Non-Volatile FPGA Programming Guide

For: VCU1525, BCU1525, XBB1525, CVP-13, XUPP3R, XUPVV4, AES-KU040
By Zetheron Technology (Updated: Sept 18/2018)

When mining with FPGA's, it is highly desirable to set up your mining card with a non-volatile configuration, so that if you have to unplug your rig and move it, or if there is a power outage or software crash, that the rig can automatically start up on its own without extensive manual user interaction. This guide shows how to configure your FPGA with non-volatile programming.

1. Configure the core voltage in a non-volatile fashion:

This step depends on which FPGA you are using:

VCU1525, BCU1525, XBB1525: Connect your modified DC1613A dongle to the card and run LTPowerPlay software as usual (http://zetheron.com/Downloads/VCU1525_DC1613A_Adapter.pdf). Once you have set the core voltage to your desired level, the click the icon RAM->NVRAM to program the flash memory on the LTC3884 chip. A clock icon will be displayed, and in just a few seconds the configuration will be non-volatile. Now, at power up, your 1525 card will boot up at your desired core voltage. Please note that for effective non-volatile setup, going significantly below 0.70V can prevent the card from programming automatically at start up.

CVP-13, XUPP3R, XUPVV4: Run bwconfig-gui.exe (as usual) and configure the core voltage. ***All options configured with the Bittware GUI are automatically non-volatile***.

AES-KU040: Using the power supply modification as described on the Zetheron website using the 1.5K resistor and rotary potentiometer, the value you have set by manually rotating the potentiometer will automatically be fixed upon boot-up.

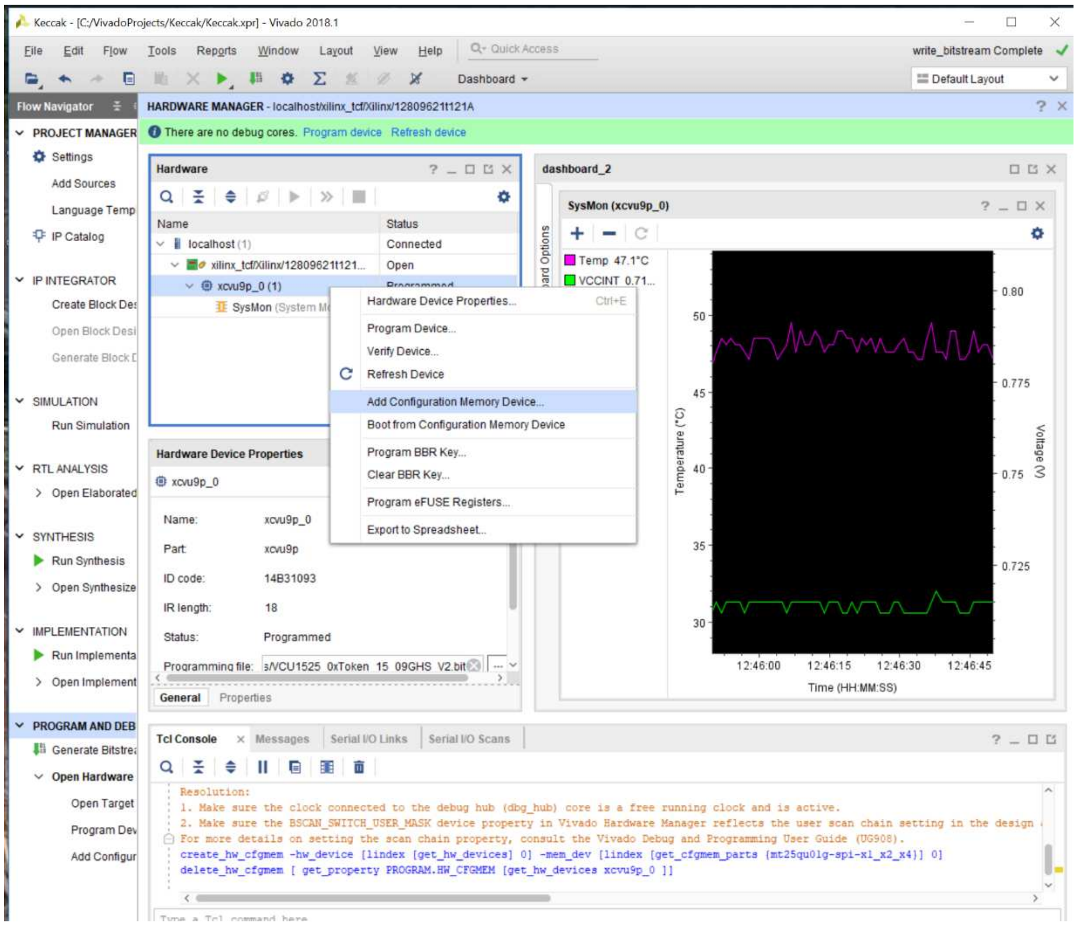
2. Download the appropriate MCS file from the Zetheron Website

For this step, you want to load the bitstream into the FPGA in a non-volatile fashion. High end FPGA's only allow RAM configuration, so to get around this, high end FPGA cards include an external serial flash memory. We load the bitstream into this serial flash memory (external to the FPGA), and once it is there, the board will automatically 'check' the flash memory upon power-up to see if there is a valid bitstream there. If there is, the FPGA will self-program with the contents of the flash memory. Programming the flash memory takes much longer than a direct RAM load of the bitstream. Programming the FPGA directly via Vivado (in a volatile fashion) takes around 40-80 seconds depending on which card you have, but programming the flash memory can take 10-60 minutes depending on your configuration. Once the flash is programmed, the FPGA is very versatile. At power up it will rapidly program itself from the flash in around 15 seconds. Further, if the FPGA 'crashes' while mining due to the core voltage dropping too low, instead of re-programming the FPGA with Vivado, you can either (A) reboot the FPGA via software, or (B) use Vivado to 'boot from configuration memory', which is much faster than a standard programming operation. A RAM bitstream is stored as a .BIT file, but a flash memory configuration bitstream is stored as an .MCS file. The MCS file format is much larger, taking 220MB instead of 77MB for the VU9 FPGA. Go to the Zetheron downloads page:
http://zetheron.com/index.php/downloads/
Download the MCS file for your hardware, for the algorithm that you are interested in.

3. Program the Flash Memory on your FPGA card

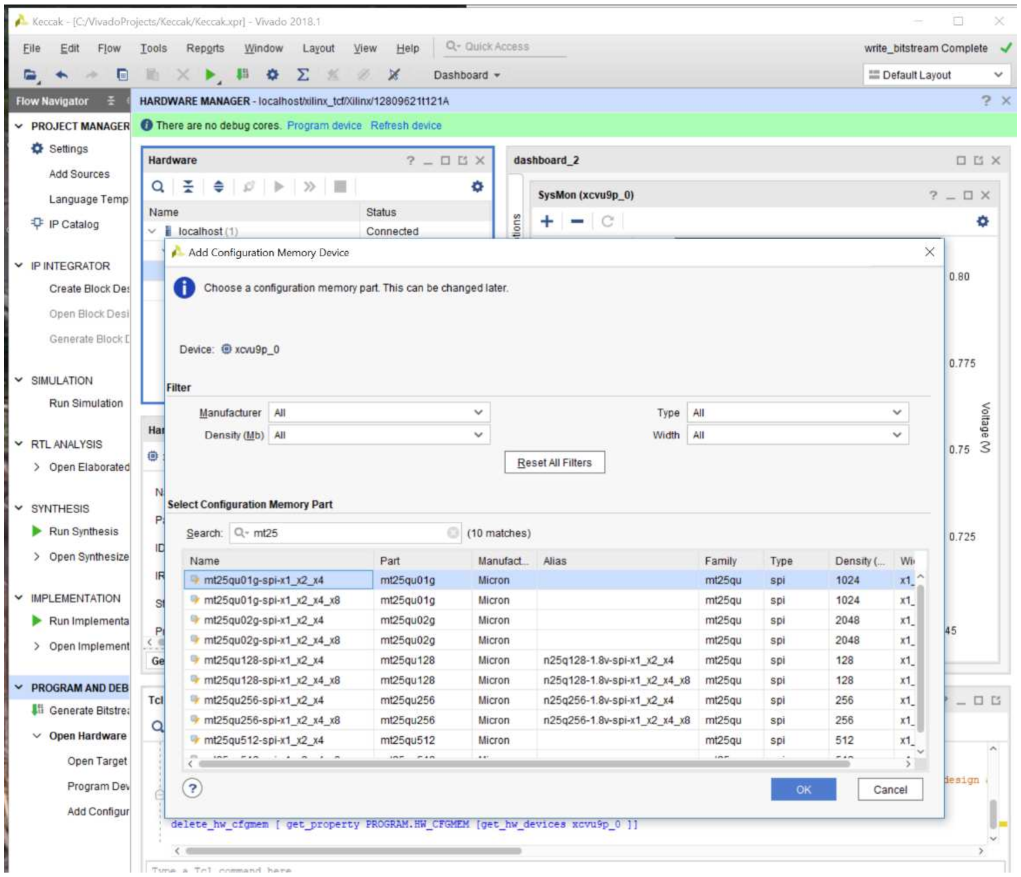
Now that you have set up a non-volatile core voltage, and downloaded the appropriate MCS file, it is time to load the MCS file into the flash memory on your FPGA card. If you have a Bittware card, you can perform this step either with the Bittware GUI or with Vivado Lab Edition. If you have a Xilinx card you must use Vivado Lab edition. This section explains the procedure using Vivado Lab edition which does work on all cards.

Launch Vivado Lab Edition as usual, and launch Hardware Manager--> Open Target--> Auto Connect. Once you see the list of FPGA's in the window, right click on the FPGA identifier (xcvu9p_0 or similar, see below), and select Add Configuration Memory Device:



```
Resolution:
1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active.
2. Make sure the BSCAN_SWITCH_USER_MASK device property in Vivado Hardware Manager reflects the user scan chain setting in the design
For more details on setting the scan chain property, consult the Vivado Debug and Programming User Guide (UG908).
create_hw_cfgmem -hw_device [lindex [get_hw_devices] 0] -mem_dev [lindex [get_cfgmem_parts {mt25qu01g-spi-x1_x2_x4}] 0]
delete_hw_cfgmem [ get_property PROGRAM.HW_CFGMEM [get_hw_devices xcvu9p_0 ]]
```

Next, you must select the exact model of flash memory from a long list.



There are no debug cores. Program device Refresh device

Name	Status
localhost (1)	Connected

Add Configuration Memory Device

Choose a configuration memory part. This can be changed later.

Device: xcvu9p_0

Filter

Manufacturer: All
Type: All
Density (Mb): All
Width: All

Reset All Filters

Select Configuration Memory Part

Search: mt25 (10 matches)

Name	Part	Manufact...	Alias	Family	Type	Density (...	Wi
mt25qu01g-spi-x1_x2_x4	mt25qu01g	Micron		mt25qu	spi	1024	x1_
mt25qu01g-spi-x1_x2_x4_x8	mt25qu01g	Micron		mt25qu	spi	1024	x1_
mt25qu02g-spi-x1_x2_x4	mt25qu02g	Micron		mt25qu	spi	2048	x1_
mt25qu02g-spi-x1_x2_x4_x8	mt25qu02g	Micron		mt25qu	spi	2048	x1_
mt25qu128-spi-x1_x2_x4	mt25qu128	Micron	n25q128-1.8v-spi-x1_x2_x4	mt25qu	spi	128	x1_
mt25qu128-spi-x1_x2_x4_x8	mt25qu128	Micron	n25q128-1.8v-spi-x1_x2_x4_x8	mt25qu	spi	128	x1_
mt25qu256-spi-x1_x2_x4	mt25qu256	Micron	n25q256-1.8v-spi-x1_x2_x4	mt25qu	spi	256	x1_
mt25qu256-spi-x1_x2_x4_x8	mt25qu256	Micron	n25q256-1.8v-spi-x1_x2_x4_x8	mt25qu	spi	256	x1_
mt25qu512-spi-x1_x2_x4	mt25qu512	Micron		mt25qu	spi	512	x1_

```
delete_hw_cfgmem [ get_property PROGRAM.HW_CFGMEM [get_hw_devices xcvu9p_0 ]]
```

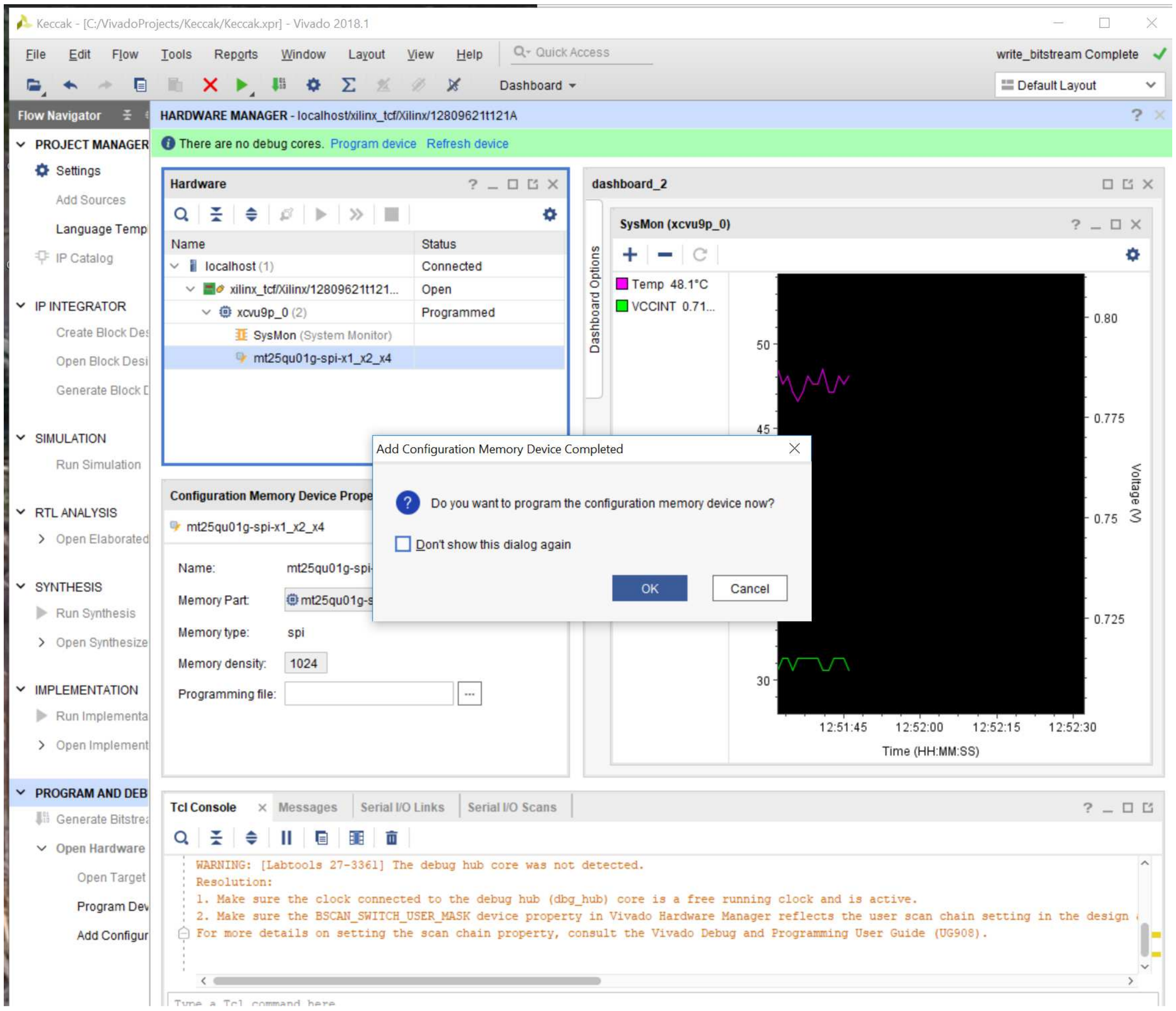
The memory you choose depends on which FPGA card you have:

VCU1525, BCU1525, XBB1525: Choose ***mt25qu01g-spi-x1_x2_x4***

XUPP3R, XUPVV4, CVP-13: Choose ***mt25qu02g-spi-x1_x2_x4***
[NOTE: On the Bittware boards you can also perform the entire flash memory programming step without using Vivado at all, by using the bwconfig-gui.exe utility]

AES-KU040: Your board will have either:
Micron ***N25Q256A***, or
Spansion ***S25FL256SAGMFIR0***

Once you have selected the correct memory device, click OK, and the following dialog will appear asking if you want to program the memory now:



Name	Status
localhost (1)	Connected
xilinx_tcf/Xilinx/12809621t121...	Open
xcvu9p_0 (2)	Programmed
SysMon (System Monitor)	
mt25qu01g-spi-x1_x2_x4	

Configuration Memory Device Prope

mt25qu01g-spi-x1_x2_x4

Name:	mt25qu01g-spi-
Memory Part:	mt25qu01g-s
Memory type:	spi
Memory density:	1024
Programming file:	

Add Configuration Memory Device Completed

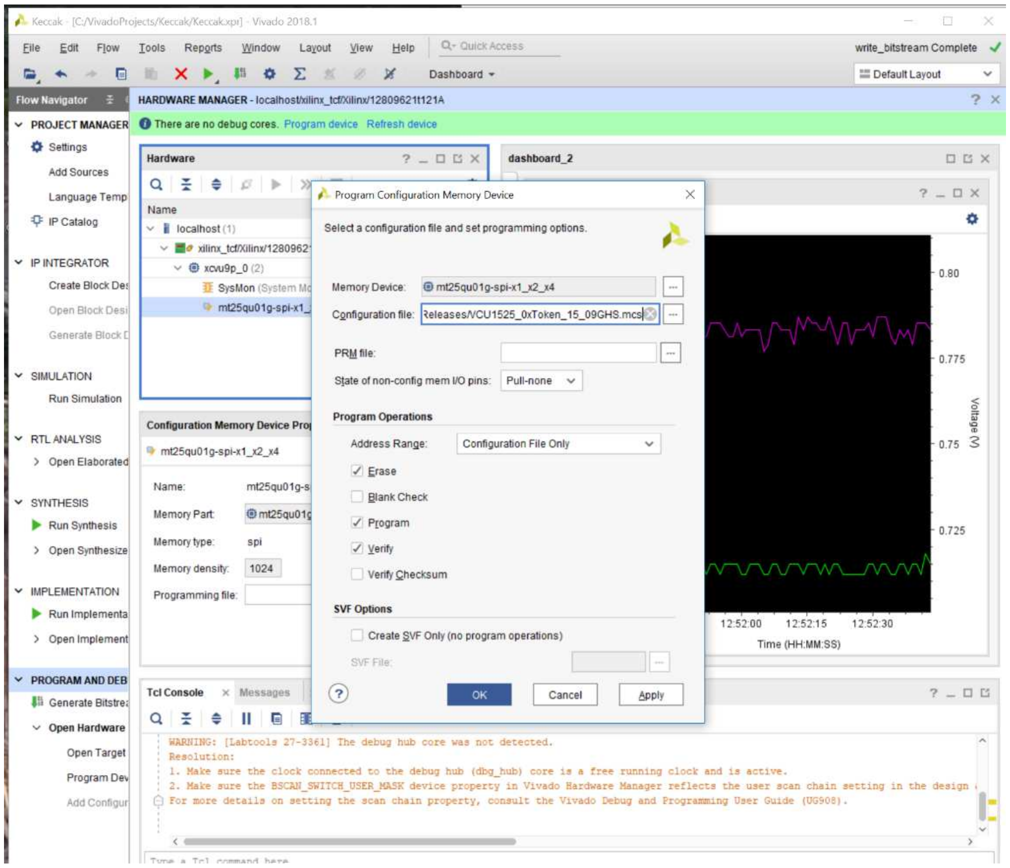
Do you want to program the configuration memory device now?

☐ Don't show this dialog again

OK Cancel

```
WARNING: [Labtools 27-3361] The debug hub core was not detected.
Resolution:
1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active.
2. Make sure the BSCAN_SWITCH_USER_MASK device property in Vivado Hardware Manager reflects the user scan chain setting in the design
For more details on setting the scan chain property, consult the Vivado Debug and Programming User Guide (UG908).
```

Click 'OK' to program the memory. A new dialog will appear with lots of options:



HARDWARE MANAGER - localhost/xilinx_tcf/Xilinx/12809621t121A

There are no debug cores. Program device Refresh device

Program Configuration Memory Device

Select a configuration file and set programming options.

Memory Device: mt25qu01g-spi-x1_x2_x4

Configuration file: Releases/VCU1525_0xToken_15_09GHS.mcs

PRM file:

State of non-config mem I/O pins: Pull-none

Program Operations

Address Range: Configuration File Only

- [✓] Erase
- [ ] Blank Check
- [✓] Program
- [✓] Verify
- [ ] Verify Checksum

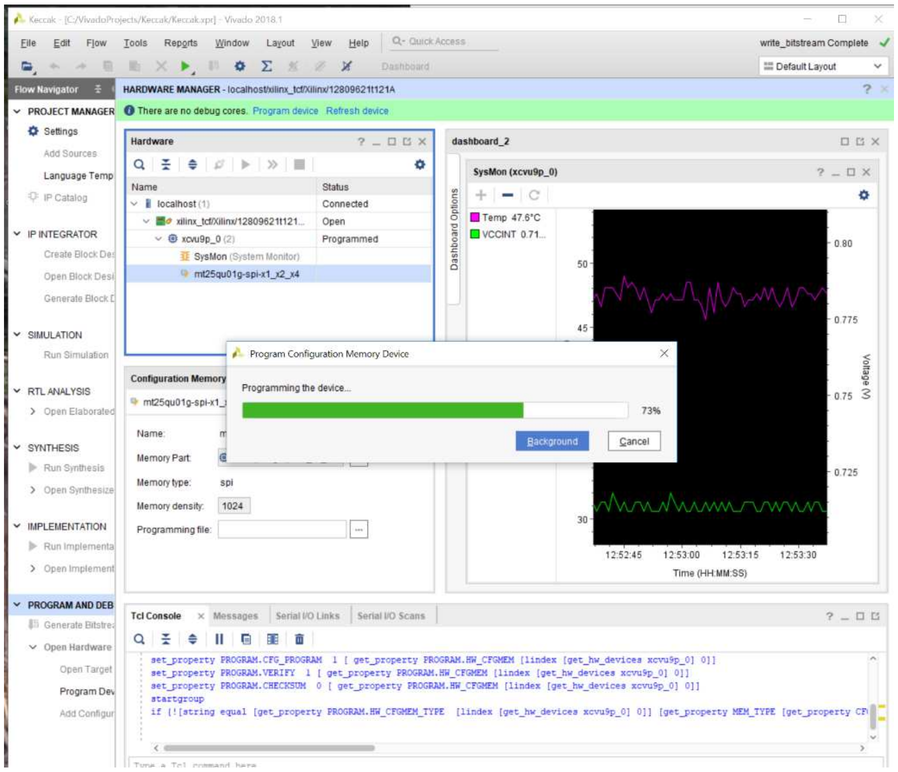
SVF Options

- [ ] Create SVF Only (no program operations)

SVF File:

```
WARNING: [Labtools 27-3361] The debug hub core was not detected.
Resolution:
1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active.
2. Make sure the BSCAN_SWITCH_USER_MASK device property in Vivado Hardware Manager reflects the user scan chain setting in the design
For more details on setting the scan chain property, consult the Vivado Debug and Programming User Guide (UG908).
```

In the above dialog, for 'Configuration File', choose the MCS file that you downloaded from the Zetheron website. Leave the rest of the options as their default (see above). Then click 'OK' to program the flash memory. Vivado will begin the programming procedure which can take 10-60 minutes:



Name	Status
localhost (1)	Connected
xilinx_tcf/Xilinx/12809621121...	Open
xcvu9p_0 (2)	Programmed
SysMon (System Monitor)	
mt25qu01g-spi-x1_x2_x4	

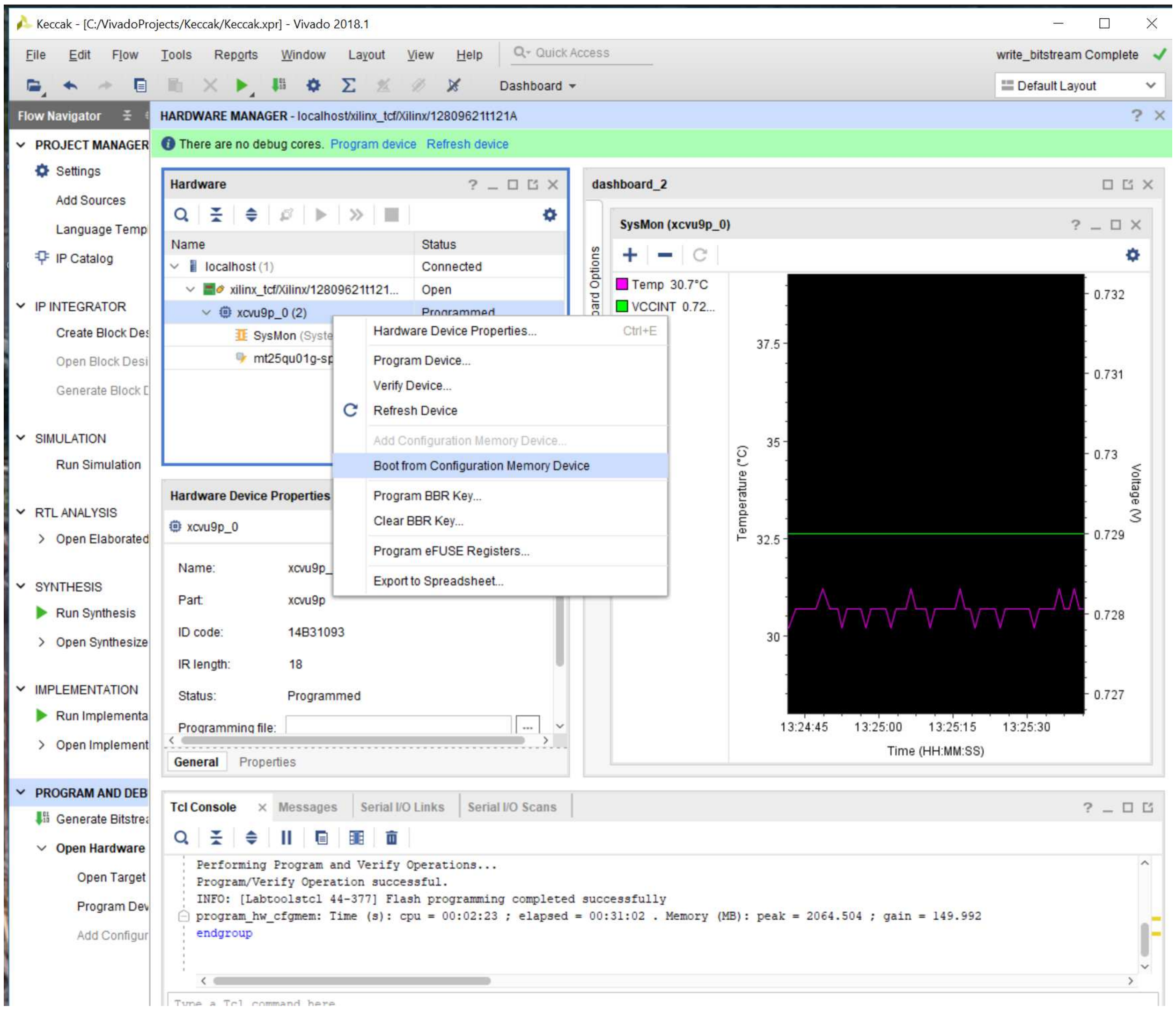
Program Configuration Memory Device

Programming the device...

73%

```
set_property PROGRAM.CFG_PROGRAM  1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices xcvu9p_0] 0]]
set_property PROGRAM.VERIFY  1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices xcvu9p_0] 0]]
set_property PROGRAM.CHECKSUM  0 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices xcvu9p_0] 0]]
startgroup
if {![string equal [get_property PROGRAM.HW_CFGMEM_TYPE  [lindex [get_hw_devices xcvu9p_0] 0]] [get_property MEM_TYPE [get_property CF
```

Once programming finishes, the flash memory is programmed, but that memory is external to the FPGA, and the FPGA itself is not yet programmed. Now we must transfer the contents of the external flash memory into the FPGA itself. You can do that by cutting power to the FPGA and then powering up again (resulting in the fastest programming), or alternatively you can do it in Vivado by right clicking on the FPGA icon and selecting 'Boot from Memory Configuration Device:'



Name: xcvu9p_
Part: xcvu9p
ID code: 14B31093
IR length: 18
Status: Programmed
Programming file:

```
Performing Program and Verify Operations...
Program/Verify Operation successful.
INFO: [Labtoolstcl 44-377] Flash programming completed successfully
program_hw_cfgmem: Time (s): cpu = 00:02:23 ; elapsed = 00:31:02 . Memory (MB): peak = 2064.504 ; gain = 149.992
endgroup
```

When you select Boot from Configuration Memory Device, the contents of the flash memory will be loaded into the FPGA itself and the FPGA is now programmed. This step can be fast or slow, it depends on many factors.

You are now finished the non-volatile setup procedure, and your FPGA card can now withstand a power-outage and start up on its own.

TROUBLESHOOTING

If you try 'Boot from Configuration Memory Device' and the process fails, try raising the core voltage on the FPGA. If it still fails, you may need to redo the process from the beginning. Once it is working, it should continue working without issues.