RFID Duplicator

1st Kyle Heinz
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA
kyleh51@mit.edu

2nd Walter Truitt
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA
wtruitt@mit.edu

3rd Christina Crow
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA
ccrow@mit.edu

***Abstract*—We present a design for a Radio Frequency Identification (RFID) duplicator, a system that can both intercept and transmit near-field RFID signals encoded with Binary Phase-shift Keying (BPSK) in the 125kHz band. Particularly, this will be used with MIT ID cards. Implemented using an FPGA as well as analog receive and transmit circuits, the system outputs a 125kHz carrier signal which is picked up by the coil in a passive tag and then BPSK-encoded. This signal is then read by the receive antenna, stored in memory, and decoded into a series of bits. Compared with known MIT ID bit sequences, the user-identifying bits are isolated and stored. The system has the option to put that ID on a programmable active tag. Writing to an active tag involves an in-depth protocol that is determined by the manufacturer that requires extra modules and FSMs to communicate with these tags. Once programmed, these tags can output the same waveform of the MIT ID cards.**

Keywords—Radio Frequency Identification, Binary Phase-shift Keying, Field-programmable gate array, Near-field

I. Physical System - Kyle

The system consists of an Urbana FPGA from AMD with:

- A receive antenna circuit
- A transmit antenna circuit
- Both stages involve a wire-loop antenna about 7.5cm in diameter with 16 turns of 22 AWG copper wire.

A. Receive Circuitry

In the receive stage, this antenna connects to a notch filter, limiting the effect of the 125kHz signal and making it easier to distinguish between peaks. This signal is inverted and offset before being fed into an AD7476-A 12-bit ADC which inputs into the FPGA.

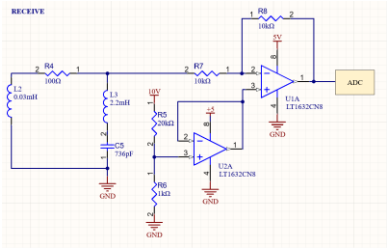


Fig. 1 – Receive circuitry

B. Transmit Circuitry

In the transmit stage, the DAC, an 8-bit PMOD R2R, outputs a signal at 2.5Vpp and 0.05A into a common drain current gain stage, resulting in a signal out at 1.5Vpp and 0.25A and a 3X power gain. With this magnification, more power is supplied to the circuitry within the ID card, allowing it to produce a larger, more detectable output signal.

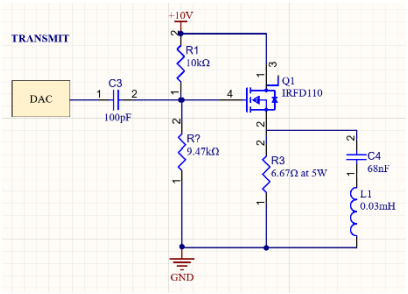


Fig. 2 – Transmit circuitry

II. Receiving

The received signal seen by the FPGA is a digitized combination of the 125kHz carrier frequency and the BPSK signal from the ID card as shown below.

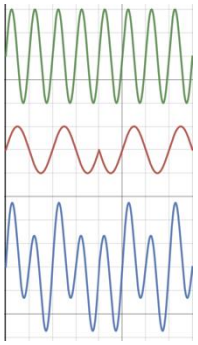


Fig. 3 – 125kHz carrier signal (green), BPSK signal (red), received signal (blue)

Bits are conveyed 32 peaks at a time, and every 32 peaks, a bit is recorded. When two adjacent peaks have roughly the same value (as determined by a tolerance threshold), the bit flips. In observed results, the receive network takes in an approximately

850mVpp analog signal with around 50mV of clearance between high and low peaks.

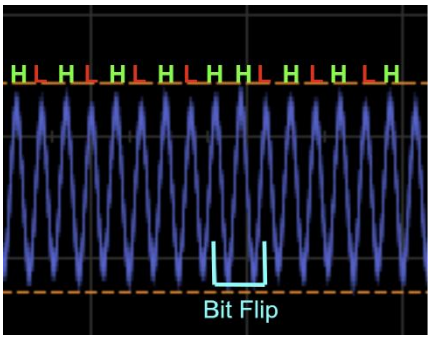


Fig. 4 – Observed receive signal with highlighted bit flip

An MIT ID sequence consists of 224 bits with a majority of fixed bits and a select few variable user bits. Using the Manta Integrated Logic Analyzer, we were able to determine that the signal starts with 30 zeroes, followed by a sequence of 20 constant bits, and 32 of the unique MIT ID bits.

III. Decoding - Christina

The digitized analog signal is passed through the decode system where it is converted into a bit stream of potential ID bits and compared against the expected MIT ID signature. If it matches, the green light will be emitted, and the 224 bits can be stored into memory at the location indicated by switches [15:13].

The first module of the decode system is a peak finder that tracks the input ADC values and outputs both a peak detected signal and the magnitude of that peak. A peak is found by storing four values at a time: the current value and past three values. A value is determined to be a peak if it is greater than both its previous and following values or equal to the previous and greater than the two previous values.

The second decode module takes as inputs the peak detected signal and the peak magnitude. As seen in figure 4, a bit flip is indicated by two low or two high peaks consecutively. This module outputs a bit flip signal when one of these characteristics is detected.

Lastly, the bit flip signal and peak detected signal are utilized to create the full 224-bit ID signature. This module detects the beginning of the MIT ID signature using the 30 leading zeros, during which no bit flip should be detected. Following the leading zeros, each bit is signaled by 16 62.5 kHz BPSK cycles, and therefore a new bit flip signal should be detected within approximately 31 peaks. Once enough bits are collected, they are compared to the expected MIT-unique bits to determine a valid signal. If it is a valid MIT signal, all 224 bits are collected, and the user has the option to store to memory or discard them.

IV. Transmitting - Walter

A. *Directly outputting a signal*

When directly outputting a specific ID signal, we select the spoofed 125kHz BPSK waveform which is generated by a spoofer module. The spoofer module works by alternating between a high and low version of the 125kHz carrier wave. When a bitflip occurs, this module repeats the same peak it is on then continues to alternate for at least 31 peaks of the carrier signal. This spoofed signal can be used by setting sw[2] on the FPGA to high and viewed by an oscilloscope probe.

B. *Writing to an active card*

To communicate with an active 125kHz RFID tag, (in our case the ATMEL T5577) there is a specific communication protocol called downlink, that must be followed and implemented on our FPGA. To communicate with this downlink protocol, the standard 125kHz sine wave signal is turned on and off at regular intervals to send packets. This packet structure, shown in Fig. 5, consists of an opcode + lockbit + 4 bytes of data + data address. In order to send the 224-bit data to the chip, this process must be repeated 7 times.

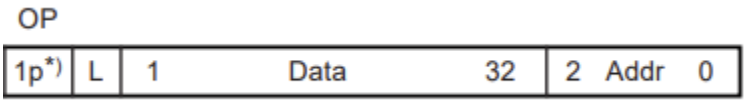


Fig. 5 – Downlink Packet Structure

The protocol also determines the length of time in which the 125kHz signal must be on before being turned off to send 1's and 0's to the chip. Per the ATA5577 datasheet [1], write mode is initiated when the signal is turned off for at least 8 periods and no more than 50 periods. After write mode is enabled, the signal is interrupted by a gap of length 8-20 cycles. To send a 1, the time between gaps is 56 cycles. To send a 0 the time between gaps is 24 cycles. After the data packet has been sent, the chip enters a 5.6 millisecond programming cycle. This timing description is shown in Fig. 6 for better understanding, where S_{gap} represents the length of gap needed to initiate write mode and W_{gap} represents the length of gap used to send data.

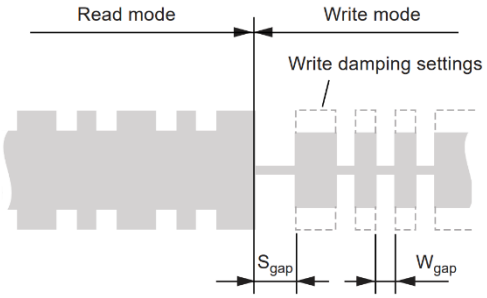


Fig. 6 – Downlink Protocol Example

Because of the strict timing nature of the protocol, we have chosen to implement it with a finite state machine. This FSM is specifically built to send MIT ID numbers to the ATA5577 by taking in the 224-bit wide RFID data selected, separating the data into 7 blocks of 32 bits, applying metadata such as page and block addresses, and finally sending these blocks one at a time to the chip by toggling a mux connected to the DAC and transmit circuits. This is shown below in Fig.7.

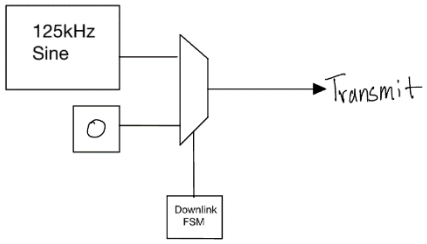
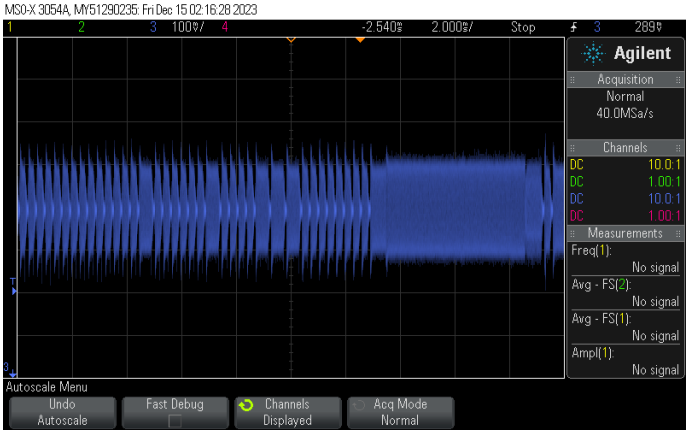


Fig. 7 – Downlink FSM Function

After programming the active RFID tags with the 224 bits that are selected by the user, the tag still does not behave in a way that we want. The default configuration for these specific tags is a data bit rate of RF/8 or 125/8kHz and a modulation scheme called direct modulation. To change these we write configuration data to the ATA5577's page 0 block 0 address. This requires a separate configure module that is accessed by putting sw[1] and sw[0] both high. By writing the string:

0000_0000_0000_1000_0001_0000_1110_0000

We are able to change this tag to behave with a data bit rate of 125/32kHz (which is what we expect from the MIT cards), a modulation scheme of BPSK, and we also tell it to read all the way to the end of its memory block (block 7). The figure below shows the oscilloscope output of this writing scheme for better understanding. In the image you will see multiple pulses of the 125kHz sine wave – a short pulse indicating a zero and a long pulse indicating a one. This is followed by a programming delay and then by a short reset code to send the card into its regular read mode. All of this functionality was determined by studying the ATA5577C datasheet.



V. Graphical Interface - Christina

The graphical interface for this project consists of a display of stored RFID signatures and their respective BPSK waveforms. Specifically, the 22 MIT specific and 32 user specific bits of each MIT ID stored in memory are displayed in rows on the upper left hand of the screen. Since the decoder module stores up to eight unique ID signatures to memory, up to eight can be displayed to the screen. The right half of the screen will contain a compressed BPSK signal representing the first 32 bits of one of the stored IDs. The stored signal to be displayed in its wave form is determined by the user through switches 13-15 on the FPGA. The selected ID binary sequence is highlighted on the display for convenience. The figure below details the HDMI signal generation modules as a block diagram.

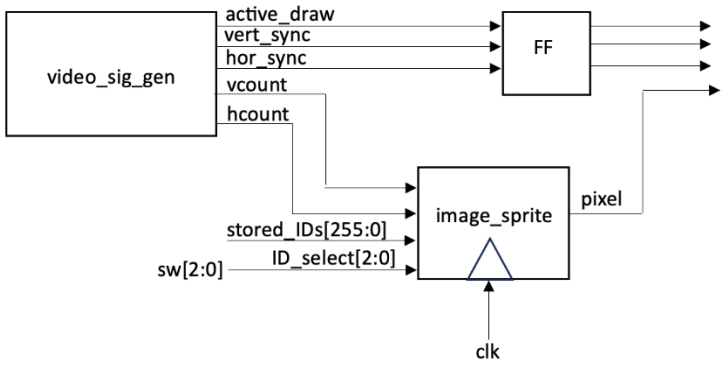


Fig. 8 – Graphical Interface Block Diagram

The image_sprite module contains the logic necessary to produce the correct RGB value for each individual pixel. It receives inputs of the vertical and horizontal index of the current pixel, the 32-bit unique codes from all eight stored RFID signatures, and the ID_select value from switches on the FPGA. It outputs RGB values as required by the HDMI protocol. Since the only graphics desired on the screen are the text (both white text on black background and highlighted, black text on white background) and waveform representations of zero and one, these pixel patterns are standardized as six separate templates stored as .mem files in BRAM accessed by image_sprite. The display will be in black and white, allowing the templates to hold single binary values to encode the RGB of each pixel. To use these templates, the screen is separated into regions by constant indices that represent the spaces for a single template. When image_sprite recognizes that the current index is within a space, it determines whether the value displayed should be a one or a zero, waveform or binary value, and highlighted or unhighlighted from stored_IDs and ID_select. It then requests and outputs the corresponding pixel's black/white value from the correct template.

The video_sig_gen module is the same as that written for 6.205 Lab 4. It produces the necessary signals and horizontal and vertical index values to generate HDMI video. A flip-flop will be added to pipeline the circuit, stalling it by two clock cycles to ensure that the pixel output of image_sprite will be sent out with the correct additional signal generating signals.

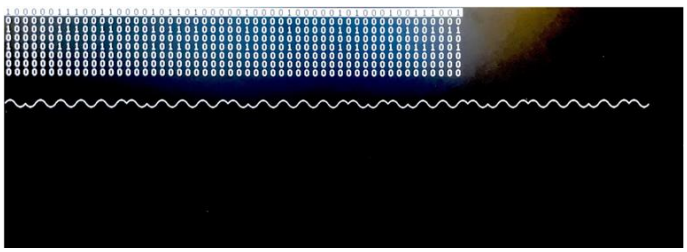


Fig. 9 – Graphical Interface Display

VI. Conclusion - Kyle

Our system allowed for the reading and copying of RFID signatures at close touch proximity. The card was required to be directly on our antenna, so a future design improvement could focus on extending the range of the transmit antenna, allowing more energy to flow into the card and to create a stronger signal for the receive circuitry.

As we focused on reading the peaks of the incoming signal, we highly recommend to future builders to either have easily distinguishable peaks or a high sampling rate and bit depth. Initially, we struggled to distinguish between peaks, forcing us to update our analog circuitry to dampen the 125kHz carrier frequency, making high and low peaks more easily distinguishable.

While peak finding was difficult, it helped remove extra electrical components from the design, allowing us to rely more on the FPGA and to avoid spending money on unnecessary parts.

With regards to timing, our system had to sample and decode the input signal fast enough to avoid missing any bits and to minimize the amount of time the ID needed to spend on the card reader. The main bottleneck was with the ADC which serialized each sample and sent it in on a 20MHz clock. The rest of the system ran on a 100MHz clock except for the graphics display which required 74.25 MHz and 371.25 MHz clocks for the HDMI connection.

Limited amounts of BRAM were used for storing image sprite files in the graphics display. The rest of the system was purely logic, so not a lot of storage was needed, even for the 8 identification signatures which were 224 bits.

Ultimately, the system proved successful at reading from tags, writing to active tags, and storing and displaying the user information. Thus, all goals set at the start of the project were met. While the system worked for the 125kHz signals, we soon discovered that MIT had updated their system to the HID iCLASS, an RFID network that operates in the 13.56MHz range. As our ADC and DAC were much slower than that frequency, we were unable to adjust our design to operate in that range. Future groups approaching the issue will have to account for the need for this higher sampling rate. Although we were not able to open any doors on campus, we demonstrated a proof of concept that can be replicated and adjusted to work on any system operating in the 125kHz range. All it would take is a scope on a card being stimulated to determine that systems' constant bits which can be swapped for the known MIT ID ones.

VII. Contributions - Kyle

This project was the result of a strong collaborative effort as well as a lot of research into different domains like radio frequency circuitry, RFID, and digital design. Kyle built the analog circuitry and worked on the ADC decoding as well as the peak finding and sequence decoding modules. Walter worked on the ADC decoding, the DAC output, the active tag protocol, and the sequence decoding module. Christina worked on the graphics display, the bit flip detector module, and the sequence decoding module. All three of us spent time debugging and integrating our modules into the top-level file. We are extremely grateful to Joe Steinmeyer, the instructor of the course, for helping with debugging as well as all of the other TAs who helped along the way. In addition, Dave Lewis played a huge role in sourcing parts for our design, and we appreciate his effort.

References

[1] Microchip, "ATA5577C – Read/Write LF RFID IDIC 100 kHz to 150 kHz,"ATA5577C Datasheet, Feb. 2020

[2] Github Code Repository: https://github.mit.edu/kyleh51/6.2050-RFID-Duplicator.git