TEST YOUR KNOWLEDGE – KNIGHT RIDER ROTARY ENCODER

Last updated: July 9, 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](#_Toc76739331)

[1.1 Prerequisites 3](#_Toc76739332)

[1.2 Required Items 3](#_Toc76739333)

[1.3 Reference Documents 3](#_Toc76739334)

[2.0 Lab Background 4](#_Toc76739335)

[2.1.1 Finite State Machine Design 6](#_Toc76739336)

[3.0 Lab Steps 7](#_Toc76739337)

[3.1 Modules provided 7](#_Toc76739338)

[3.2 Step 1: Design and simulate the Rotary Encoder 7](#_Toc76739339)

[3.3 Step 2: Compile and download the knight\_rider module to your FPGA development board 7](#_Toc76739340)

[3.4 Assemble the top level design 8](#_Toc76739341)

[3.5 Connecting the rotary encoder to the FPGA board 8](#_Toc76739342)

[3.6 Compile the project and program the board 9](#_Toc76739343)

[4.0 Document Revision History 10](#_Toc76739344)

# Introduction

***Knight Rider Light Sequencer with Rotary Encoder Speed Control:***

**Now that you have completed the Knight Rider LED sequencing circuit on the DE10-Lite development kit, we will next introduce the use of a handy piece of hardware called a rotary encoder to control the speed of the LED light sequencer.**

## Prerequisites

* This lab assumes you have completed the following self-guided or instructor led lab: [OUWINTRO](https://www.intel.com/content/www/us/en/programmable/support/training/course/ouwintro.html)
* Basic knowledge of digital electronics including sequential logic and state machines
* Basic understanding of Verilog coding

## Required Items

* [Terasic DE10-Lite development kit](https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=1021)
* [Rotary Encoder](https://www.amazon.com/gp/product/B07QL6V4WP/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1)
* [Jumper wires](https://www.amazon.com/gp/product/B07GD2BWPY/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1)

## Reference Documents

Table 1‑2. Reference Documents

|  |  |
| --- | --- |
| Document | Document No./Location |
| .zip file containing three Verilog modules | [KnightRider\_RotaryEncoder.zip](https://github.com/intel/FPGA-Devcloud/blob/master/main/QuickStartGuides/TestYourKnowledge/TestYourKnowledge_KnightRiderRotaryEncoder/KnightRider_RotaryEncoder.zip) |

# Lab Background

If you worked on the knight\_rider project you might have spent a few iterations to derive the parameter COUNTER\_SIZE value 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 5-10 minutes recompiling your code and downloading the programming image (.sof file) to the DE10-Lite kit. We will investigate a new means to tune the value of clock speed using a rotary encoder switch so that you can quickly change it’s value without recompiling your design.

Often when working with designs that require some type of position or rotation sensing, an electro-mechanical device called an *encoder* is used. It may either encode linear position (linear encoder) or rotation (rotary encoder). In this lab you will be introduced to the *rotary* *encoder*: a device where using the proper electrical interface you can determine the rotation angle of a shaft. A few example applications are: a) an interface between a motor and a control circuit to provide information to control the speed of the motor; b) a volume or tuner control on a digital audio system.  
The goal of this design is to use the encoder signals as inputs to a counter. We will count up or down based on the direction of the rotation. A hex display will show the magnitude of the counter output which will in turn determine the speed of the Knight Rider LED sequencer.



Figure 1: Rotary Encoder

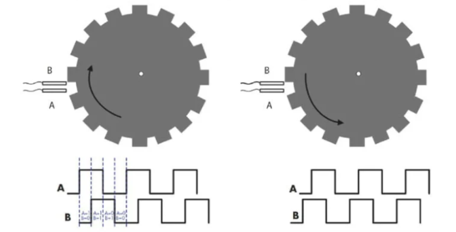


Figure 2: Rotary encoder theory of operation

The operation of a mechanical rotary encoder is achieved with two contacts placed in above a rotating conductive wheel such that signals A and B produce 90 degree offset square waves. Signal A leads or lags B based on the direction of rotation. By designing a finite state machine (FSM) we can generate signals count enable, rotation direction, and a third on/off signal when the button is pressed. We will use this FSM to generate master clock 50MHz divided frequencies for control of the Knight Rider light sequencer. Rotary encoders are available from a variety of manufacturers and number of teeth which indicate the number of clicks to complete 1 revolution.

### Finite State Machine Design

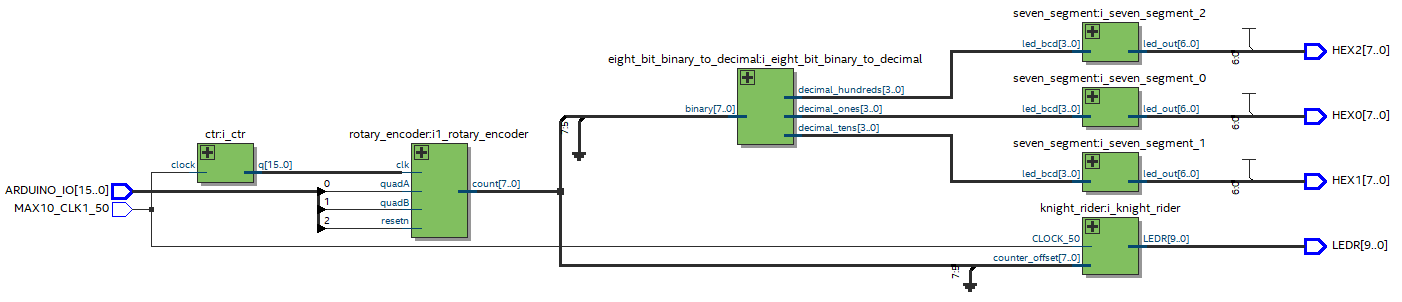
Diagram

Description automatically generated

Figure 3: The rotary encoder state machine transitions are controlled by inputs A and B encoded as signal quadctl[1:0]

#### Block Diagram

\*Implementation details might vary



# Lab Steps

## Modules provided

**knight\_rider**: Light sequencer function. Includes the clock\_divider module to determine sequencing rate

**rotary\_encoder**: This is what you have to build for this lab (not included). Create a new Verilog file for this.

**eight\_bit\_binary\_to\_decimal**: converts binary to ones/tens/hundreds digits

**seven\_segment**: Takes ones/tens/hundreds digits and decodes seven segment display values

\*Note: These are separate Verilog files that must be added to your own project

## Step 1: Design and simulate the Rotary Encoder

It’s always a good idea to simulate state machines prior to assembling the top level design. Refer to Figure 2 for the waveforms of inputs A and B. In theory, the clock can be considerably faster than the pulses from the rotary encoder. It is assumed you know how to either use the University Waveform View application within Quartus, or write a testbench to stimulate the rotary encoder with the A and B waveforms offset by 90 degrees.

Helpful hint: use the “Insert Template” feature in Quartus: File > New > Verilog HDL File. Inside the file editor: Right Click > Insert Template > Verilog HDL > Full Designs > State Machines. This will provide you with a Verilog template of either a Moore or Mealy state machine.

## Step 2: Compile and download the knight\_rider module to your FPGA development board

In Quartus, set the project navigator tab to files if not already in this state. Right click on knight\_rider.v and set to top level entity. Compile the design and confirm there are no errors. Open the programmer and select rotary\_encoder\_project.sof and download to the DE10-Lite FPGA development board. Observe the LEDs sequencing at the appropriate frequency.

## Assemble the top level design

Connect the various modules in the top level and ensure that the project compiles properly with the appropriate pin locations based on your FPGA development board. Refer to the following section on connecting the rotary encoder and the pin locations for that.

## Connecting the rotary encoder to the FPGA board

The Rotary Encoder has 5 terminals: Power, ground, A, B and Switch. Each manufacturer might label these slightly differently. For the Tegg 2 PCS KY-040 360 Degree Rotary Encoder Module Brick Sensor clickable Switch with Knob Cap for Arduino that is listed in the link in section 1, the mapping is shown below.

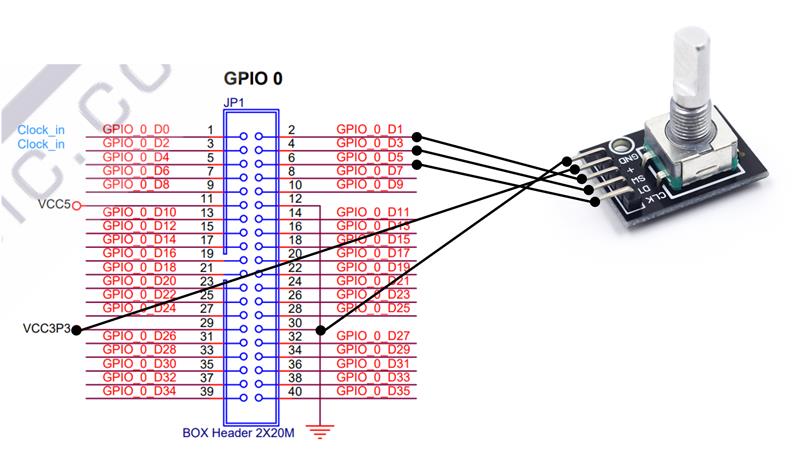
A: DT

B: CLK

Button: SW

Power: +

Ground: GND



\*These connections are for the DE1-SoC FPGA kit. Find the schematics from the Terasic website.

To make the appropriate connections to the Arduino header you will need to refer to the schematic of the DE10-Lite. This can be downloaded from this [link](https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=218&No=1021&PartNo=4). Look for the DE10-Lite CD-ROM manual and download that file. Unzip the file and under the Schematic folder, open up the file called de10-lite.pdf. On page 13, you will observe the Arduino connections. Use 5 male to female jumper wires to make the connections. Connect the CLK pin to Arduino\_IO0, DT pin to Arduino\_IO1, SW pin to Arduino\_IO2, Power to an appropriate VCC5 pin and GND to a ground socket on the Arduino header.

## Compile the project and program the board

Make sure that your top level entity is set to the appropriate top level file and compile your design. Once the jumper wires are connected to the rotary encoder, program the .sof file to the DE10-Lite kit and experiment with the operation. Make sure that the operation shows hex values 1 through 8 and the sequencing rate of the LEDs progressively gets faster with the higher rate. Turning the rotary encoder clockwise should speed up the sequencing rate and counter-clockwise should reduce the sequencing rate. Pushing the button should reset the counter back to 1. Make appropriate changes to your code until you get the appropriate behavior you are seeking.

# Document Revision History

|  |  |  |
| --- | --- | --- |
| Date | Author | Changes |
| 7/9/2021 | RK | * Initial release, created new document using official template and included new sections |
|  |  |  |