com:zybo-z7-20:part0:1.0


# ==== Create the project ==== #
create_project -force $proj_name $proj_dir -part $part_name

```

Figure 3. Change the proj_name if desired, and set the constraint file path.

Open a Command Prompt utility in Windows (Figure 4):

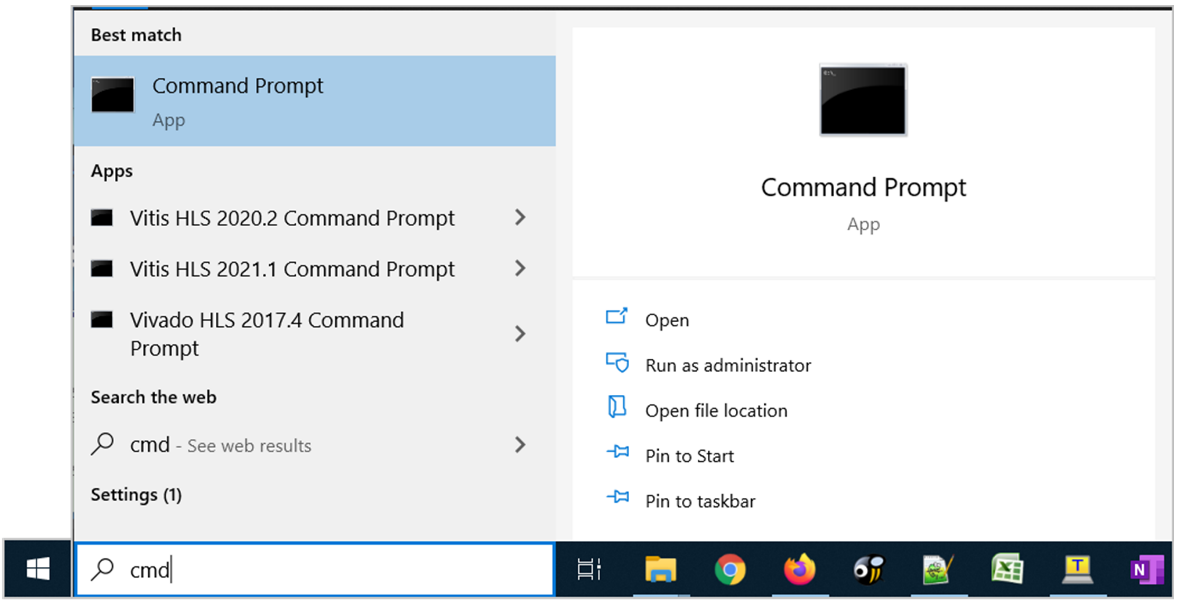
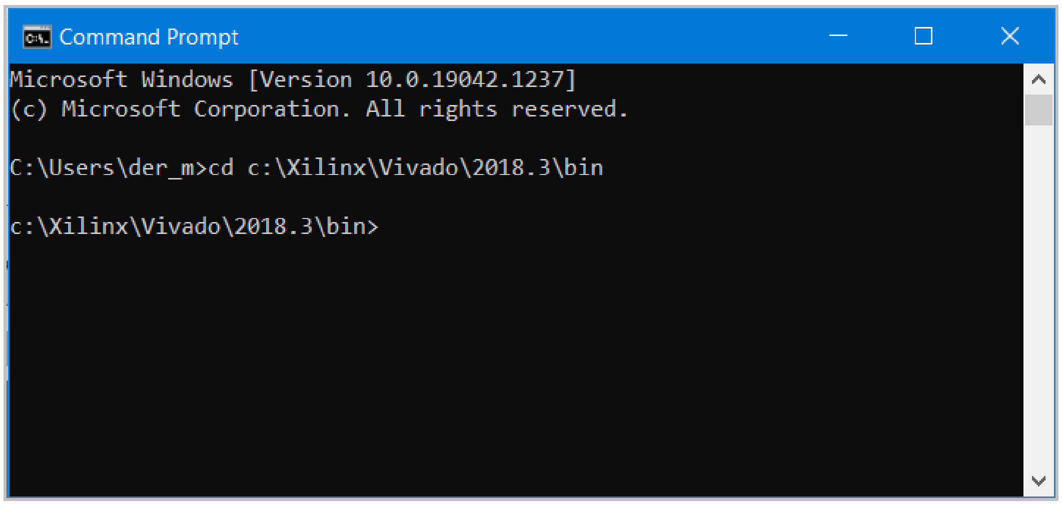


Figure 4. Open a Command Prompt utility in Windows.

Change the directory to the “bin” folder of the Vivado tool version being used. For example, version 2018.3 is used in Figure 5. Assuming that Vivado is installed on the C drive, the path is:

- C:\Xilinx\Vivado\<ver>\bin

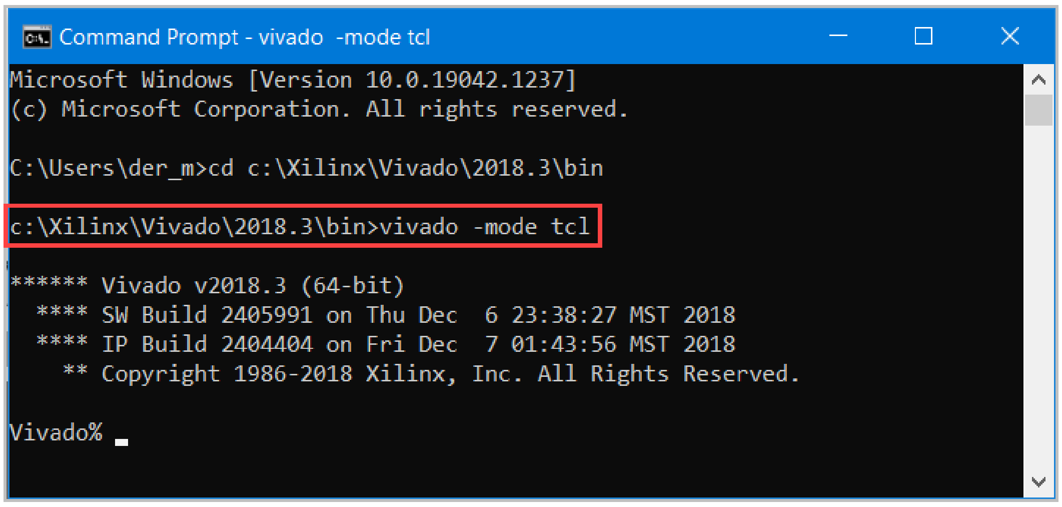


```
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\der_m>cd c:\Xilinx\Vivado\2018.3\bin

c:\Xilinx\Vivado\2018.3\bin>
```

Figure 5. Change the directory to the bin folder of the Vivado tool version.

Next, run Vivado in TCL mode by typing the following (see also Figure 6):

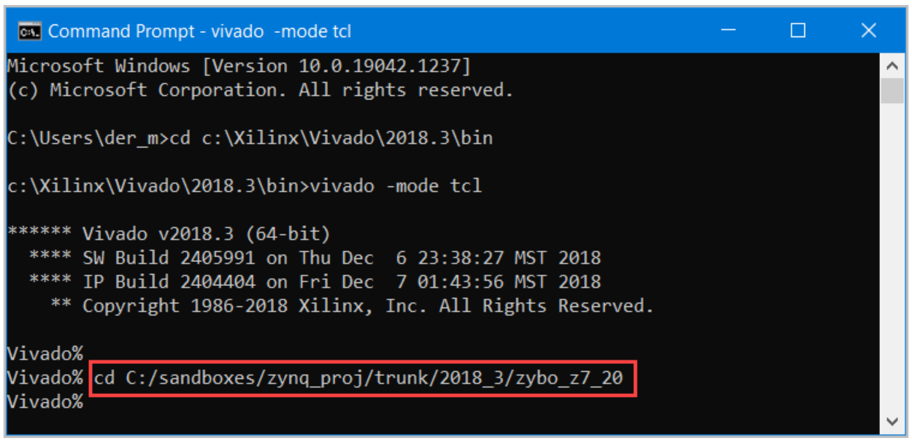
- vivado -mode tcl



```
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\der_m>cd c:\Xilinx\Vivado\2018.3\bin

c:\Xilinx\Vivado\2018.3\bin>vivado -mode tcl

****** Vivado v2018.3 (64-bit)
  **** SW Build 2405991 on Thu Dec  6 23:38:27 MST 2018
  **** IP Build 2404404 on Fri Dec  7 01:43:56 MST 2018
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

Vivado% _
```

Figure 6. Run Vivado in TCL mode.

The command prompt marker will change to **Vivado%**. Change directory again, this time to the location where the project will be built (Figure 7). To elaborate, this is the location where the TCL file was saved in Figure 2. Note that forward slashes must be used when entering paths in the Vivado command line.

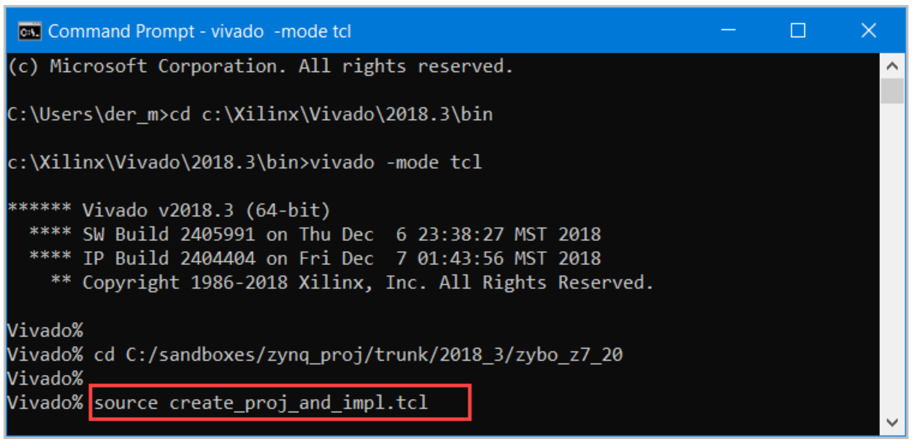


```
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\der_m>cd c:\Xilinx\Vivado\2018.3\bin

c:\Xilinx\Vivado\2018.3\bin>vivado -mode tcl

****** Vivado v2018.3 (64-bit)
  **** SW Build 2405991 on Thu Dec  6 23:38:27 MST 2018
  **** IP Build 2404404 on Fri Dec  7 01:43:56 MST 2018
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

Vivado%
Vivado% cd C:/sandboxes/zynq_proj/trunk/2018_3/zybo_z7_20
Vivado%
```

Figure 7. At the Vivado% prompt, change to the directory where the project will be built.

Now, the TCL script can be executed. Type the following at the command line (Figure 8):

- source create_proj_and_impl.tcl



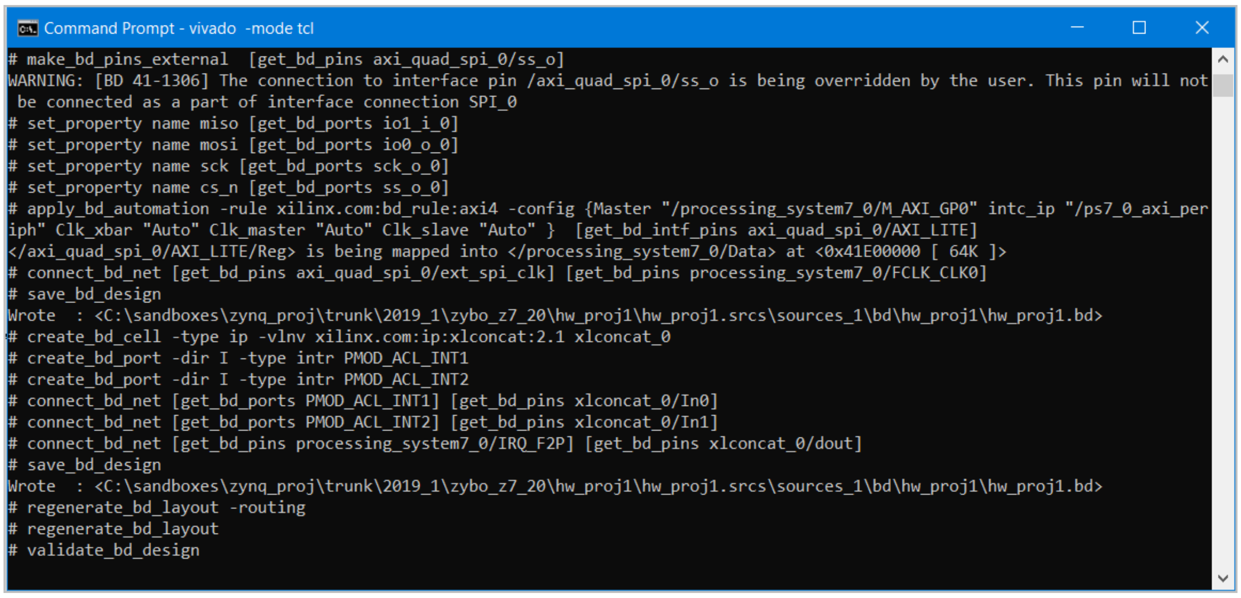
Command Prompt - vivado -mode tcl

```
(c) Microsoft Corporation. All rights reserved.

C:\Users\der_m>cd c:\Xilinx\Vivado\2018.3\bin

c:\Xilinx\Vivado\2018.3\bin>vivado -mode tcl

****** Vivado v2018.3 (64-bit)
  **** SW Build 2405991 on Thu Dec  6 23:38:27 MST 2018
  **** IP Build 2404404 on Fri Dec  7 01:43:56 MST 2018
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

Vivado%
Vivado% cd C:/sandboxes/zynq_proj/trunk/2018_3/zybo_z7_20
Vivado%
Vivado% source create_proj_and_impl.tcl
```

Figure 8. Execute the TCL file by typing '*source create_proj_and_impl.tcl*'

Assuming that there are no issues, the TCL file should run (Figure 9). The build process takes about 10-15 minutes on a medium-spec PC. The reader should be patient during the build process, as it will often appear as if nothing is happening; if it takes more than 30 minutes, however, then it is likely that a fatal error has occurred. The command window should be inspected to see if any errors are flagged.

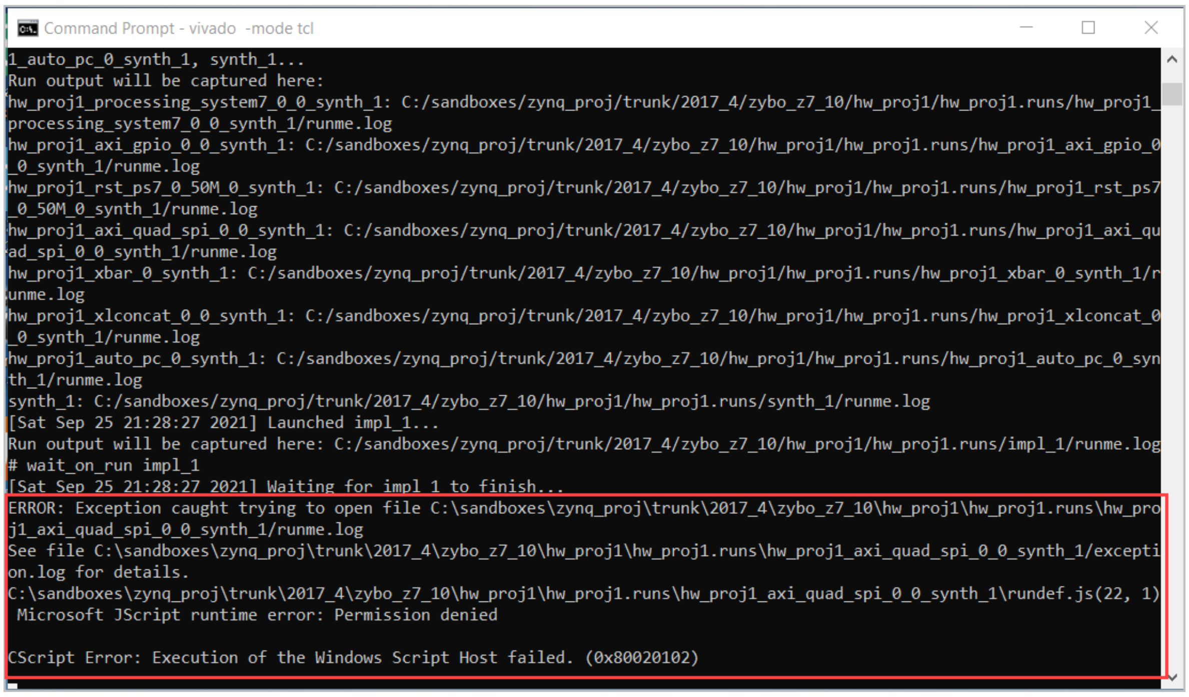
(NOTE also that the issue shown in Figure 10 is not as sinister as it appears- the build process continues in the background and should complete as normal.)



Command Prompt - vivado -mode tcl

```
# make_bd_pins_external  [get_bd_pins axi_quad_spi_0/ss_o]
WARNING: [BD 41-1306] The connection to interface pin /axi_quad_spi_0/ss_o is being overridden by the user. This pin will not
 be connected as a part of interface connection SPI_0
# set_property name miso [get_bd_ports io1_i_0]
# set_property name mosi [get_bd_ports io0_o_0]
# set_property name sck [get_bd_ports sck_o_0]
# set_property name cs_n [get_bd_ports ss_o_0]
# apply_bd_automation -rule xilinx.com:bd_rule:axi4 -config {Master "/processing_system7_0/M_AXI_GP0" intc_ip "/ps7_0_axi_per
iph" Clk_xbar "Auto" Clk_master "Auto" Clk_slave "Auto" }  [get_bd_intf_pins axi_quad_spi_0/AXI_LITE]
</axi_quad_spi_0/AXI_LITE/Reg> is being mapped into </processing_system7_0/Data> at <0x41E00000 [ 64K ]>
# connect_bd_net [get_bd_pins axi_quad_spi_0/ext_spi_clk] [get_bd_pins processing_system7_0/FCLK_CLK0]
# save_bd_design
Wrote  : <C:\sandboxes\zynq_proj\trunk\2019_1\zybo_z7_20\hw_proj1\hw_proj1.srcs\sources_1\bd\hw_proj1\hw_proj1.bd>
# create_bd_cell -type ip -vlnv xilinx.com:ip:xlconcat:2.1 xlconcat_0
# create_bd_port -dir I -type intr PMOD_ACL_INT1
# create_bd_port -dir I -type intr PMOD_ACL_INT2
# connect_bd_net [get_bd_ports PMOD_ACL_INT1] [get_bd_pins xlconcat_0/In0]
# connect_bd_net [get_bd_ports PMOD_ACL_INT2] [get_bd_pins xlconcat_0/In1]
# connect_bd_net [get_bd_pins processing_system7_0/IRQ_F2P] [get_bd_pins xlconcat_0/dout]
# save_bd_design
Wrote  : <C:\sandboxes\zynq_proj\trunk\2019_1\zybo_z7_20\hw_proj1\hw_proj1.srcs\sources_1\bd\hw_proj1\hw_proj1.bd>
# regenerate_bd_layout -routing
# regenerate_bd_layout
# validate_bd_design
```

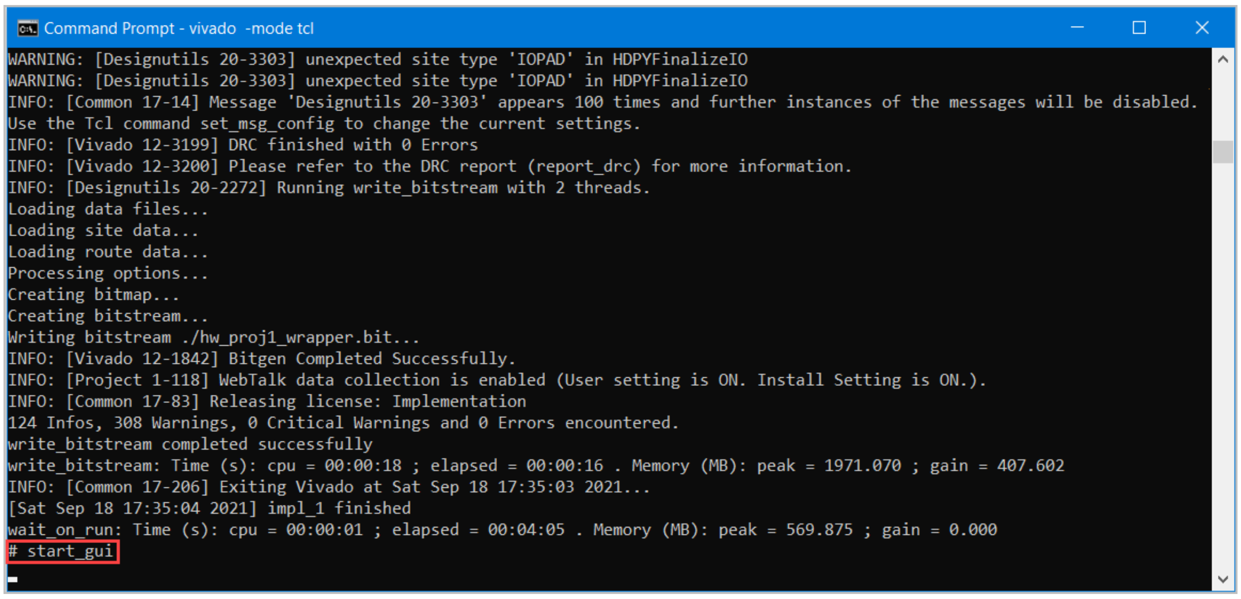
Figure 9. The build process proceeds if no issues are encountered...



```
1_auto_pc_0_synth_1, synth_1...
Run output will be captured here:
hw_proj1_processing_system7_0_0_synth_1: C:/sandboxes/zynq_proj/trunk/2017_4/zybo_z7_10/hw_proj1/hw_proj1.runs/hw_proj1_
processing_system7_0_0_synth_1/runme.log
hw_proj1_axi_gpio_0_0_synth_1: C:/sandboxes/zynq_proj/trunk/2017_4/zybo_z7_10/hw_proj1/hw_proj1.runs/hw_proj1_axi_gpio_0
_0_synth_1/runme.log
hw_proj1_rst_ps7_0_50M_0_synth_1: C:/sandboxes/zynq_proj/trunk/2017_4/zybo_z7_10/hw_proj1/hw_proj1.runs/hw_proj1_rst_ps7
_0_50M_0_synth_1/runme.log
hw_proj1_axi_quad_spi_0_0_synth_1: C:/sandboxes/zynq_proj/trunk/2017_4/zybo_z7_10/hw_proj1/hw_proj1.runs/hw_proj1_axi_qu
ad_spi_0_0_synth_1/runme.log
hw_proj1_xbar_0_synth_1: C:/sandboxes/zynq_proj/trunk/2017_4/zybo_z7_10/hw_proj1/hw_proj1.runs/hw_proj1_xbar_0_synth_1/r
unme.log
hw_proj1_xlconcat_0_0_synth_1: C:/sandboxes/zynq_proj/trunk/2017_4/zybo_z7_10/hw_proj1/hw_proj1.runs/hw_proj1_xlconcat_0
_0_synth_1/runme.log
hw_proj1_auto_pc_0_synth_1: C:/sandboxes/zynq_proj/trunk/2017_4/zybo_z7_10/hw_proj1/hw_proj1.runs/hw_proj1_auto_pc_0_syn
th_1/runme.log
synth_1: C:/sandboxes/zynq_proj/trunk/2017_4/zybo_z7_10/hw_proj1/hw_proj1.runs/synth_1/runme.log
[Sat Sep 25 21:28:27 2021] Launched impl_1...
Run output will be captured here: C:/sandboxes/zynq_proj/trunk/2017_4/zybo_z7_10/hw_proj1/hw_proj1.runs/impl_1/runme.log
# wait_on_run impl_1
[Sat Sep 25 21:28:27 2021] Waiting for impl_1 to finish...
ERROR: Exception caught trying to open file C:\sandboxes\zynq_proj\trunk\2017_4\zybo_z7_10\hw_proj1\hw_proj1.runs\hw_pro
j1_axi_quad_spi_0_0_synth_1/runme.log
See file C:\sandboxes\zynq_proj\trunk\2017_4\zybo_z7_10\hw_proj1\hw_proj1.runs\hw_proj1_axi_quad_spi_0_0_synth_1/excepti
on.log for details.
C:\sandboxes\zynq_proj\trunk\2017_4\zybo_z7_10\hw_proj1\hw_proj1.runs\hw_proj1_axi_quad_spi_0_0_synth_1\rundef.js(22, 1)
 Microsoft JScript runtime error: Permission denied

CScript Error: Execution of the Windows Script Host failed. (0x80020102)
```

Figure 10. This error is not as fatal as it seems; the project should continue to build in the background and will eventually complete as expected.

Finally, the build process should complete, and the Vivado GUI will automatically start (Figure 11):



```
WARNING: [Designutils 20-3303] unexpected site type 'IOPAD' in HDPYFinalizeIO
WARNING: [Designutils 20-3303] unexpected site type 'IOPAD' in HDPYFinalizeIO
INFO: [Common 17-14] Message 'Designutils 20-3303' appears 100 times and further instances of the messages will be disabled.
Use the Tcl command set_msg_config to change the current settings.
INFO: [Vivado 12-3199] DRC finished with 0 Errors
INFO: [Vivado 12-3200] Please refer to the DRC report (report_drc) for more information.
INFO: [Designutils 20-2272] Running write_bitstream with 2 threads.
Loading data files...
Loading site data...
Loading route data...
Processing options...
Creating bitmap...
Creating bitstream...
Writing bitstream ./hw_proj1_wrapper.bit...
INFO: [Vivado 12-1842] Bitgen Completed Successfully.
INFO: [Project 1-118] WebTalk data collection is enabled (User setting is ON. Install Setting is ON.).
INFO: [Common 17-83] Releasing license: Implementation
124 Infos, 308 Warnings, 0 Critical Warnings and 0 Errors encountered.
write_bitstream completed successfully
write_bitstream: Time (s): cpu = 00:00:18 ; elapsed = 00:00:16 . Memory (MB): peak = 1971.070 ; gain = 407.602
INFO: [Common 17-206] Exiting Vivado at Sat Sep 18 17:35:03 2021...
[Sat Sep 18 17:35:04 2021] impl_1 finished
wait_on_run: Time (s): cpu = 00:00:01 ; elapsed = 00:04:05 . Memory (MB): peak = 569.875 ; gain = 0.000
# start_gui
```

Figure 11. When the build process completes, Vivado should automatically start.

At this point, the user can export the design and start software development. As mentioned earlier, the associated step-by-step documents can be used for this phase, starting at Section 2.5 in each case (Hardware Hand-off). (For example, Figure 12 shows the hardware being exported in Vivado 2018.3).

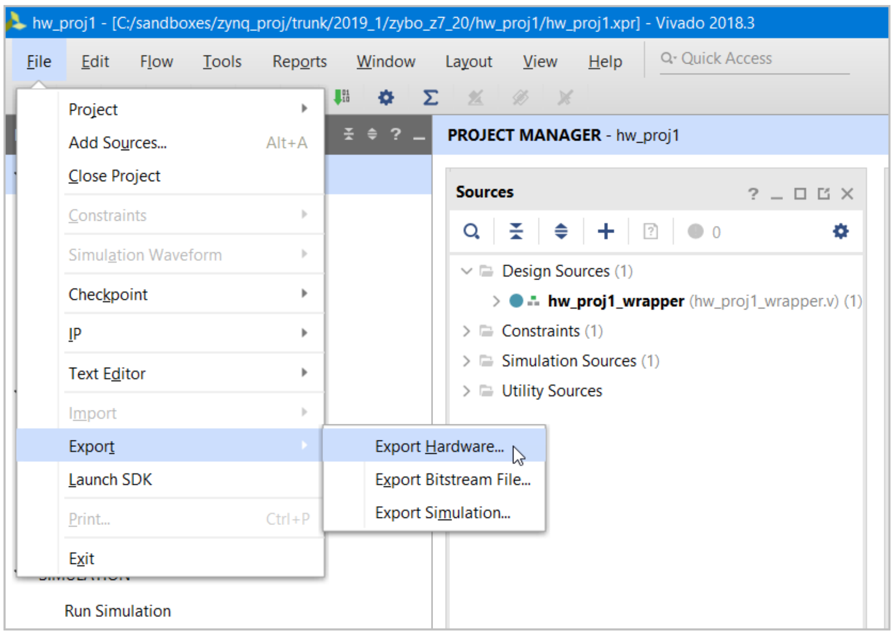


Figure 12. Hardware hand-off in Vivado 2018.3

Also, a reminder of the issue seen in Vivado 2021.1: the GUI must be restarted if the user wants to examine the block diagram or source code.