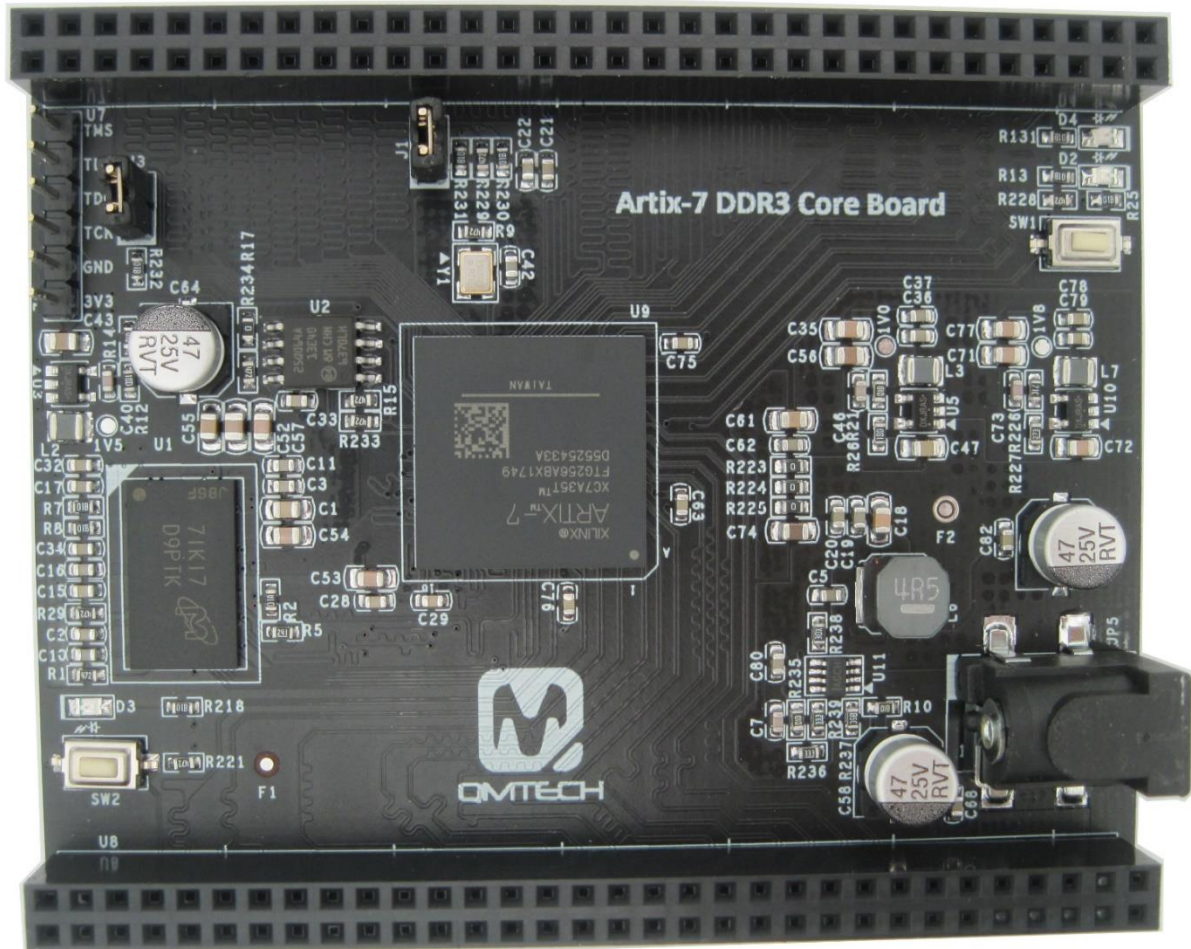


# QMTECH\_ARTIX7\_XC7A35T DB

## USER MANUAL(VIVADO 2018.2)



### Preface

The QMTECH<sup>®</sup> XC7A35T DDR3 Development Kit uses Xilinx Artix<sup>®</sup>-7 devices to demonstrate the highest performance-per-watt fabric, transceiver line rates, DSP processing, and AMS integration in a cost-optimized FPGA. Featuring the MicroBlaze<sup>™</sup> soft processor and 1,066Mb/s DDR3 support, the family is the best value for a variety of cost and power-sensitive applications including software-defined radio, machine vision cameras, and low-end wireless backhaul.

For more information, updates and useful links, please visit QMTECH Official Website:

<http://www.chinaqmtech.com>



QMTECH

## Table of Contents

1.	VIVADO 2018.2 INTRODUCTION .....	3
1.1	OVERVIEW .....	3
1.2	VIVADO_2018.2 ENVIRONMENT .....	3
2.	FPGA DOWNLOAD .....	4
3.	SPI FLASH PROGRAM .....	13
4.	REFERENCE .....	17
5.	REVISION .....	18

上海勤谋电子科技有限公司

# 1. Vivado 2018.2 Introduction

## 1.1 Overview

QM\_XC7A35T has mounted an 8MB SPI Flash provided by Micron, chip part number is N25Q064A. And the FPGA hardware design allows the FPGA boots from external SPI Flash after power up. The following chapters describes the FPGA download and SPI flash program by using Vivado 2018.2:

- (1) \*.bit file downloaded into FPGA, RAM based content will lost during power up stage;
- (2) \*.mcs file programmed into external SPI Flash, Flash based content is non-volatile and retained during power up stage.

## 1.2 Vivado\_2018.2 Environment

The test examples contained in QM\_XC7A35T release package are all developed with Xilinx ViVado 2018.2. Users could download the Vivado 2018.2 from Xilinx official website.

<https://www.xilinx.com/support/download.html>

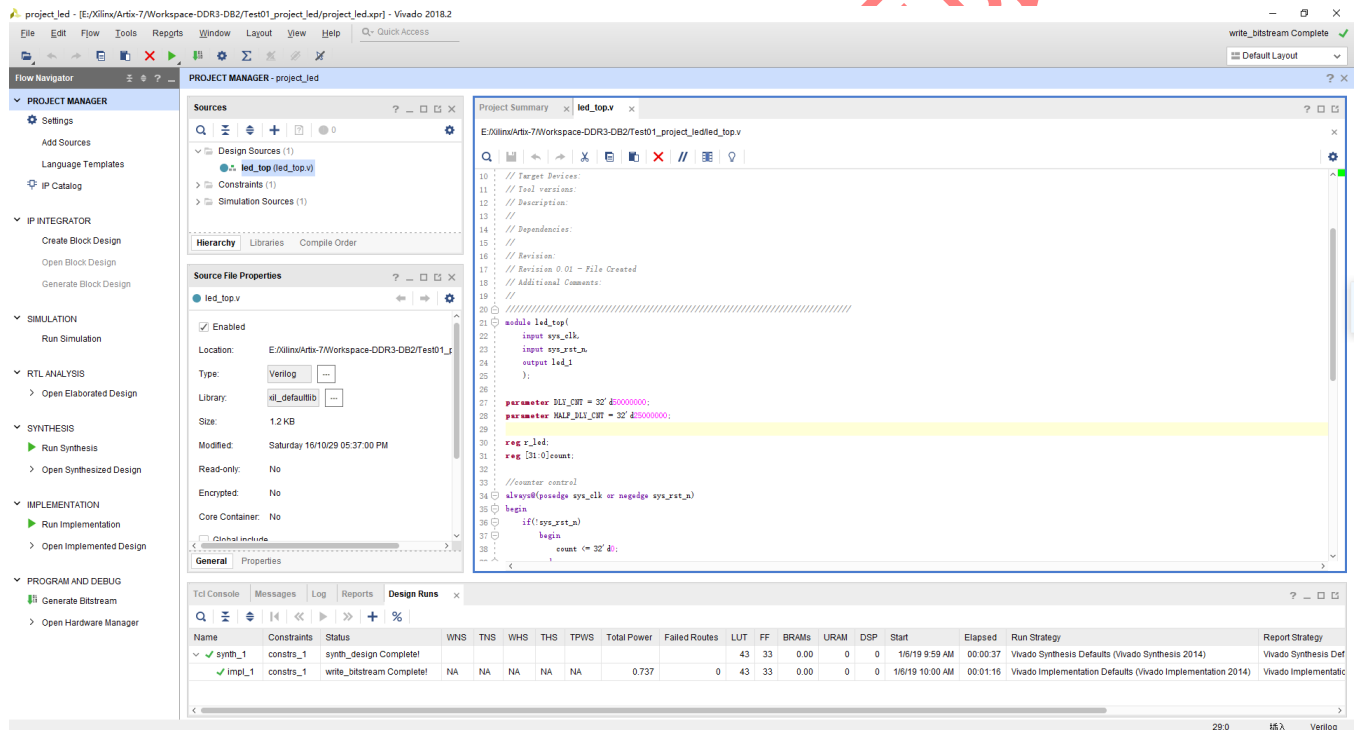


Figure 1-1. Vivado 2018.2 GUI

## 2. FPGA Download

Users could download \*.bit file directly into FPGA to verify the RTL behavior performs correctly or not. In this section we use the example project Test01\_project\_led to demonstrate the procedure of downloading led\_top.bit into FPGA.

First step is to make the example project Test01\_project\_led compiled without any error generated. Since this test example is already verified, users could click “Flow Navigator” → “Project MANAGER” → “PROGRAM AND DEBUG” → “Generate Bitstream” to get the led\_top.bit generated directly. If users want to test with some customized project, please follow below steps to generate the \*.bit file.

Users could click the **【Run Synthesis】** button shown in below image highlighted with red rectangle to SYNTHESIS the example project. The SYNTHESIS progress is displayed in the tab of “Design Runs” which is also highlighted in below image. Users could get the compile info in the tabs like “Log”, “Message”:

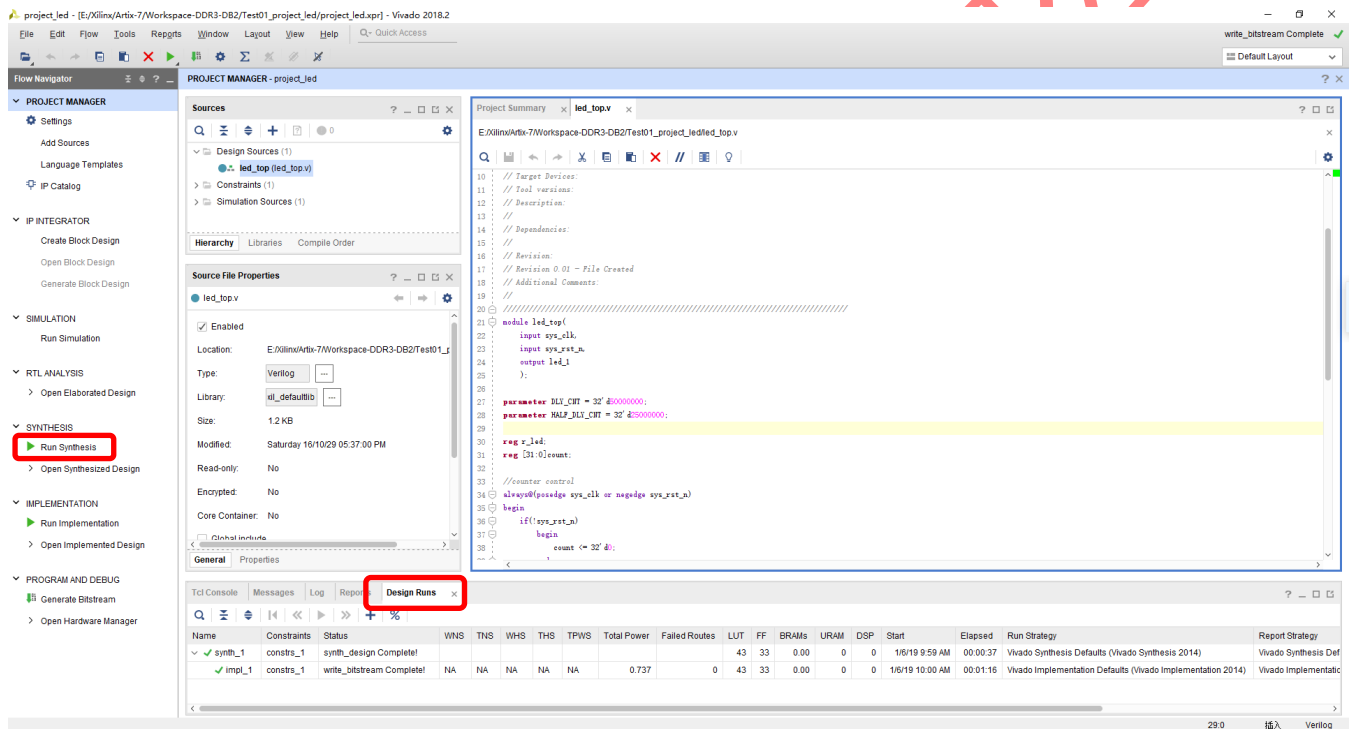


Figure 2-1. SYNTHESIS

There will be a “Synthesis Completed” window popup once the example project successfully synthesized.

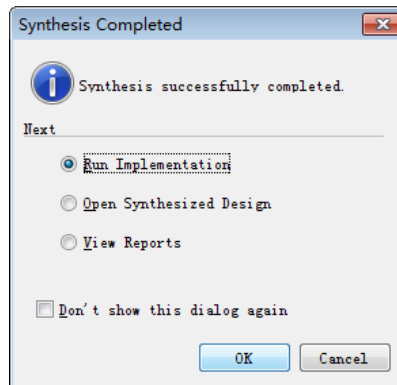
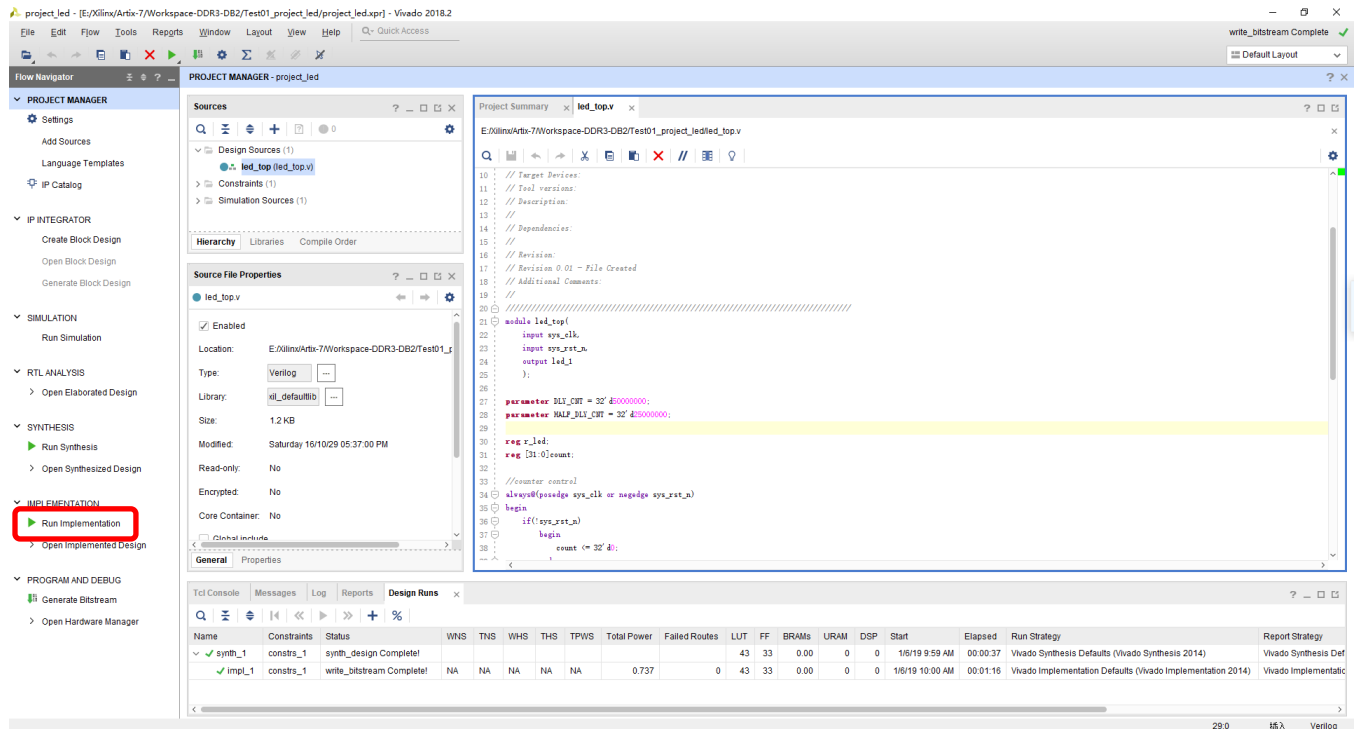


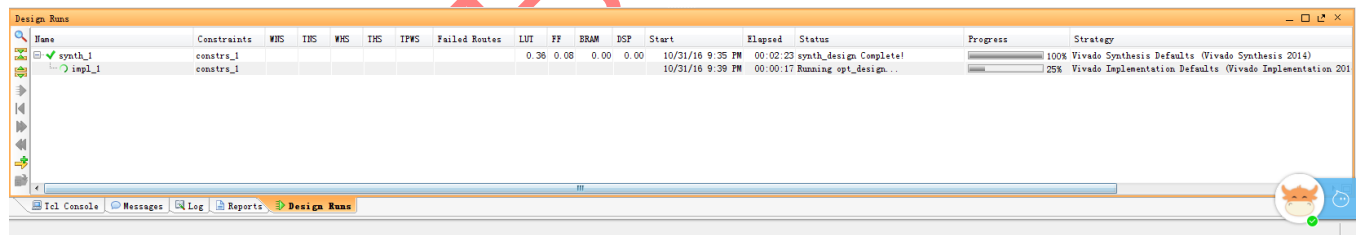
Figure 2-2. Synthesis Completed

Users could press the **【ok】** button shown in the previous image to start the project “Implementation”. Or press the **【Cancel】** button shown in the previous image and then click “Run Implementation” button highlighted with red rectangle in below image to start a new implementation.



**Figure 2-3. Run Implementation**

The implementation progress info could be retrieved from the “Design Runs” window. Users could get the implementation info in the tabs like “Log”, “Message” and “Reports”:



**Figure 2-4. Implementation**

If there's no error generated during Implementation process, users could start the \*.bit file generation stage. Please make sure the constraint file LED.xdc is already contained in the project.

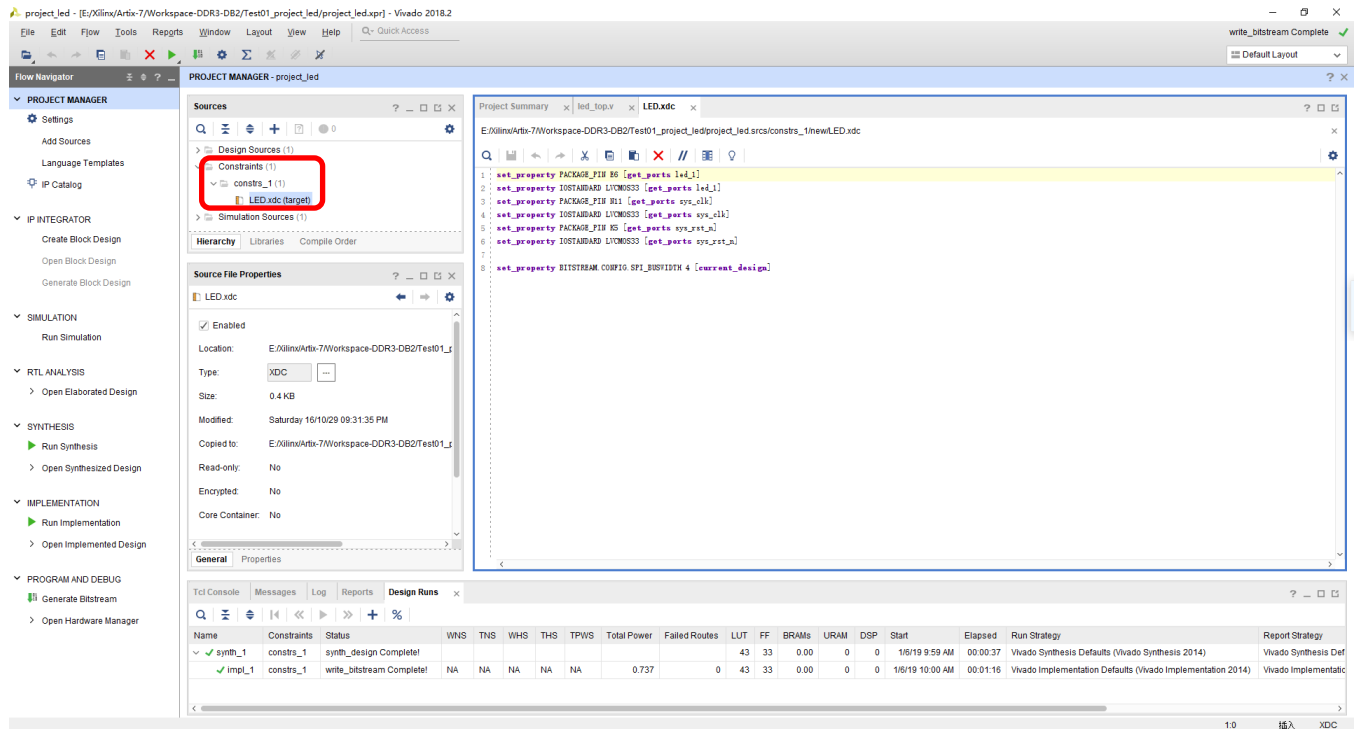


Figure 2-5. LED.xdc Constraint File

There'll be a Implemented Completed window popup once the Implementation process finished. Users could click **【OK】** to start the bitstream generation.

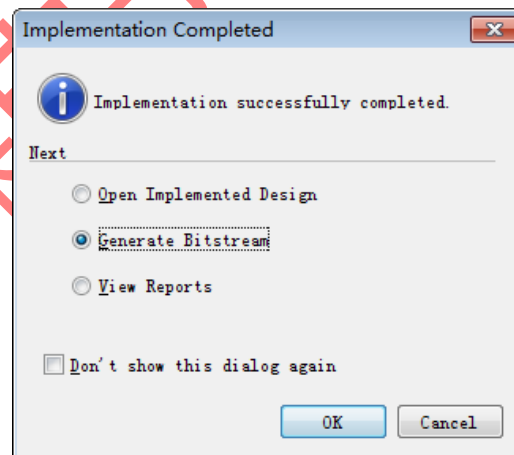


Figure 2-6. Generate Bitstream

Users could also click the **【Cancel】** button shown in the previous image and start to generate the bitstream manually. And then users could click **【Generate Bitstream】** shown in below image to start the bitstream generation.

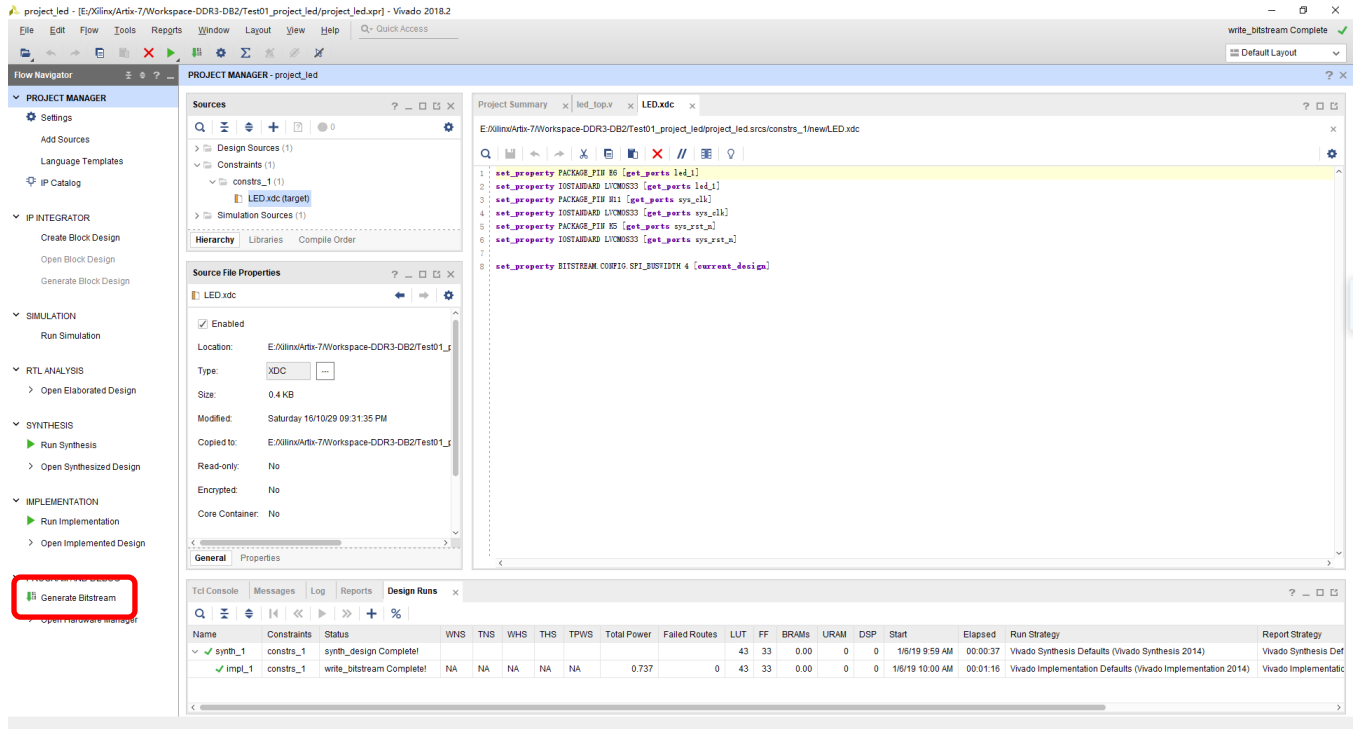


Figure 2-7. Generate Bitstream

Once the \*.bit correctly generated without any error, users could download this \*.bit into FPGA directly. Make sure the Xilinx USB platform cable is correctly connected to the FPGA board's JTAG interface. And then click the **【Open Target】** button shown in below image:

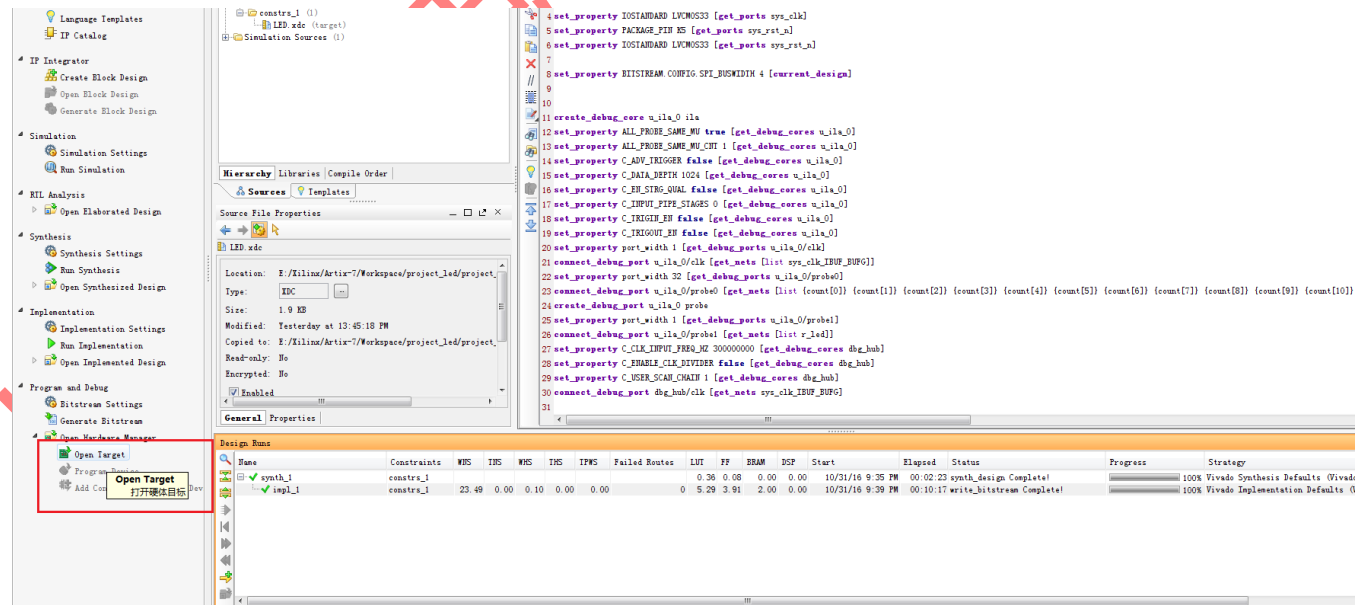


Figure 2-8. Open Target



Click **【Next】** button:

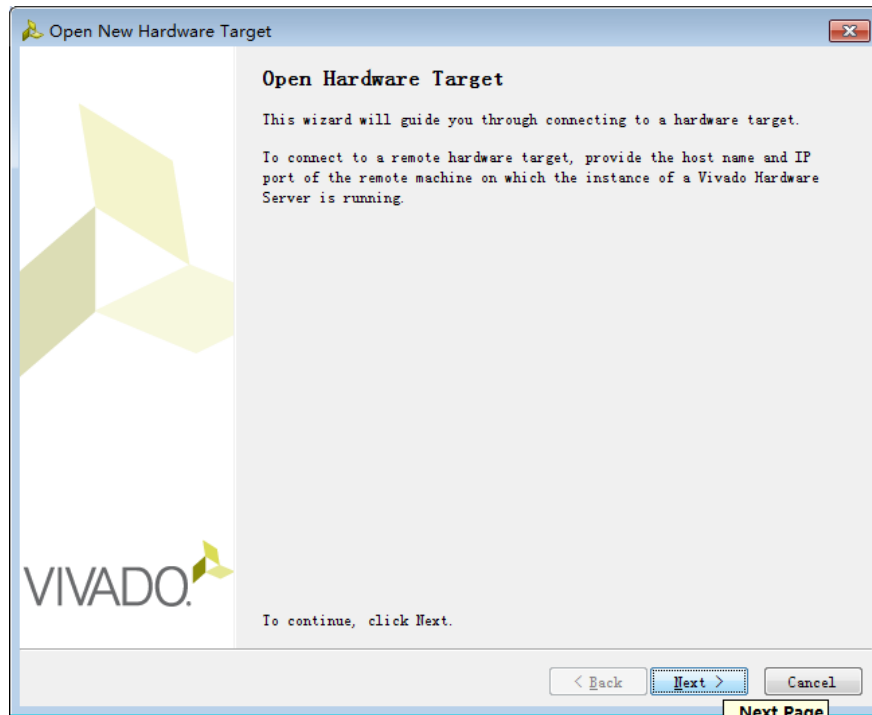


Figure 2-9. “Next”

Click **【Next】** button:

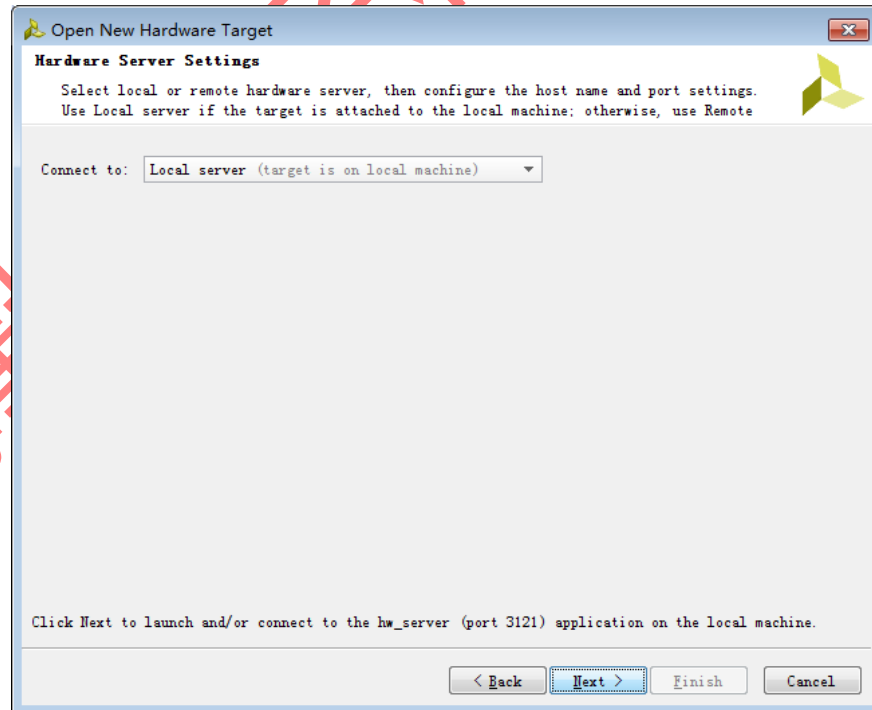


Figure 2-10. “Next”



Below image shows that the Hardware Target “xc7a35t\_0” is successfully detected, and then click 【Next】 :

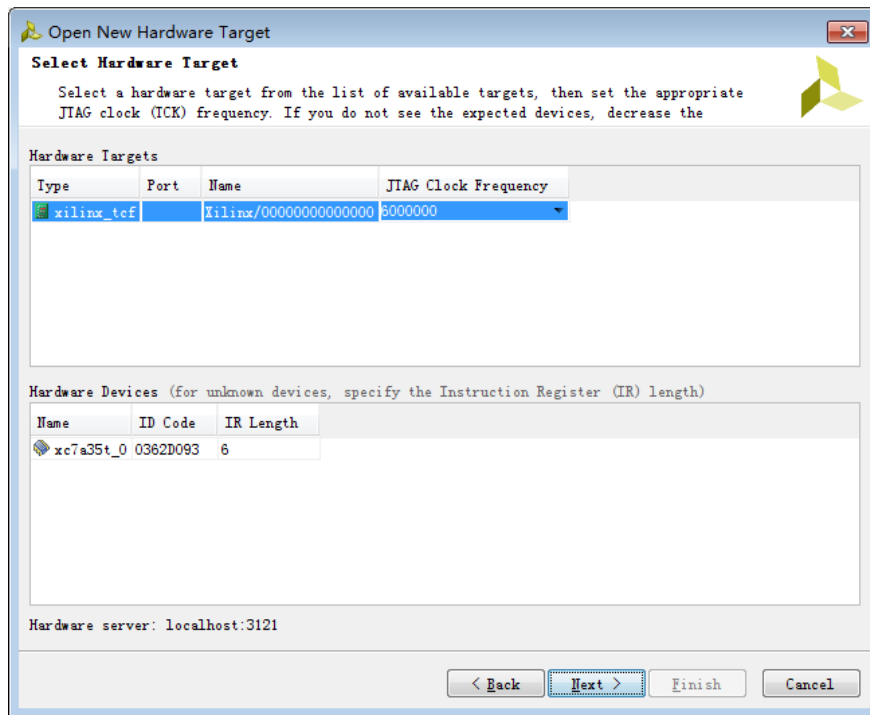


Figure 2-11. Target Detected

Click 【Finish】 button:

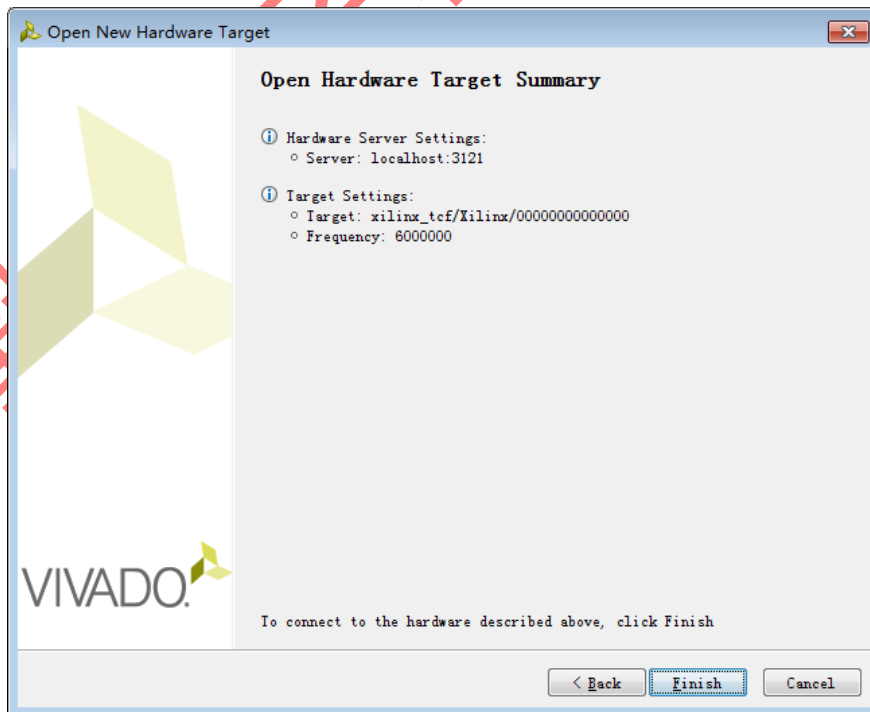
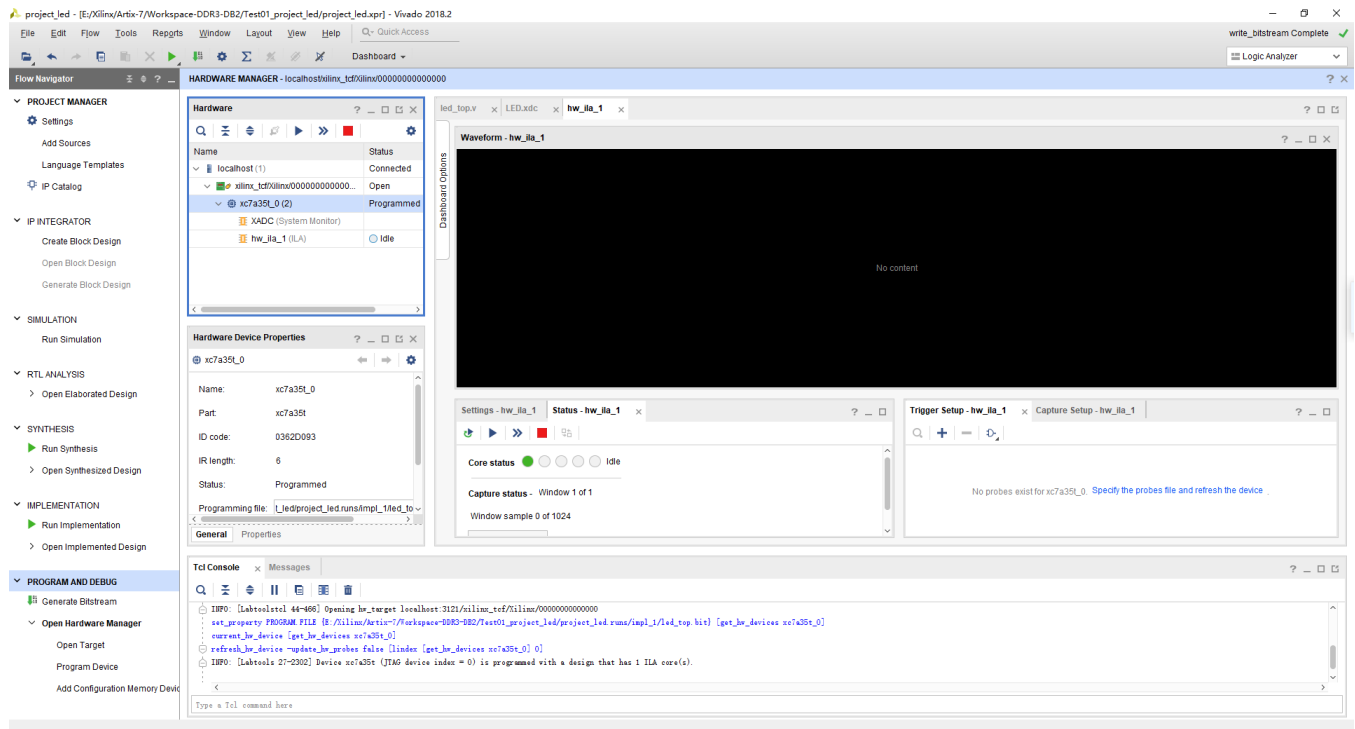


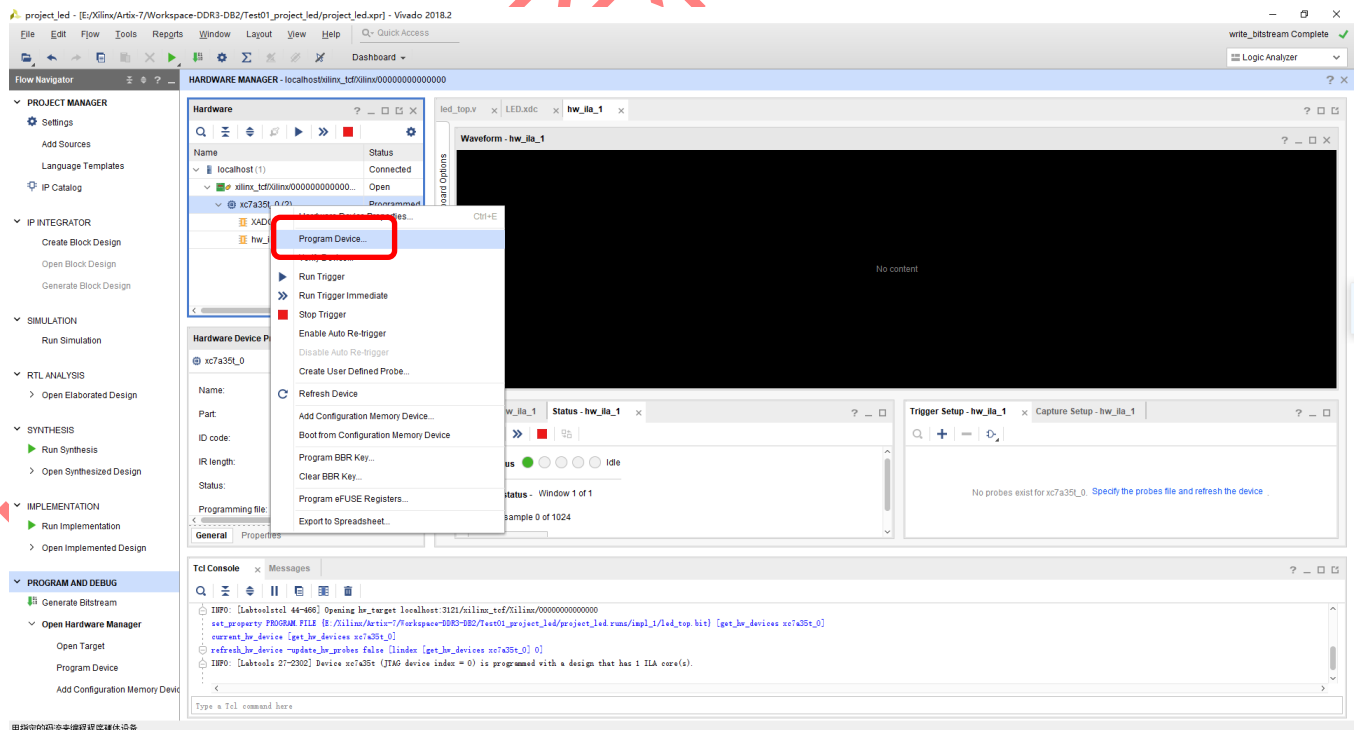
Figure 2-12. “Finish”

Below image shows the main page of the “Hardware Manager”:



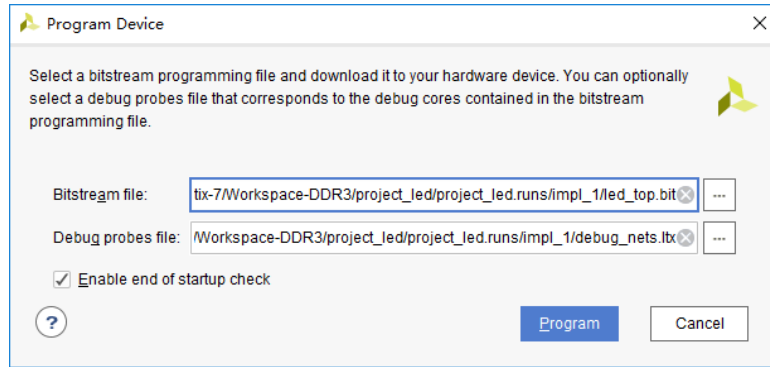
**Figure 2-13. Hardware Manager**

Right click the detected chip “xc7a35t\_0 and choose 【Program Device】 to start the \*.bit file download:



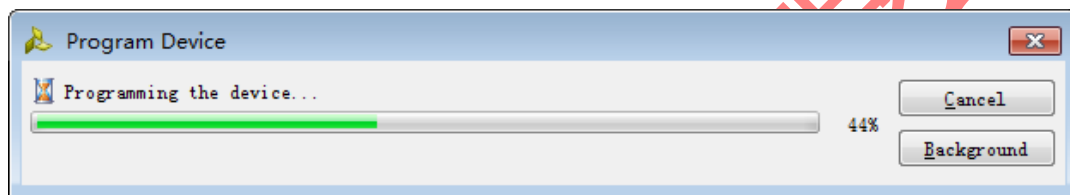
**Figure 2-14. Program Device**

Choose the previously generated led\_top.bit file and click the 【Program】 :



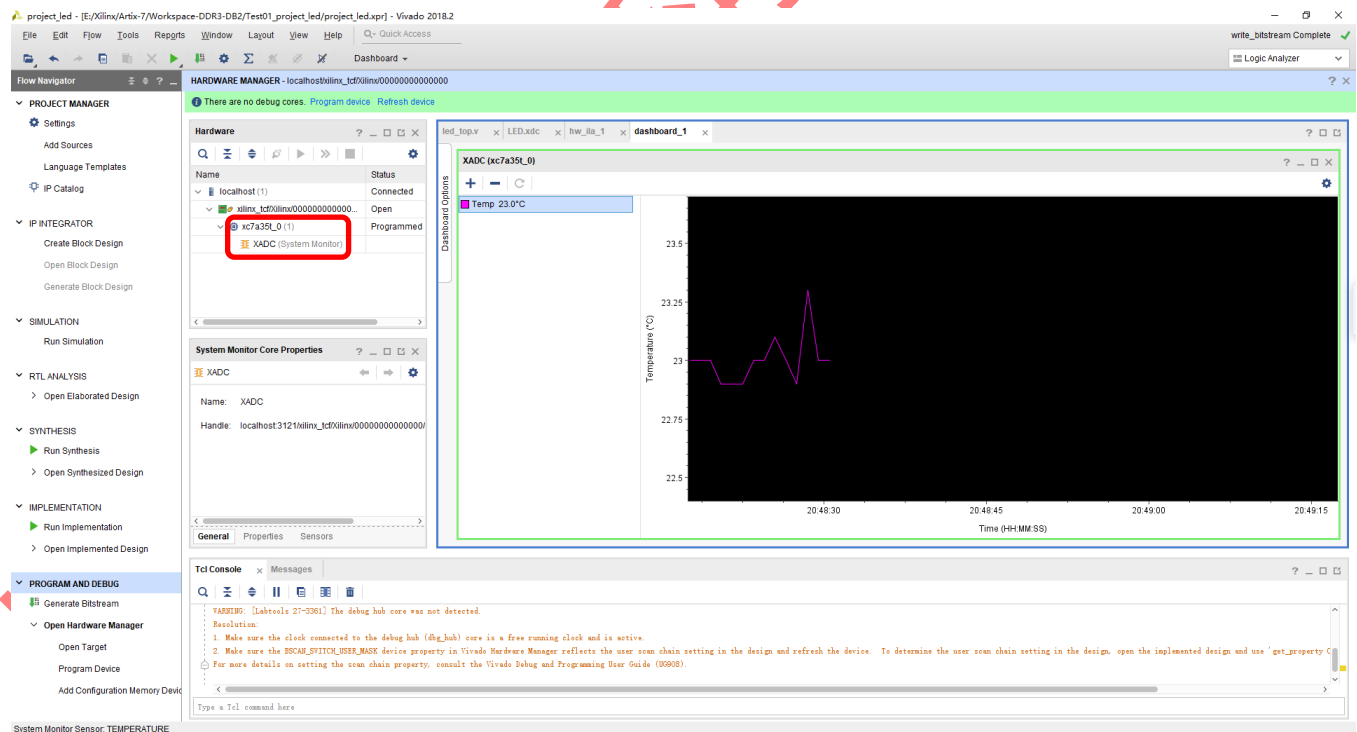
**Figure 2-15. Choose \*.bit file**

Below image shows the progress of the FPGA downloading:



**Figure 2-16. Progress Bar**

Users could monitor the FPGA internal XADC, Core Voltage supply status, and die temperature by clicking 【XADC (System Monitor)】 shown in below image:



**Figure 2-17. XADC**

If there's ILA debug core embedded in the example project, users could use this tool to Monitor the waveform and Debug the RTL code. Users could double click the 【hw\_ila\_1 (ILA)】 button and then the waveform monitor window “ILA – hw\_ila\_1” will be displayed. The sampling signals and sampling buffer

depth could be configured in the Properties tab. Users could click the **【Run Trigger Immediate】** to start waveform capture.

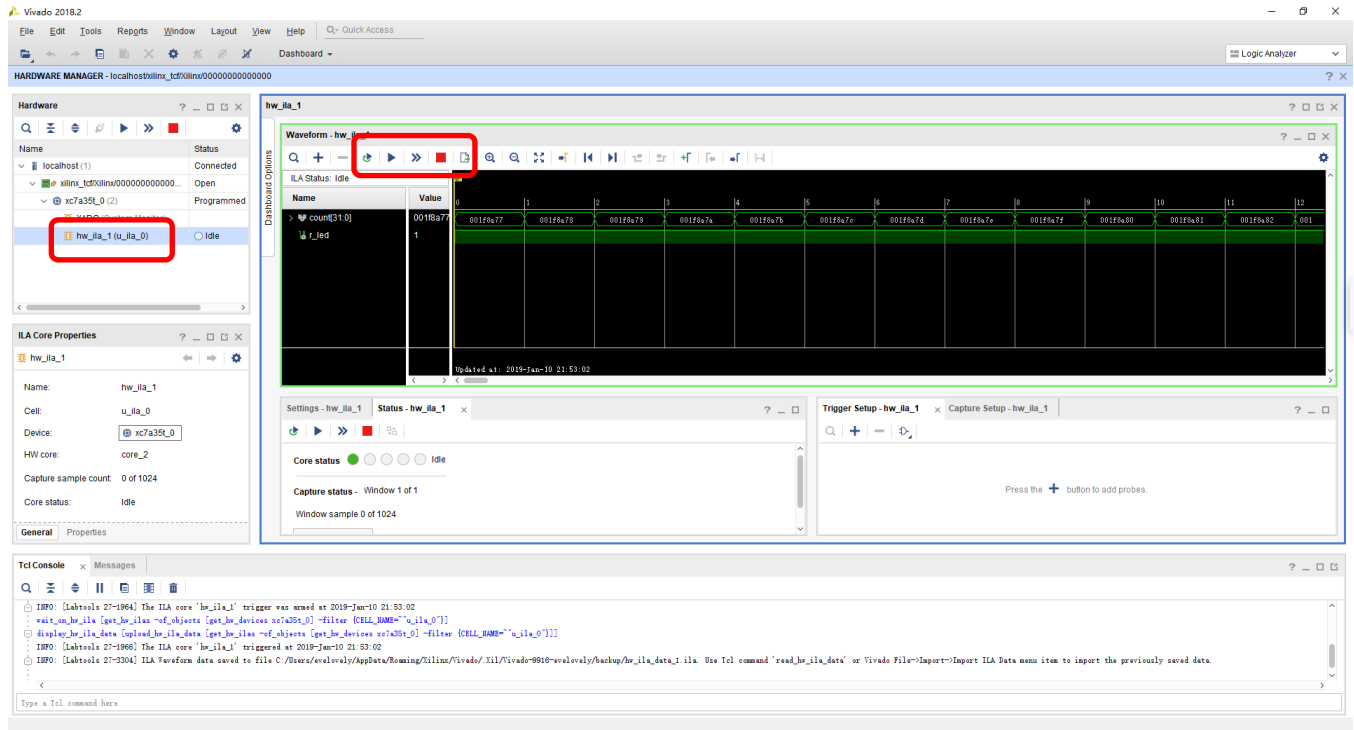


Figure 2-18. ILA Debug

Waveform in hw\_ila\_1:

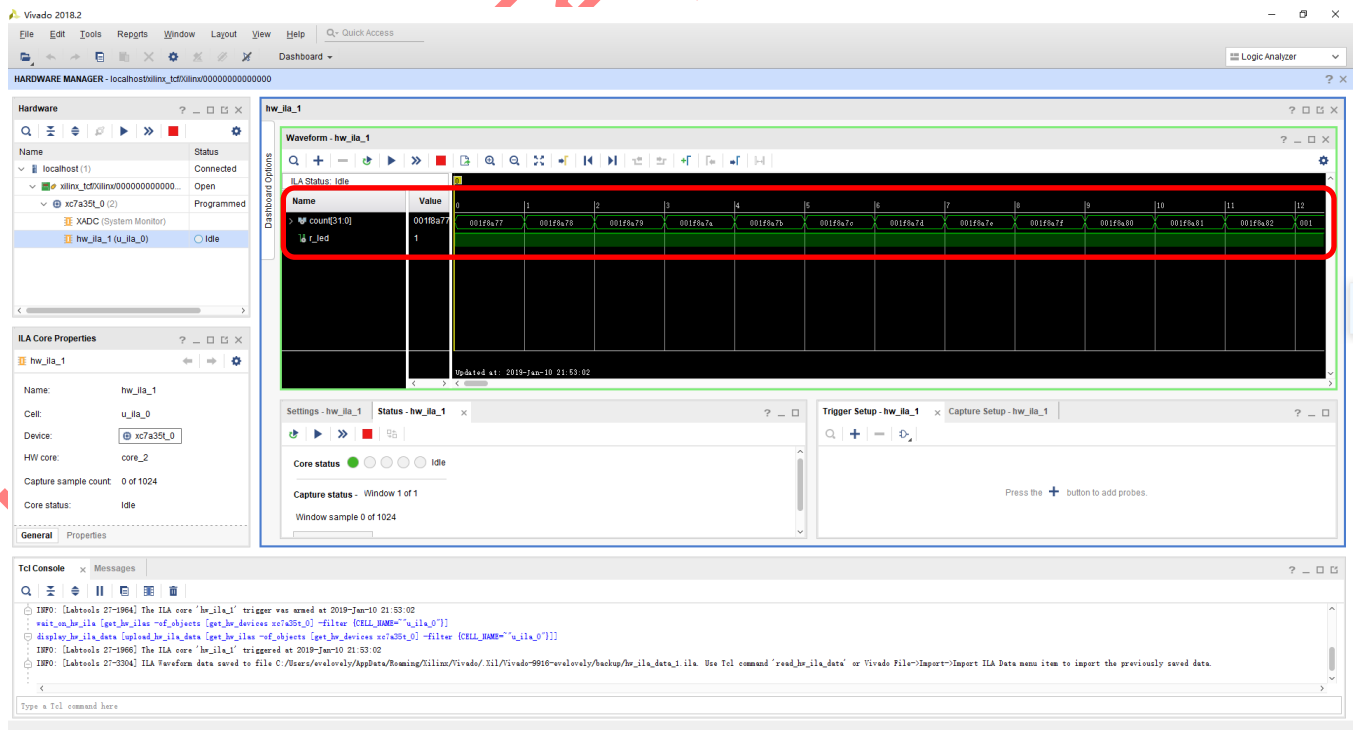


Figure 2-19. Waveform

### 3. SPI Flash Program

In the previous chapter, we described how to use Vivado Program tool to download \*.bit file into FPGA. But the storage memory embedded in FPGA is SRAM based which means all the content will be flushed during the power on stage. On the QM\_XC7A35T core board, there's a non-volatile SPI flash mounted. And the XC7A35T FPGA supports to load bitstream from external SPI flash during power on. In this chapter we will introduce the way to program the bitstream into SPI flash.

Since the \*.bit file could not be programmed into SPI flash directly, the file format conversion needs to be done at the very beginning stage.

Users could use Vivado2018.2 【Generate Memory Configuration File】 to convert the led\_top.bit into led\_top.mcs file. Below image shows where this file format convert tool locates:

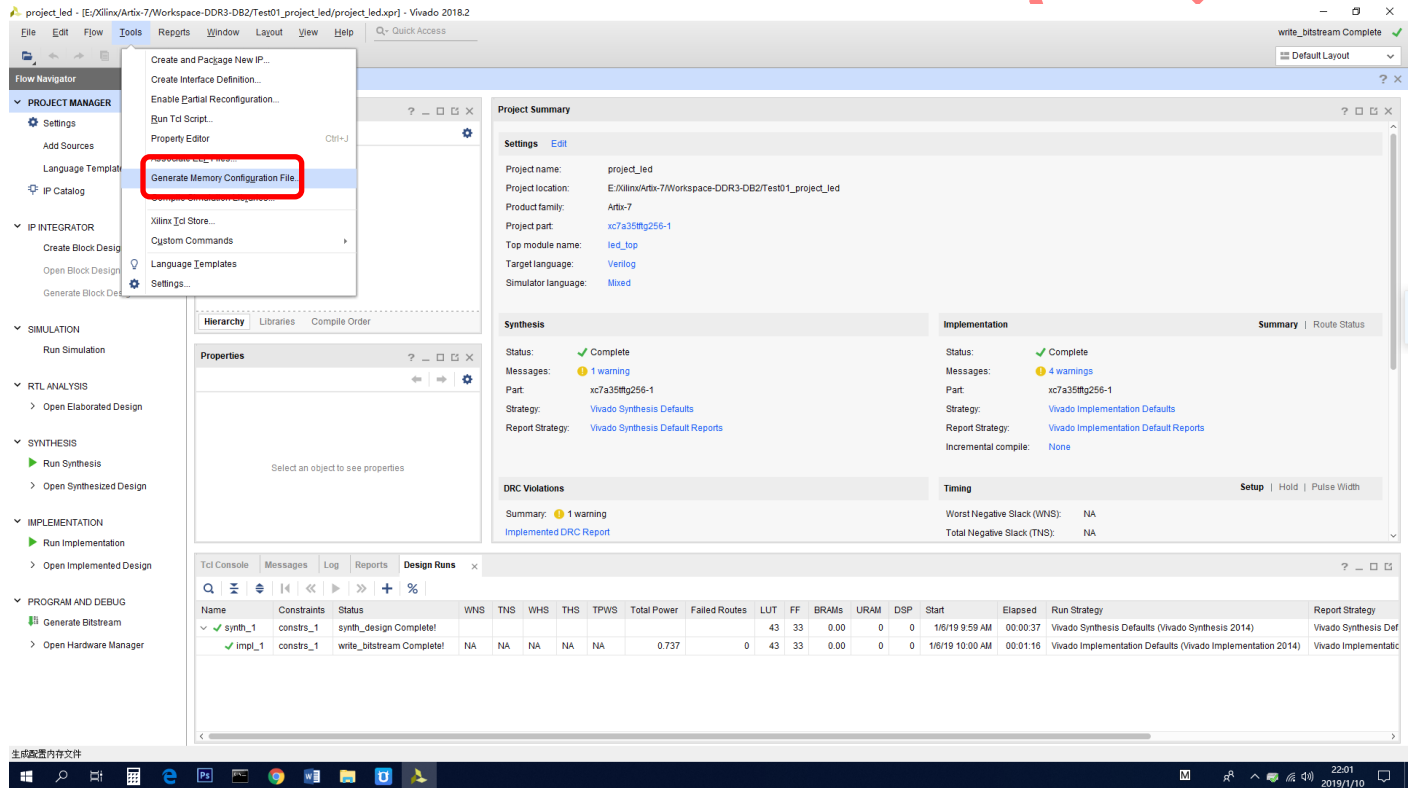
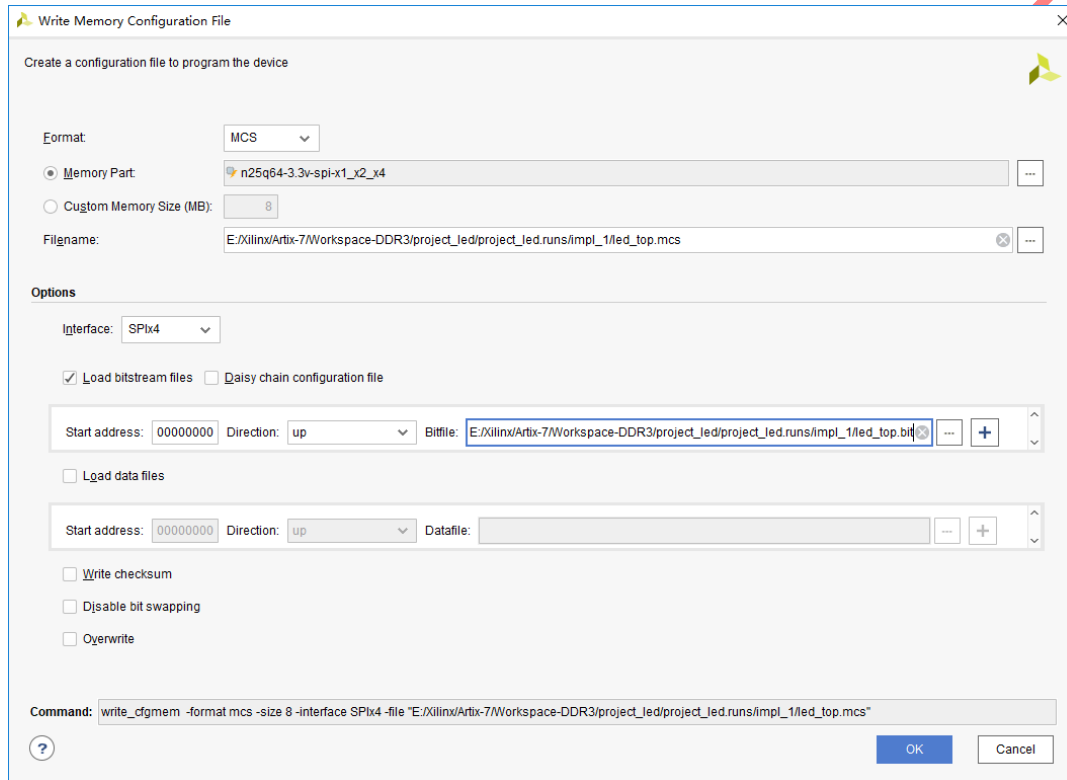


Figure 3-1. Generate Memory Configuration File Tool

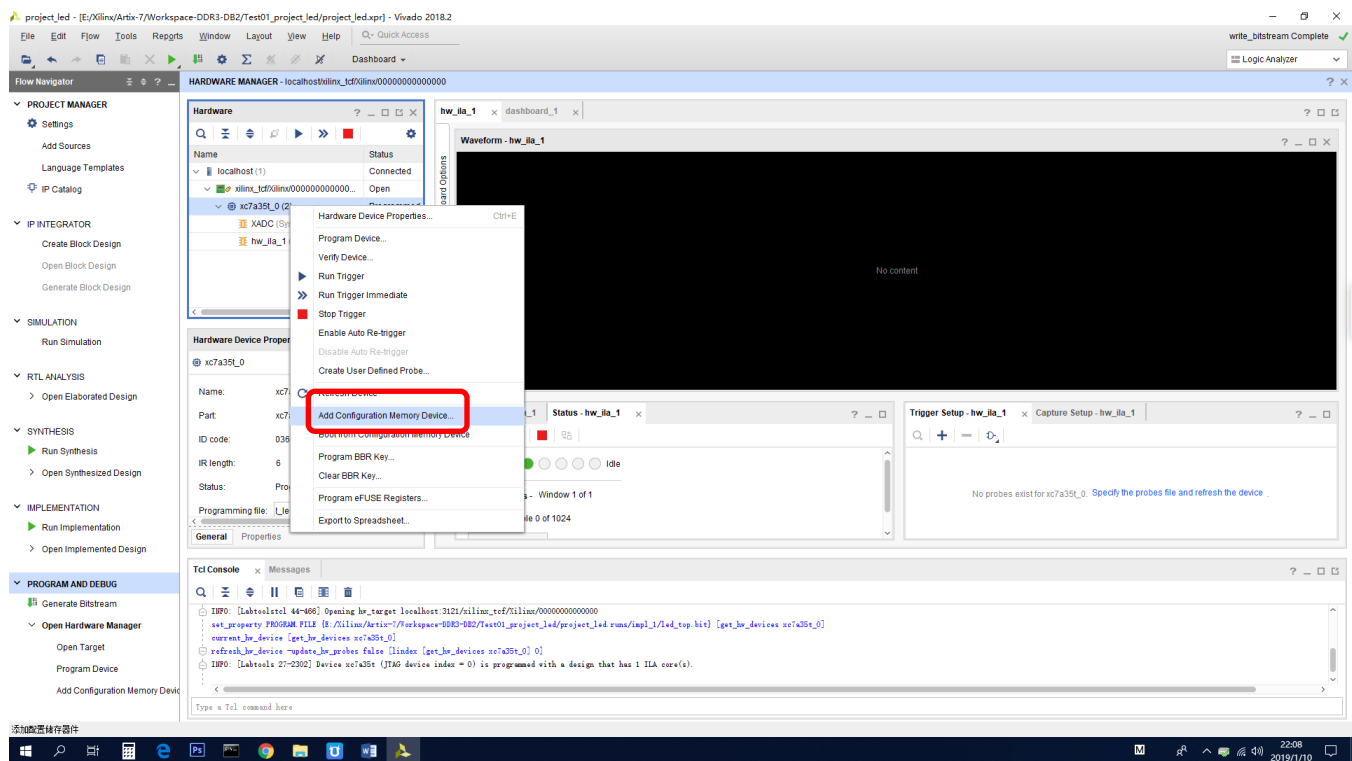
Below image shows the example configuration for the file format conversion:

- SPI Flash Part Number: N25Q064A
- SPI Flash bus width: SPIx4
- Input File: led\_top.bit
- Output File: led\_top.mcs
- Start Address: 0x000000



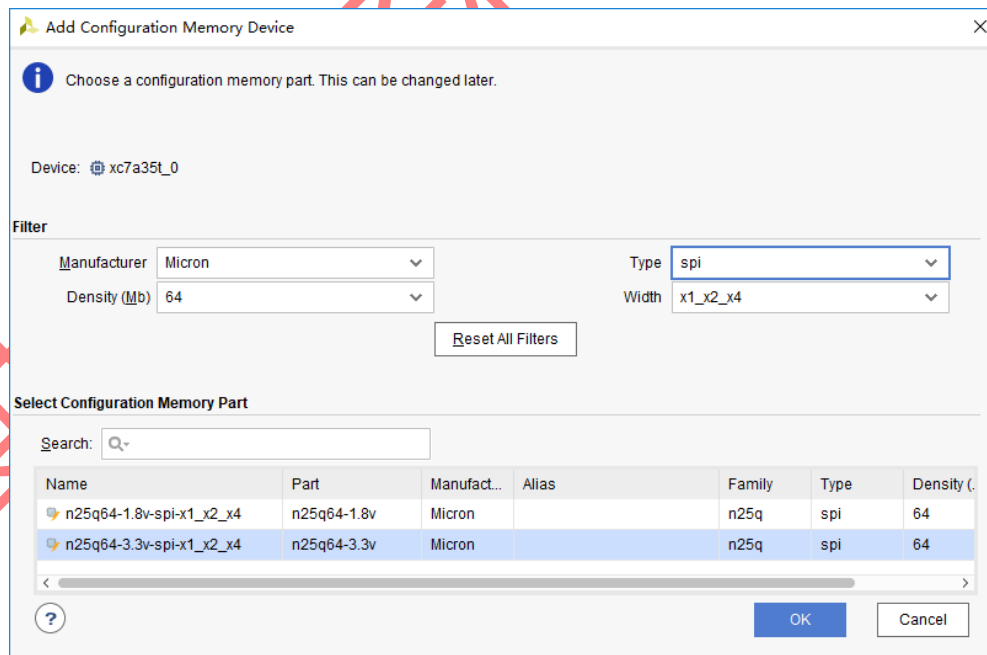
**Figure 3-2. Configuration**

Once the led\_top.mcs file successfully generated, users could program this file into the SPI Flash. Make sure the Xilinx USB Platform Cable is correctly connected to FPGA's JTAG interface. And use Hardware Manager to connect the device "xc7a35t\_0 (1)". Then right click on the detected FPGA device and choose "Add Configuration Memory Device..". Below image shows the program procedure.



**Figure 3-3. Add Memory Device**

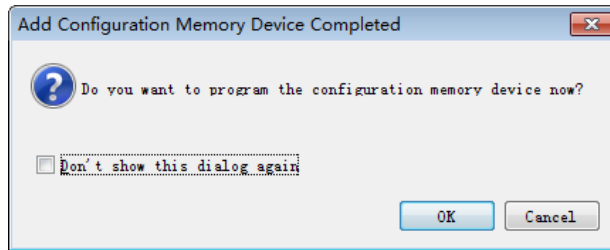
Choose the SPI Flash chip part: N25Q064A provided by Micron. And then click **【OK】** button:



**Figure 3-4. Choose SPI Flash**

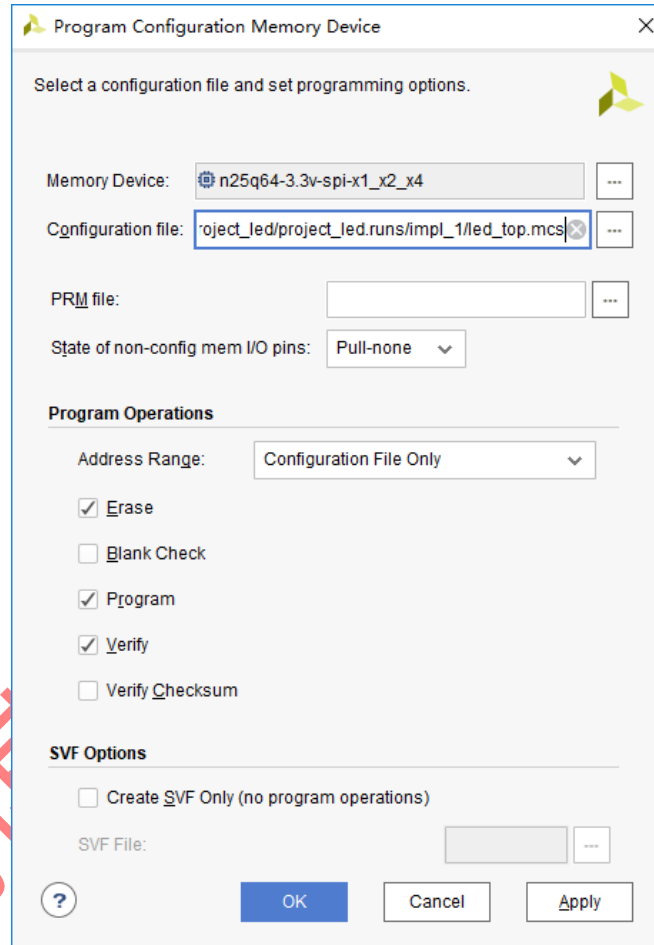
Click **【OK】** button shown in below image to start the Program:





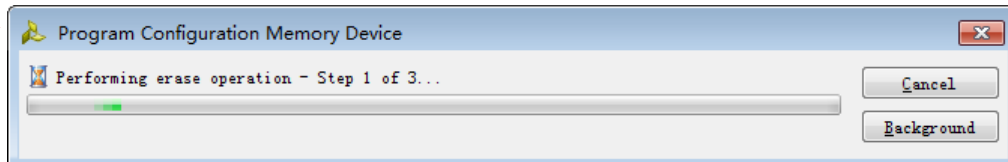
**Figure 3-5. Start to Program**

Choose the led\_top.mcs file and click **【OK】** button.



**Figure 3-6. Start to Program**

Below image shows the progress of the SPI flash programming.



**Figure 3-7. Progress Bar**

Once the program is successfully finished, users could repower on the board to check whether the FPGA correctly loads the content from SPI flash and implemented functionality is correctly running on FPGA.

#### 4. Reference

- [1] ug470\_7Series\_Config.pdf
- [2] ds181\_Artix\_7\_Data\_Sheet.pdf
- [3] ug475\_7Series\_Pkg\_Pinout.pdf
- [4] n25q\_64a\_3v\_65nm.pdf
- [5] MT41J128M16JT-125K.pdf
- [6] MP2359.pdf
- [7] NCP1529-D.PDF

上海勤谋电子科技有限公司

## 5. Revision

Doc. Rev.	Date	Comments
0.1	1/1/2018	Initial Version.
1.0	12/1/2018	V1.0 Formal Release.
2.0	12/01/2019	V2.0 Formal Release.
3.0	10/01/2019	Software Upgraded to Vivado 2018.2.

上海勤谋电子科技有限公司