TEST YOUR KNOWLEDGE: KNIGHT RIDER ROM

To learn more, visit http://fpgauniversity.intel.com

About this Document

This is a short lab to test your knowledge in assembling a small FPGA project on the DE10-Lite remote development kit. This project assumes you have successfully completed the Knight Rider LED sequencing circuit from the Introductiuon to Quartus class (see: https://www.intel.com/content/www/us/en/programmable/support/training/course/ouwintro.html for recording and https://github.com/intel/FPGA-Devcloud/tree/key/main/HandsFree/Devkits/DE10-Lite/IntroQuartus for the remote lab running on the remote console DE10-Lite board. Next we will introduce you to 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 3-4 minutes recompiling your code and downloading the programming image (.sof file) to the remote board. We will investigate new means to tune the value of COUNTER_SIZE so that you can quickly change its value without recompiling your design.

Using this tutorial

To run this project you will ned the Quartus Prime Lite design software and MAX10 library on your PC. Refer to the Introduction to Quartus manual for download details.

Related Information

- For any questions, problems or concerns, please feel free to leave a *new issue* in our GitHub site: https://github.com/intel/FPGA-Devcloud/issues
- For more information: University Workshop: Introduction to FPGAs and the Intel Quartus Prime Software

ROM Editor

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 remote 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.

STEP 1: CREATE MEMORY INITIALIZATION FILE (MIF)

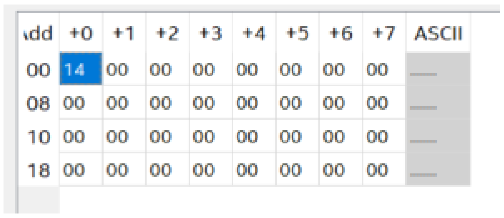
The quickest way to create a new MIF file:

File → New → Memory Initialization File. This will create a memory initialization file.

A window asking for "Numbers of Words and Word Size" will pop up. 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.

Save it and enter a file name to store the ROM contents (eg clock_divider_tap.mif).

You should see something like Figure 1 below.



ıdd	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
00	14	00	00	00	00	00	00	00	—
08	00	00	00	00	00	00	00	00	—
10	00	00	00	00	00	00	00	00	—
18	00	00	00	00	00	00	00	00	—

Figure 1: Example of MIF file

STEP 2: LAUNCH THE ROM EDITOR FROM THE IP CATALOG

Enter ROM in the search field of the IP Catalog. If you do not see the IP Catalog shown in Figure 2, go to View → Utility Windows → IP Catalog.

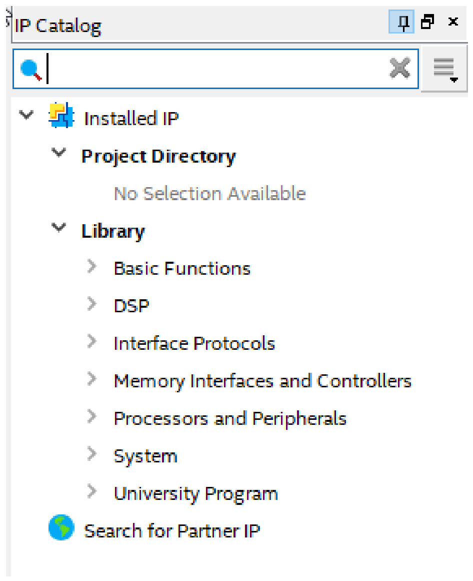


Figure 2: IP Catalog

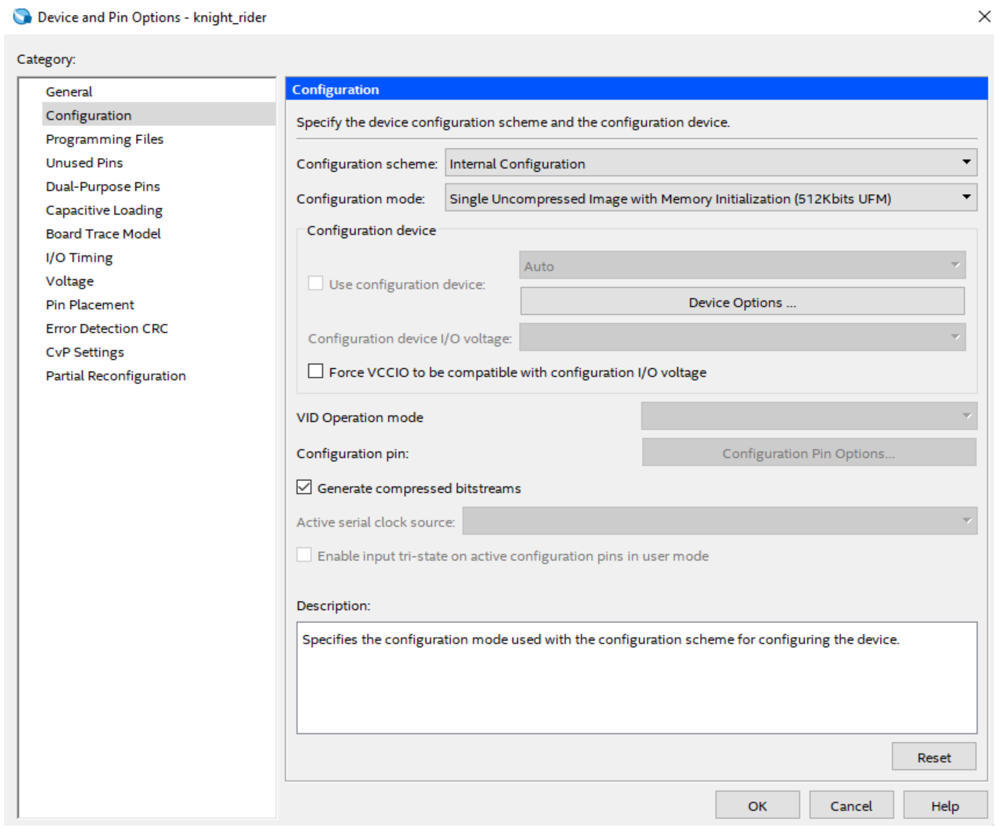
Select 1-Port ROM.
Make the name of the new IP block_ROM.
Edit the fields so you can make the smallest ROM possible to hold a single word that you can store the values of the COUNTER_SIZE parameter.
Use Auto / Single Clock defaults.
Hit next and select the simplest configuration that will meet your needs.
Hit next and enter the MIF file name that you used in Step 1.
Select Allow In-System Content Editor box – you will use this feature.
Hit next with defaults for the EDA tab to proceed to the 3rd summary tab.
In the last configuration panel select the checkbox Instantiation template file and hit finish.

STEP 3: INTERNAL CONFIGURATION WITH ERAM

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.


Device and Pin Options - knight_rider

Category:

Configuration

Specify the device configuration scheme and the configuration device.

Configuration scheme: Internal Configuration

Configuration mode: Single Uncompressed Image with Memory Initialization (512Kbits UFM)

Configuration device

☐ Use configuration device: Auto

Configuration device I/O voltage:

☐ Force VCCIO to be compatible with configuration I/O voltage

VID Operation mode

Configuration pin:

☑ Generate compressed bitstreams

Active serial clock source:

☐ Enable input tri-state on active configuration pins in user mode

Description:

Specifies the configuration mode used with the configuration scheme for configuring the device.

Figure 3: Device and Pin Options Configuration Window

If you do not see the option shown in Figure 3, make sure that the correct device is selected. For the DE-10 Lite development board, you should see device name MAX 10 10M50DAF484C7G highlighted.

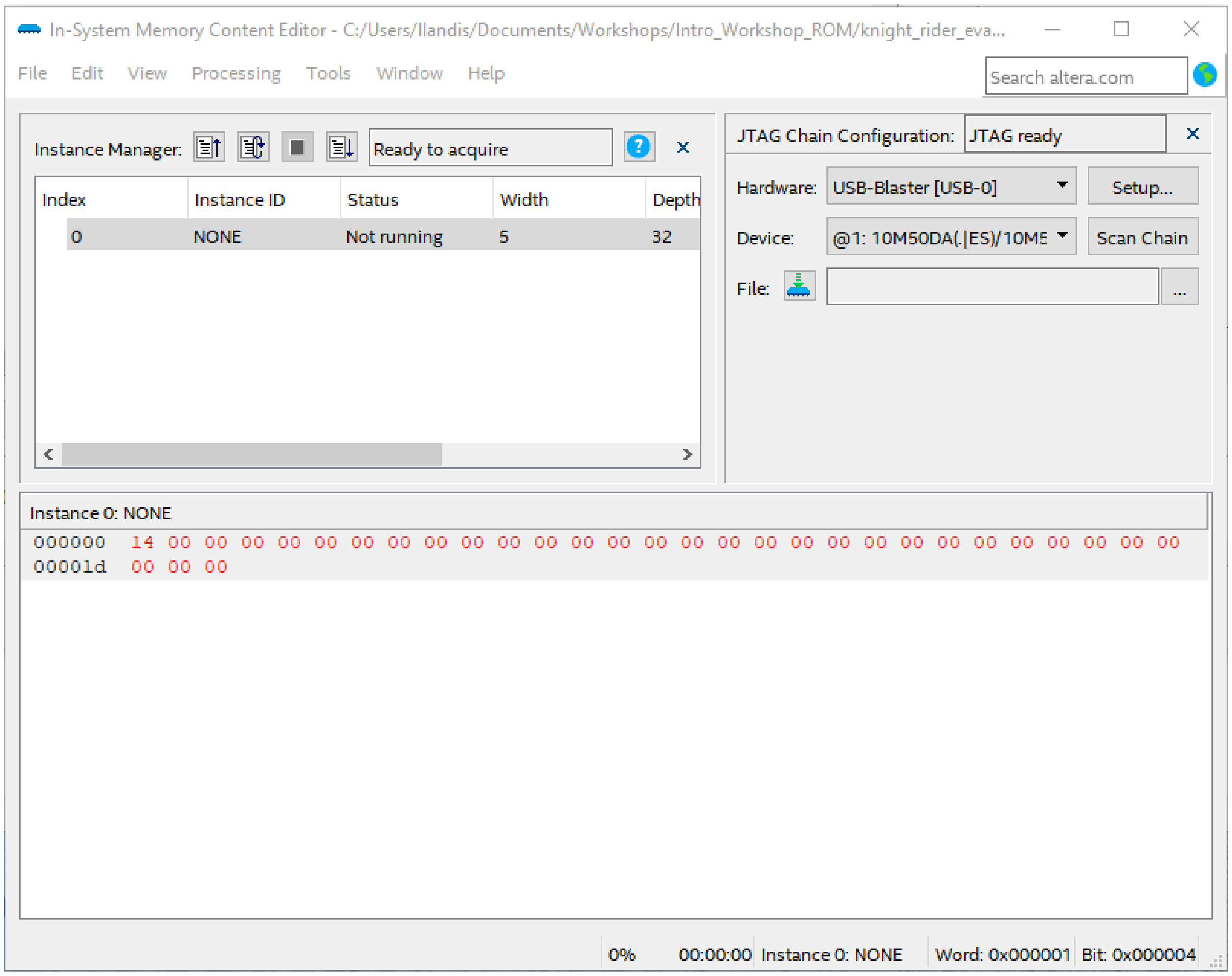
STEP 4: INSTANTIATE ROM MODULE

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.

You will also need to instantiate knight_rider into top.v or rely on your solution from the Introduction to Quartus training.

Compile, work out syntax errors and your DE10-Lite Remote Console Board should launch.

While your board is running, open Tools → In-System memory Editor.


In-System Memory Content Editor - C:/Users/llandis/Documents/Workshops/Intro_Workshop_ROM/knight_rider_eva...
File Edit View Processing Tools Window Help
Search altera.com
Instance Manager:
Ready to acquire
JTAG Chain Configuration: JTAG ready
Hardware: USB-Blaster [USB-0]
Setup...
Device: @1: 10M50DA(.|ES)/10M5
Scan Chain
File:

Index	Instance ID	Status	Width	Depth
0	NONE	Not running	5	32

```
Instance 0: NONE
000000  14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001d  00 00 00
```

```
0%    00:00:00 Instance 0: NONE    Word: 0x000001  Bit: 0x000004
```

Figure 4: In-System Memory Content Editor Window ROM Edit Example

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.

Congratulations! Knight Rider ROM is complete.

DOCUMENT REVISION HISTORY

List the revision history for the application note.

Name	Date	Changes
Samantha Banda	8/19/2020	Initial Release of guide