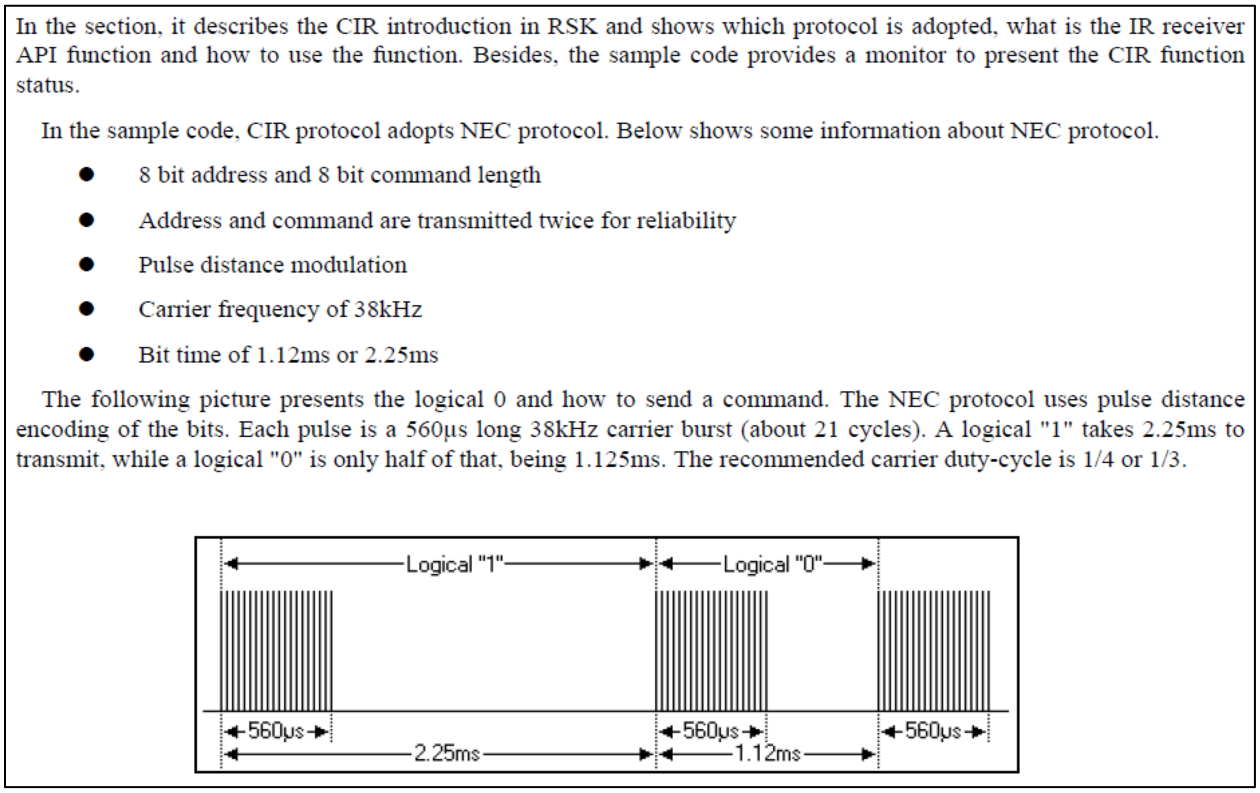
Lab 8 – Pulse Distance Coding Part 1, Designing a Datapath Component

Introduction

Pulse Distance Coding is a technique to encode zeros and ones into a serial stream by differentiating the distance between pulses. A common application is the transmission of Consumer Infrared (CIR) remote control codes. The protocol you will use in this lab is one that is used by NEC Corporation in their IR remote control devices. The following figure is copied from their application note on R8C/27.

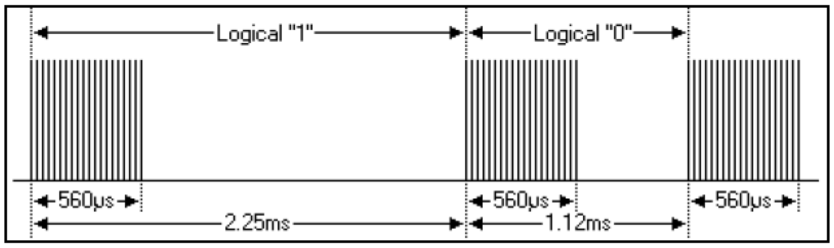


In the section, it describes the CIR introduction in RSK and shows which protocol is adopted, what is the IR receiver API function and how to use the function. Besides, the sample code provides a monitor to present the CIR function status.

In the sample code, CIR protocol adopts NEC protocol. Below shows some information about NEC protocol.

- 8 bit address and 8 bit command length
- Address and command are transmitted twice for reliability
- Pulse distance modulation
- Carrier frequency of 38kHz
- Bit time of 1.12ms or 2.25ms

The following picture presents the logical 0 and how to send a command. The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560μs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.



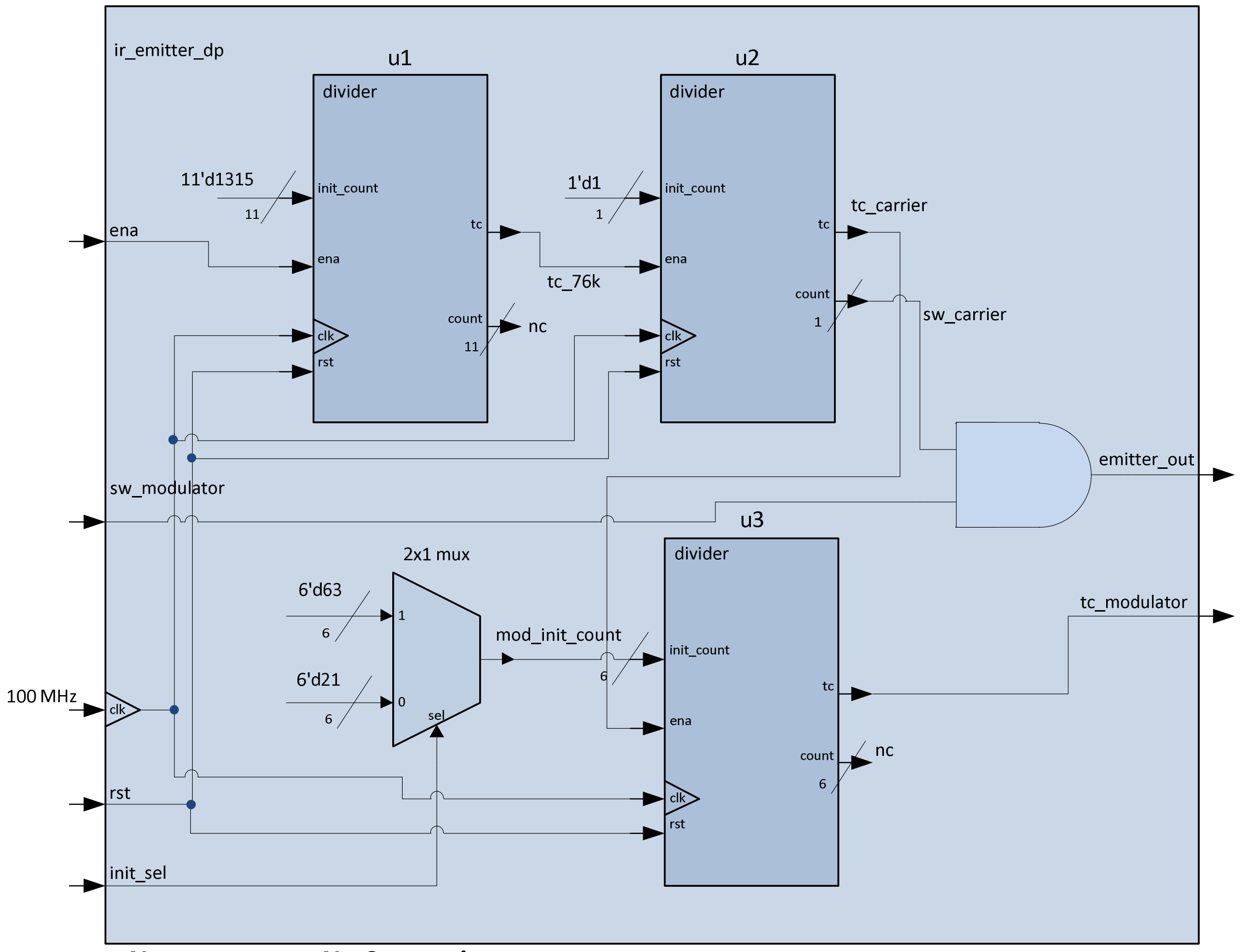
You will be creating the ***datapath*** piece of a design that will generate emitter signals using this coding technique. In a future lab you will be designing a ***controller*** that will provide the control signals for this block. The input control signals to the datapath will determine whether to transmit a zero or a one for each bit.

You will be reusing the parameterized divider module you designed in Lab 7.

Detailed Specification:

A block diagram for the main data path module, ***ir_emitter_dp***, is illustrated below. This design is very similar to the ***ir_emitter*** in Lab 7 with a few exceptions: a) a 2x1 multiplexer is used to select between two different modulation counts, decimal 21 or 63; b) the ***sw_modulator*** signal will now be an input, coming from an external controller, rather than the internal ***divider*** module (instance ***u4*** in Lab 7). These changes allow us to select between two different modulation times (counts), which are used in the NEC protocol on the previous page.

In a following lab the controller will supply four control signals to this datapath: ***ena***, ***sw_modulator***, ***init_sel***, and ***rst***. The output signal, ***tc_modulator***, will be an input to the controller, signaling it when to change states. The ***emitter_out*** signal will eventually drive an IR emitter, similar to Lab 7.



Note: nc means No Connection

Design

A skeleton file is provided for the datapath module, ***ir_emitter_dp***. This is the only module that you need to design.

Complete the design for the IR emitter datapath, ***ir_emitter_dp.sv***, based on the above specification (block diagram). I ***do not*** recommend that you copy your Lab 7 ***ir_emitter*** module and edit from it. There are too many places to make errors.

This is a *simulation only* design lab. You will not be implementing this in hardware until a following lab. Be sure the simulation works, otherwise you will be debugging this design in a following lab.

Simulation/Verification

In the simulation view, there is one test bench wrapper for the above module: ***tb_ir_emitter_dp.sv***. This file should not be edited.

Verification step:

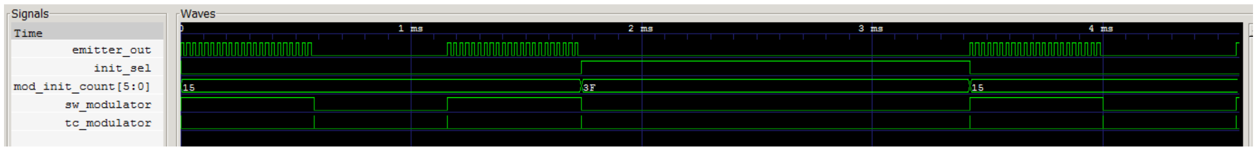
- ***tb_ir_emitter_dp.sim*** is used to simulate the *ir_emitter_dp* module. This is not a self checking test bench.

You should look carefully at the timing simulation and observe that the emitter signal is being generated and modulated. The testbench monitors and measures a few of the timing characteristics of the two output signals: ***emitter_out*** and ***tc_modulator***. If either of these signals is inactive, the testbench will end with a timeout message. A successful design will have the following output (open the ***tp_ir_emitter_dp.log*** file):

```
# run 4600us
Carrier frequency 37.993921 kHz
Next init_sel: 0
Modulation on time: 0.579030 msec
Next init_sel: 0
Modulation off time: 0.579040 msec
Next init_sel: 0
Modulation on time: 0.579040 msec
Next init_sel: 1
Modulation off time: 1.684480 msec
Next init_sel: 0
Modulation on time: 0.579040 msec
Next init_sel: 0
Modulation off time: 0.579040 msec
Next init_sel: 0
Simulation complete!!!
```

In the output above, a measurement of the carrier frequency is displayed for one cycle of ***emitter_out***. Then the times between the ***tc_modulator*** pulses are displayed for a few conditions of the input control signals that are used to generate *pulse distance coding*. Note that when ***init_sel*** changes from *0 to 1*, the modulation time increases as expected – as the ***mod_init_count*** signal to the modulation counter changes.

Below is a screen shot of the timing diagram of the simulation. Note the modulation on/off times correspond to the timing sequence in the simulation output text output above, as well as the behavior of the input control signals described in the previous paragraph.



This is not a full test bench. It is only testing a few operational possibilities that will validate your design, as the ***divider*** module you are using has already been verified in Lab 7.

Study carefully the timing diagram and refer to it in Lab 10, particularly the input and output signals. These signals will be used in that lab where you will be designing a controller that drives these signals creating the NEC pulse distance protocol.

Implementation

This is a *simulation only* lab. You will implement this design in hardware after you complete and integrate it with a controller in Lab 10.