TEST YOUR KNOWLEDGE – KNIGHT RIDER ROM

Last updated: **[July 9, 2021]**

Contents

1.0 Introduction

1.1 Prerequisites

- Completion of Introduction to Intel FPGAs and Quartus Software course
  - Link to self-guided course: OUWINTRO
  - This lab will begin from a completed version of your Knight Rider project from the last section of OUWINTRO. Please complete that lab before doing this one.
- Basic knowledge of Verilog and digital design

1.2 Reference Documents

Table 1-1. Reference Documents

The starting file for this project is a working knight_rider project from part 3 of the Introduction to Intel FPGAs and Quartus Prime Software course. This .qar file has pin assignments set for a DE10-Lite board.

If you have a different FPGA board (such as the DE1-SoC), you will need to do project set up and pin assignments **based on previous projects**.

Document	Document No./Location
Knight Rider starter file	knight_rider.qar

2.0 Lab Guide

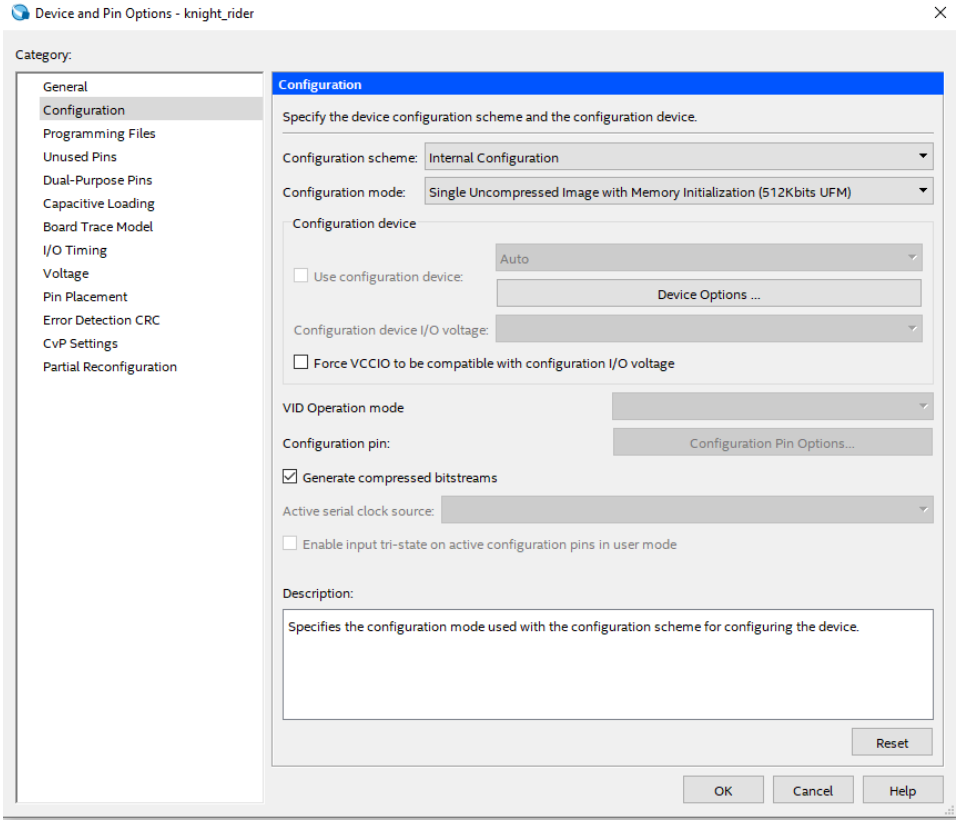
Now that you have completed the Knight Rider LED sequencing circuit on the DE10-Lite development kit, next we will use a powerful debug/bring-up technique to allow you to make parameter changes to the Verilog code. Specifically, you might have taken a few iterations to derive the parameter COUNTER_SIZE parameter to come up with the proper clock frequency for the LEDs to change at approximately 10Hz, which makes the LEDs stay on for approximately 1/10 of a second. Each time you change the value of COUNTER_SIZE, you spend 1-2 minutes recompiling your code and downloading the programming image (.sof file) to the DE10-Lite kit. We will investigate new means to tune the value of COUNTER_SIZE so that you can quickly change its value without recompiling your design.

One method to change the value of the clock divider output clock would be wire up the select signal to switches on the DE10-Lite board instead of hard wiring it to your calculated value. However, a more elegant way to introduce programmability into this circuit is to add a Read-Only Memory (ROM) block to your design and change the values of the ROM with a tool that is called the In-System Memory Content editor. After loading the image, you will now be able to change the values of the ROM through a user interface and watch the LEDs sequence at different rates through simple edits of the ROM values without recompiling your design.

2.1 Memory Initialization File setup

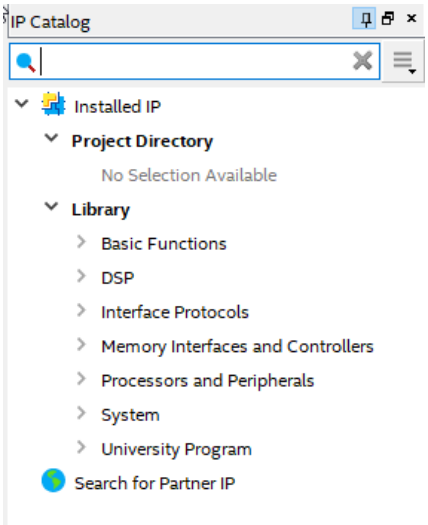
Note you will need a .mif file. Set the name of the MIF file in the ROM IP editor. The quickest for a new MIF file: File → Memory Initialization File. This will create a memory initialization file. Enter the depth and width of the ROM that you constructed in the memory editor. Make sure you name the MIF file the same name that you have entered in the IP editor.

There is a device setting you will need to make compilation work properly. Change Assignments → Device → Device and Pin Options → Configuration → Select Single Uncompressed with Memory Initialization. Without this change the compilation will not work.

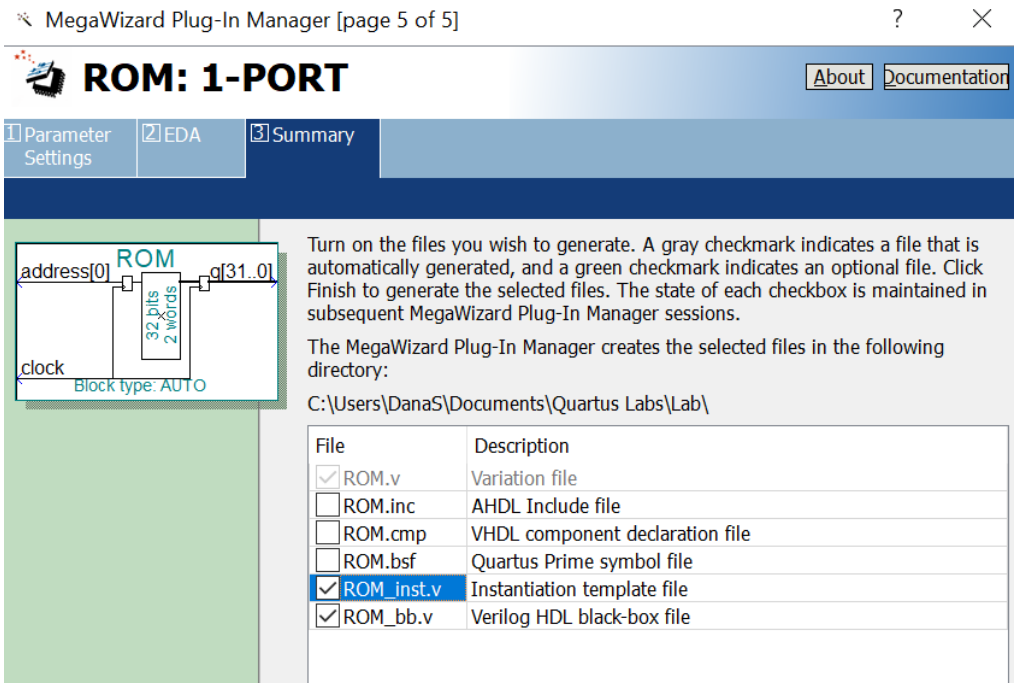


2.2 ROM setup

Next, launch the ROM editor from the IP Catalog:



Enter ROM in the search field. Select 1-Port ROM. Make the name of the new IP block ROM. Edit the fields so you can make the smallest ROM possible to hold a single word that you can store the values of the COUNTER_SIZE parameter. Use Auto / Single Clock defaults. Hit next and select the simplest configuration that will meet your needs. Hit next and enter a file name to store the ROM contents (e.g. clock_divider_tap.mif). Select Allow In-System Content Editor box – you will use this feature. Hit next with defaults. In the last configuration panel **select Instantiation Template** and hit finish.



2.3 ROM instantiation in knight_rider.v

Now you will need to instantiate your ROM module into your knight_rider.v file. Find the ROM_inst.v file for hints on how to add this module to your design.

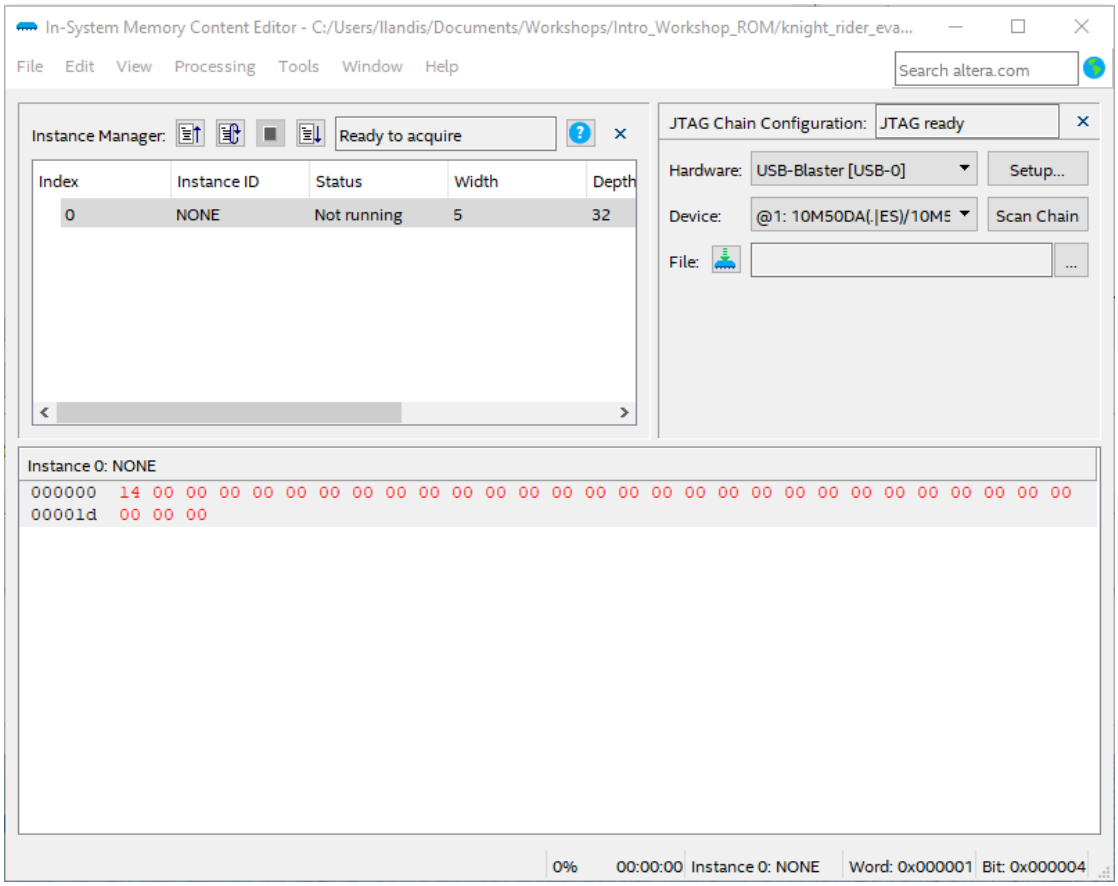
Hints for instantiating and connecting ROM module to your design and debugging:

- The ROM module should be instantiated inside the clock divider module
  - The input address can just be wired to 0
- Change the COUNTER_SIZE to 32 - there should be a wire that can represent all 32 values (how many bits?)
- Use a MUX to select between a default value for COUNTER_SIZE (like the number you used in the knight_rider lab) and the ROM output controlled by a switch
  - Initialize the switch using [0:0] SW in all the modules, in order to use the SW[0] which is already in the assignments
- The slow_clock is what must be assigned using the counter tap wire

Compile, work out syntax errors and download to your DE10-lite or DE1-SoC kit. Now launch Tools → In-System memory Editor.

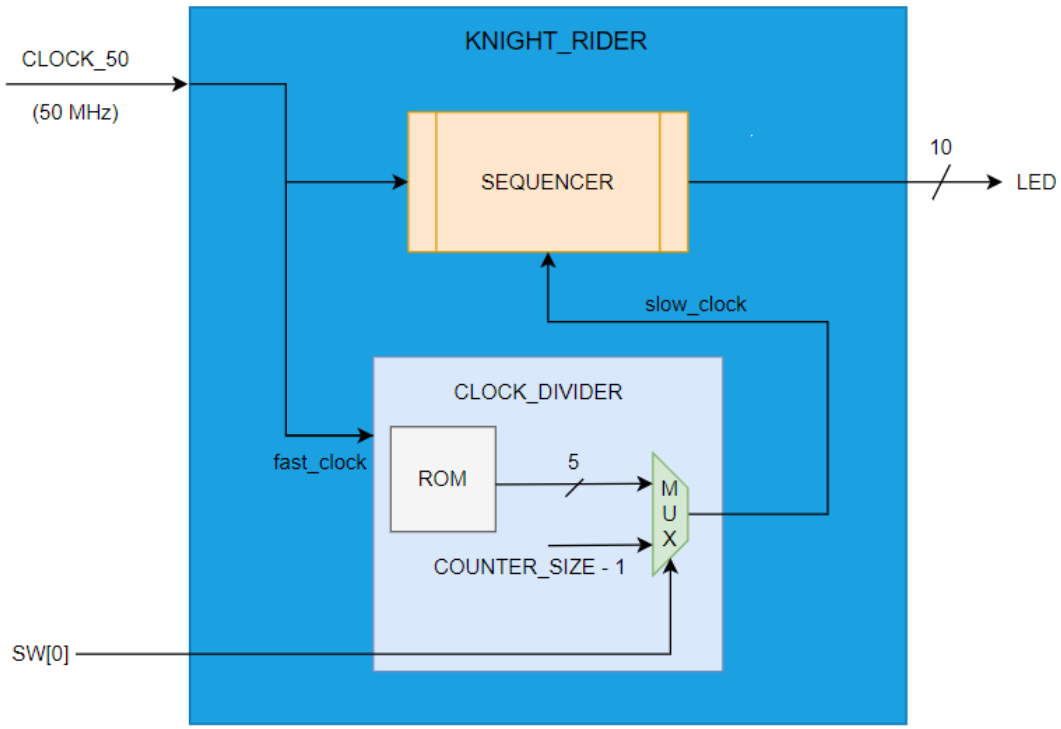
2.4 In-System Memory Content Editor

Note: The In-System Memory Content Editor displays values in hexadecimal representation



If Hardware isn't showing, select the USB blaster. Hit F7 to update the ROM map. Change the value in the ROM and hit F7 again to update to the hardware. A lower number will make the LEDs switch faster, while a lower number will indicate a slower rate of transition. Watch the LEDs blink at a different rate.

3.0 Top Level Schematic



The switch should control whether the slow_clock is assigned the default value of 23 or the value from the ROM.

4.0 Document Revision History

Date	Author	Comments
6/25/2021	RK	— Transferred guide to common Word template and added some hints
7/9/2021	RK	— Added better formatting and a few extra notes throughout the guide