



TEST YOUR KNOWLEDGE – KNIGHT RIDER ROM

Last updated: **[June 28, 2021]**

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

Copyright © 2021, Intel Corporation. All rights reserved.



Contents

1.0	Introduction.....	3
1.1	Prerequisites	3
1.2	Reference Documents.....	3
2.0	Lab Guide.....	4
3.0	Top Level Schematic	8
4.0	Document Revision History	9

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

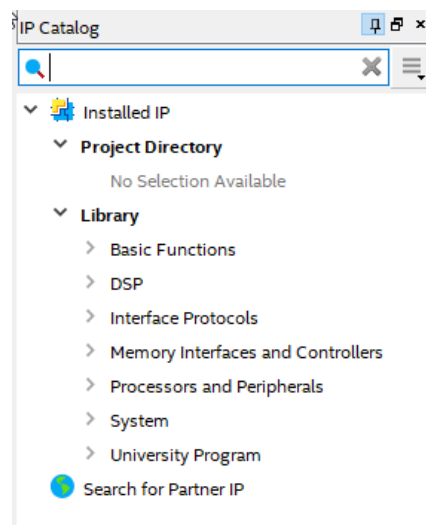
Document	Document No./Location
Knight Rider starter file	

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` 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.

Launch the ROM editor from the IP Catalog:



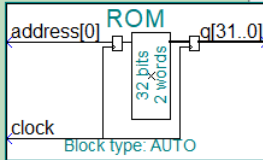
Enter ROM in the search field. Select 1-Port ROM. Make the name of the new IP block ROM. Edit the fields so you can make the smallest ROM possible to hold a single word that you can store the values of the COUNTER_SIZE parameter. Use Auto / Single Clock defaults. Hit next and select the simplest configuration 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.

MegaWizard Plug-In Manager [page 5 of 5] ? X

ROM: 1-PORT

About Documentation

Parameter Settings EDA Summary



Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a green checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

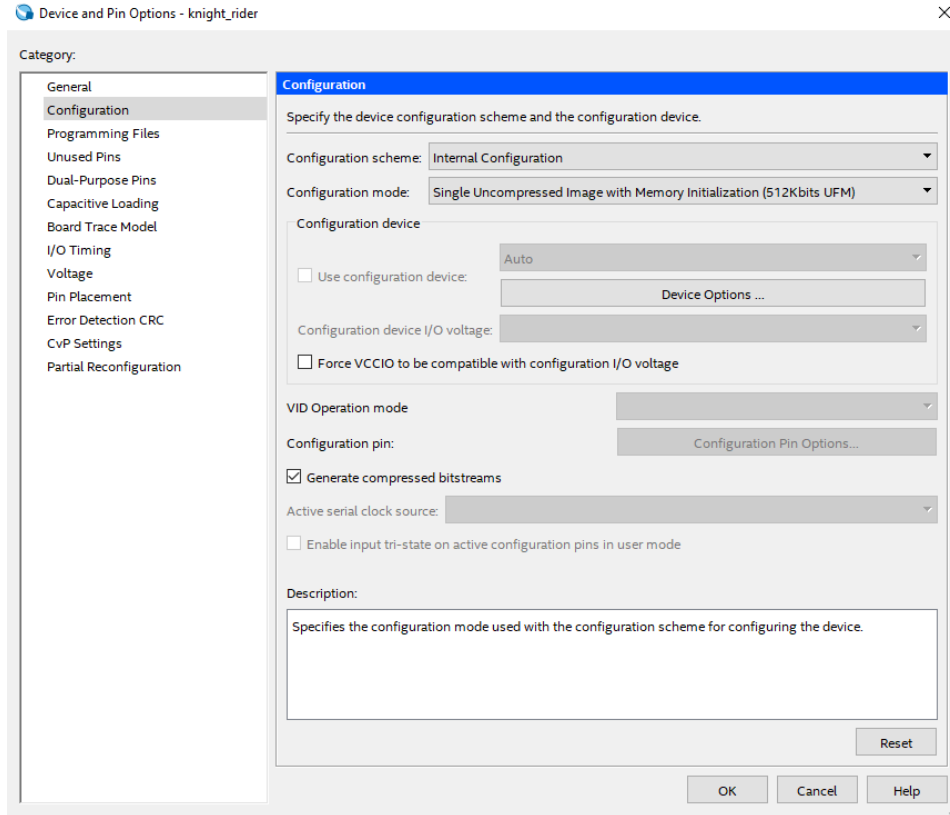
The MegaWizard Plug-In Manager creates the selected files in the following directory:

C:\Users\DanaS\Documents\Quartus Labs\Lab\

File	Description
<input checked="" type="checkbox"/> ROM.v	Variation file
<input type="checkbox"/> ROM.inc	AHDL Include file
<input type="checkbox"/> ROM.cmp	VHDL component declaration file
<input type="checkbox"/> ROM.bsf	Quartus Prime symbol file
<input checked="" type="checkbox"/> ROM_inst.v	Instantiation template file
<input checked="" type="checkbox"/> ROM_bb.v	Verilog HDL black-box file

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.

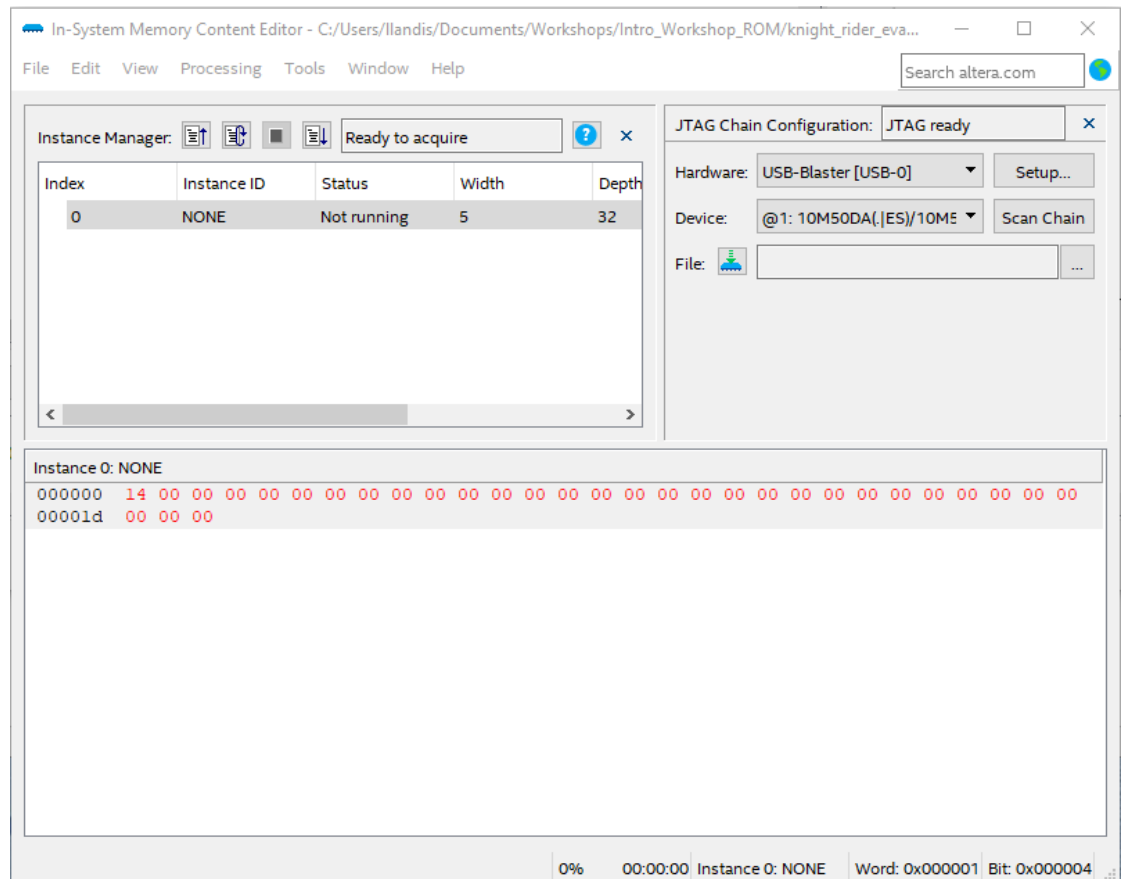


Now you will need to instantiate your ROM module into your knight_rider.v file. Find the ROM_inst.v file for hints on how to add this module to your design.

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

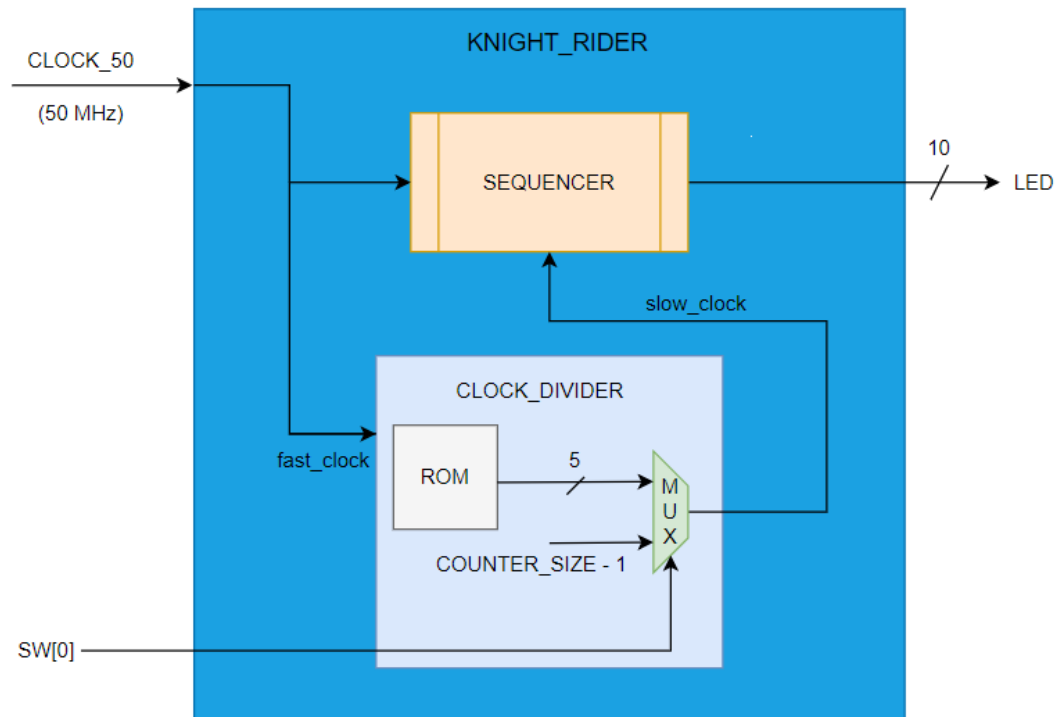
- The ROM module should be instantiated inside the clock divider module
- Change the COUNTER_SIZE to 32 - there should be a wire that can represent all 32 values (how many bits?)
- Use a MUX to select between a default value for COUNTER_SIZE (like the number you used in the knight_rider lab) and the 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 drives the display, so it is what must be assigned using the counter tap wire

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



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



4.0 Document Revision History

Date	Author	Comments
6/25/2021	RK	— Transferred guide to common Word template and added some hints