ADVANCES IN A LOW-COST SOFTWARE-DEFINED TELEMETRY SYSTEM

Michael Don and Mark Ilg
U.S. Army Research Laboratory
Aberdeen Proving Grounds, MD, 21005
michael.l.don2.civ@mail.mil
mark.d.ilg.civ@mail.mil

ABSTRACT

The United States Army Research Laboratory (ARL) has developed a Pulse Code Modulated (PCM)/Binary Frequency Shift Key (BFSK) telemetry system using a low-cost commercial Software Defined Radio (SDR). Whereas traditional radio systems are implemented in hardware, much of the functionality of SDR is defined in software. This gives them the flexibility to accommodate military telemetry standards as well as other specialized functions. After a summary of previous development, new telemetry advances are described, focusing on telemetry transmitter, Advanced Encryption Standard (AES) encryption, and layered protocol design. Many of these functions were proven in a field experiment at Yuma Proving Ground, AZ in March of 2017 where data was successfully encrypted, transmitted, decoded, and processed in real-time.

INTRODUCTION

Traditional military radio systems implemented much of the processing in dedicated hardware equipment whereas much of the functionality of SDR is defined in software. This makes them a highly versatile platform that can support military telemetry standards as well as research into new telemetry schemes. As technology advances, SDR will become faster, smaller, more power efficient, and less expensive. Already, system-on-chip technology is incorporating multiple SDR components into a single Integrated Cirtuit (IC). This makes SDR not only a platform for receivers, but also a practical option for higher volume telemetry transmitters. Due to the versatile nature of SDR, telemetry research can be adapted to future munitions applications to include sensing [1], cognitive networks [2], and robust communications.

ARL has been utilizing telemetry links since the the mid 1990's to evaluate the aerodynamic flight behavior and diagnostic analysis of ballistic munition systems. Early methods used FPGAs to sample a small set of analog sensors, frame the data, and broadcast in the S-Band of the wireless spectrum using PCM over a BFSK channel [3]. A complex set of ground support equipment including antennas, receivers, bit-synchronizers, recorders, and data visualization systems provided the collection capability to receive the data from the systems under test. To aid in the development of

smart guided weapons new telemetry systems have been developed to support interfaces including Global Position System (GPS) receivers and digital Micro Electro Mechanical Sensors (MEMSs) which have the ability to survive the harsh conditions of gun-launch. These systems relied on a Field Programmable Gate Array (FPGA) to frame the sensor data and comply with Inter-Range Instrumentation Group (IRIG) standards.

Using a FPGA for the framing and data processing were often time consuming and challenging due to the strict timing requirements and programming model. There are three possible solutions to this problem. The first is to implement a soft processor in the FPGA fabric which allows for higher level programming abstraction [4]. The disavantage of this method is that soft processors utilize valuable FPGA resources to process simple serial streams (ex. GPS receivers) or digital inertial sensors. Alternatively, tools such as Xilinx High-Level Synthesis tool can convert C/C++ to Register Transfer Level (RTL) [5], however FPGA high-level synthesis is not as straight forward as real microprocessor programming. The third method is for the FPGA to offload the protocol layer to a co-processor. In this paper, the later approach is taken, using a Digital Signal Processor (DSP) as the co-processor to ensure the most flexibility and longevity of the design.

In many munitions applications Size, Weight, and Power (SWAP) is an issue with any electronics system. Therefore, the problem is scoped into two classes, a SDR only solution and a hybrid approach of using SDR as a base-station and a custom transmitter solution embedded into the system under test. A full SDR solution would provide additional capability to include bidirectional radio links and implementation of commercial standards [6]. The hybrid solution allows for a more transitional adoption and integration with legacy proven and validated Radio Frequency (RF) front-ends. Focusing on a hybrid solution allows for laboratory evaluations of systems under test to transmit to a SDR base station which does not require costly equipment and allows for flexible data analysis tools. Whereas, during range testing, traditional telemetry equipment may be used for the RF reception and interface to the SDR back-end. For example, using range tracking antennas, RF receivers, and bit synchronization equipment and bypassing the radio portion of the SDR hardware.

In this paper we will describe the design of several components of a hybrid solution to transmit encrypted data from a guided munition. First, previous SDR telemetry development is summarized. Next, work on a SDR telemetry transmitter is presented, followed by an overview of an AES encryption scheme. Finally, the co-processor implementation of the protocol layers and processing pipeline are described.

PREVIOUS WORK

Previously, we presented initial research developing a SDR telemetry receiver suitable for our current Frequency Modulation (FM)/PCM Lower S-Band (2200-2290 MHz) [7] transmitters, which employ BFSK modulation [8]. Two SDR receiver implementations were presented. The first used a SDR to acquire Inphase/Quadrature (I/Q) data and a LabVIEW program on a host computer to implement the BFSK receiver. This design worked successfully, but was too slow for typical ARL applications. The second design moved the radio algorithms to the SDR's onboard FPGA.

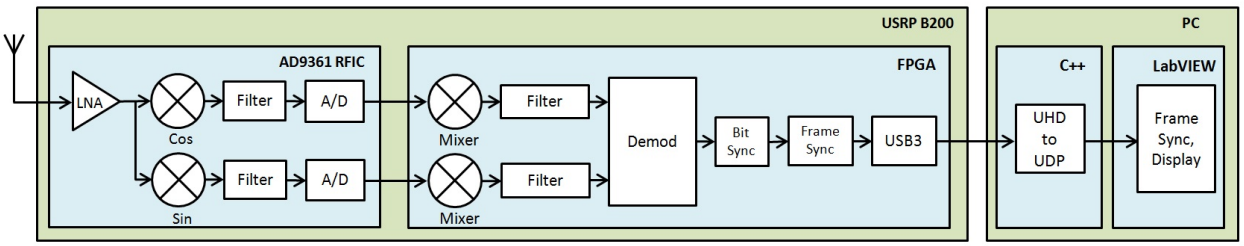

Figure 1: SDR telemetry receiver diagram.

This resulted in increased performance supporting data rates up to 10 Megabits per second (Mbs). Figure 1 shows a diagram of the SDR telemetry receiver. The Universal Software Radio Peripheral (USRP)'s Radio Frequency Integrated Circuit (RFIC) implements an I/Q demodulator, providing IF I/Q data to the FPGA. Ordinarily, the FPGA would downsample and convert the data to baseband before passing it to a Personal Computer (PC) for further analysis. In order to accelerate processing; demodulation, bit synchronization, and frame synchronization were added to the FPGA. Frame data was then passed to a PC where an intermediate C++ program received the data and sent it via User Datagram Protocol (UDP) to a LabVIEW program for display and archiving. This basic receiver architecture is ideally suited to form a basis for other custom applications.

SDR TELEMETRY TRANSMITTER

Several enhancements have been added to the SDR telemetry system including transmission capability, AES decryption, GPS time-stamping, spectrum monitoring, and an improved graphical user interface. An account of all of these improvements is out of the scope of this paper, but an overview of the AES decryption and SDR transmitter are presented here.

The USRP uses I/Q modulation for transmission given by

$$s(t) = I(t)\cos(2\pi f_c t) - Q(t)\sin(2\pi f_c t), \tag{1}$$

where $s(t)$ is the transmitted RF signal, $I(t)$ and $Q(t)$ is the I/Q data, and f_c is the carrier frequency. A BFSK transmitter modulates the frequency of carrier for symbols 1 and 0 by $+f_d$ and $-f_d$ respectively.

$$s_1(t) = \cos(2\pi t(f_c + f_d)) \tag{2}$$

$$s_0(t) = \cos(2\pi t(f_c - f_d)) \tag{3}$$

Using the angle sum identity and the fact that cosine and sine are even and odd functions respectively, $s_1(t)$ and $s_0(t)$ can be rewritten in the I/Q format from Eq. (1) with $I(t)$ and $Q(t)$ for the two symbols defined as

$$I_1(t) = \cos(-2\pi t f_d), \quad Q_1(t) = \sin(-2\pi t f_d) \tag{4}$$

$$I_0(t) = \cos(2\pi t f_d), \quad Q_0(t) = \sin(2\pi t f_d). \tag{5}$$

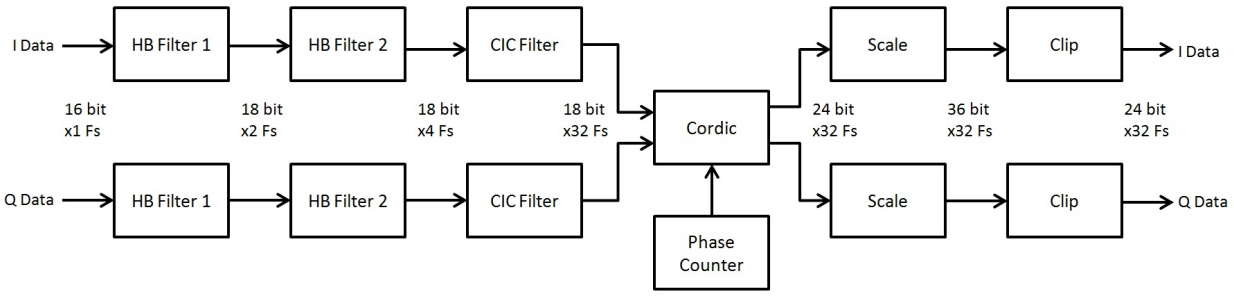

Figure 2: DUC diagram of USRP B200.

Typically, a SDR takes low-rate I/Q data generated on a PC, increases the sampling rate, and then shifts it to Intermediate Frequency (IF). Figure 2 shows a block diagram of the USRP's Digital Up Converter (DUC). Aside from the functional blocks, resolution and example sampling rate information is noted at each step for a case of upsampling by a factor of 32. The I/Q data travels through parallel paths, starting with two half-band filters that each increase the sampling rate by a factor of two. A Cascaded integrated-comb filter is used for the final upsampling. The I/Q data is then converted to IF through a Cordic block, and finally scaled and clipped before being sent to the RFIC for transmission.

For high data rate applications, however, it is more advantageous to generate the I/Q data onboard the SDR's FPGA. This has two benefits. First, it reduces requirements on the necessary transfer speed between the PC and SDR. Second, it offloads processing from the PC to the FPGA. The new transmitter diagram is shown in Figure 3, replacing the old DUC. Now 4 Mbs digital data is sent directly to the FPGA. The clock rate is set to the maximum value of 56 MHz in order to create the highest fidelity waveforms. The I/Q signals in Eqs. (4) and (5) are generated by Numerically Controlled Oscillators (NCO) implemented as Read-Only Memories (ROM). These contain 28 values representing one cosine period in order to support a frequency deviation of 2 MHz. Using the format of the original DUC at the Cordic input, these were structured as two's-complement Q2.15 values, where Qm.n is fixed point "Q" notation indicating m integer bits, n fractional bits, and one sign bit. The unsigned decimal values of the NCO were then

$$NCO = \begin{cases} [2^n x] + 2^{m+n+1}, & \text{if } x < 0 \\ [2^n x], & \text{otherwise,} \end{cases} \tag{6}$$

where x is the cosine sample value and $[x]$ denotes the nearest integer function. The I and Q phases are indexes to the ROMs, incrementing for a data bit of 0, and decrementing for a 1. Sine values for the Q data are simply obtained by shifting 90°, i.e. starting at index 7. FIR filters block the higher frequency components before using the rest of the original DUC blocks to complete the signal processing. In this manner, instead of generating and sending 56 Megasamples per second (Msps) I/Q data to the SDR, only 4 Mbs digital data is sent, the equivalent of 125 kilosamples per second (ksps) I/Q data. Figure 4 shows the spectrum of an example output of the BFSK transmitter.

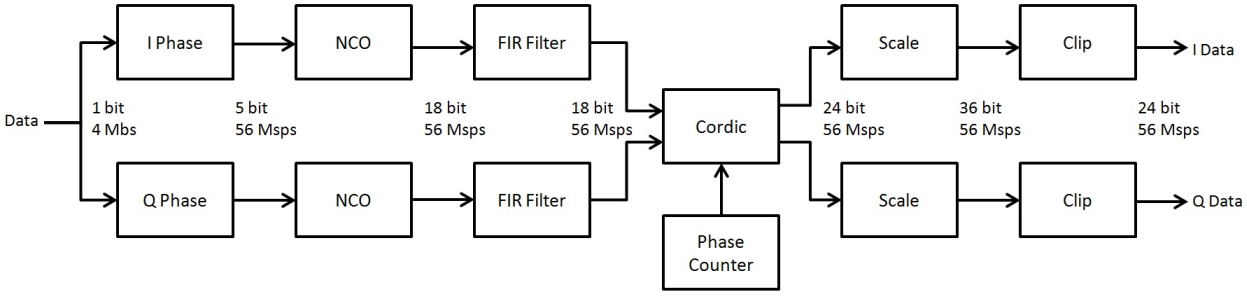

Figure 3: Modified FPGA transmitter block diagram.

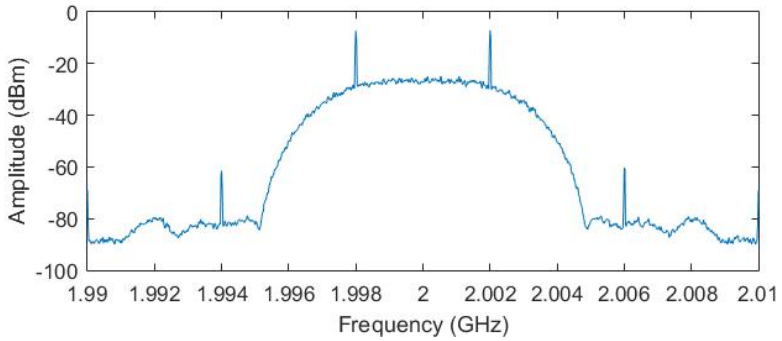

Figure 4: Spectrum measurement of transmit signal.

AES ENCRYPTION

Before presenting the encryption design, a short overview of AES is nessassary. The AES standard was established by the U.S. National Institute of Standards and Technology in 2001 [9] and approved for classified information by the NSA in 2003 [10]. It is a block cipher algorithm, i.e. encryption and decryption are applied to blocks of data as shown in Figure 5 on the left. AES uses 128 bit blocks, with 128, 196, or 256 bit keys. Blocks of data can be encrypted sequentially in order to encrypt an entire data stream. This simple application of a block cipher, called the electronic cookbook (ECB) mode, has inherent insecurities. Patterns can be detected in the data since identical data will always be encrypted the same way. This phenomenon is illustrated in Figure 6. The plaintext image on the left is encrypted using ECB mode to produce the ciphertext in the middle, where the ARL logo can still be detected. This problem can be circumvented by using the counter (CTR) mode shown in Figure 5 on the right [11]. The diagram on the left illustrates the encryption process. A counter is encrypted and xored with the plaintext to create the ciphertext. The same process is used to decrypt the chipertext, as shown on the right. The counter value increments by one for each block of data, with a random value chosen for the most significant bits called the nounce. Since it is the counter that is encrypted, and it is guaranteed to change with each block of data, there will not be any repetitive patterns in the ciphertext, as shown in Figure 6 on the right.

Using AES CTR mode with 256 bit keys, a simple encryption scheme was imposed on our typical frame format composed of 48, 16 bit words. The first two words comprise a 32 bit syncword, while the last three words are a 32 bit frame counter and a 16 bit Cyclic Redundancy Check (CRC). These

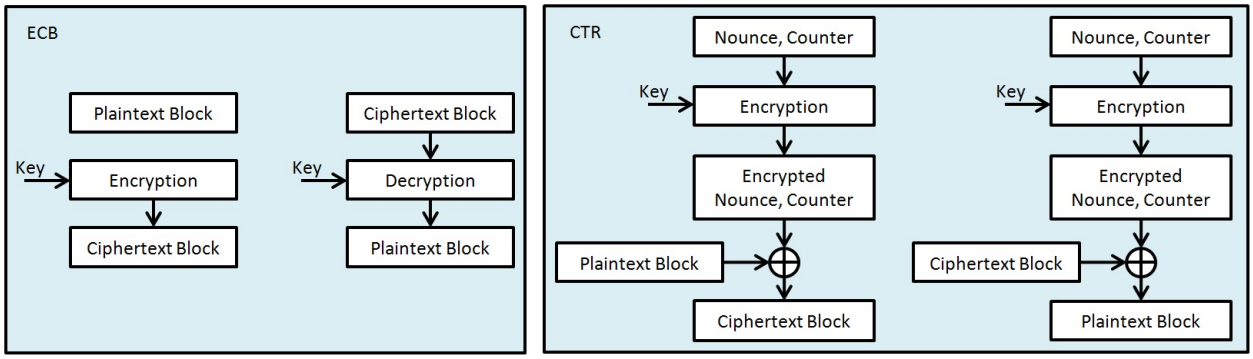

Figure 5: Block cipher encryption diagram using ECB and CTR modes.

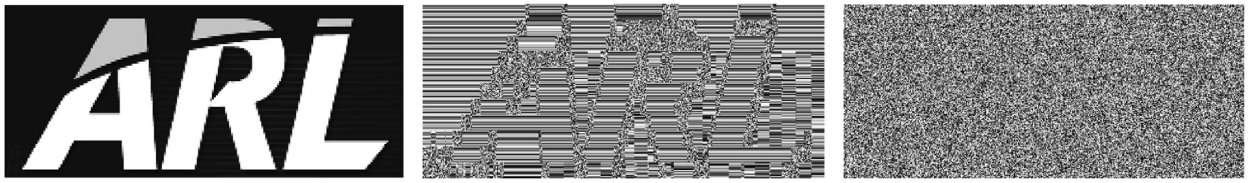

Figure 6: An image (left) encrypted using ECB mode (middle) and CTR mode (right).

words are left unencrypted, leaving 43 words of data for encryption. The first 40 data words are divided into 5, 128 bit cipher blocks. The last 3 data words are considered the most significant bits of a sixth cipher block, which can be encrypted normally, ignoring the least significant bits. The first block of 8 words is encrypted using a 96 bit nounce concatenated to the 32 bit frame counter from the previous frame. The counter value is incremented to encrypt each successive block, resulting in a final frame counter that increments by 6 for each frame.

Table 1 outlines an example frame encryption. The table starts with the last three words of the previous frame, showing the 32 bit frame count of x00000001. The first block of data in words 2 through 9 is encrypted using the previous frame count combined with a nounce. The counter continues to increment to encrypt each block of data, until the final block in words 42 though 44. The counter is incremented once more and inserted as the frame counter, which will be used to encrypt the first data block in the next frame.

This encryption scheme was implemented as a stand-alone board for encryption, and in the fabric of the SDR receiver's FPGA for decryption. Figure 7 shows a simplified block diagram of the encryption board. Unencrypted serial PCM data is input into the board, where frames are identified and words are extracted in the Frame Sync block. Detection of valid frames is indicated by asserting the Lock output. Words with their indexes are passed to into the frame encryption module, where a state machine in the Control block performs the encryption process. The state machine begins by waiting for a key load frame, which will load the AES 256 bit key and 96 bit nounce used for encryption. In order to simplify security, no data is output from the board before the key load frame is received. After it is received, all future frames are encrypted until a zeroize frame is received, or until the 32 bit frame counter reaches it's maximum value and the entire module is

Table 1: Frame encryption.

Word Index	Word Value	Encryption Counter
45,46	x00000001	
47	CRC	
0,1	SYNC	
2-9	Block1	x00000001
10-17	Block2	x00000002
...	...	...
42-44	Block6	x00000006
45,46	x00000007	
47	CRC	

reset. This allows for 38 hours of continuous transmission before reset using our typical 48 word frames and 4 Mbs data rate. A CRC module calculates a 16 bit CRC of the incoming frames, and is used to verify the key load and zeroize command frames. An Encrypting output provides feedback, indicating that the key load frame was successfully received and processed and the module is currently encrypting data. The Control block interfaces with the AES module[1] through a memory-mapped interface to encrypt the input data. Once a key load frame is received, the AES module is initialized with the key and encryption can begin. Input frames are encrypted as summarized in Table 1, with the unencrypted frame counter and CRC Out inserted at the end of the frames. Encrypted frames are serialized and passed through a 15-state RNRZ-L randomizer before being output for transmission. The final encryption board was a 1.5 inch diameter circular board using a Zilinx Artix-7 XC7A45T FPGA. Operation was verified up to 12.5 Mbs, but timing analysis indicates a maximum data rate of 152 Mbs.

For decryption, the FPGA of the SDR receiver in Figure 1 was modified to provide the additional capabilities shown in Figure 8. After the bitsync, a 15-state RNRZ-L derandomizer prepares the serial stream for frame synchronization. The CRC block then checks the frame CRC, replacing it with a 1 if correct or a 0 if incorrect. This is useful as a general telemetry system function that can be enabled independently of frame decryption. Key, nounce, and decryption enable settings are loaded into control registers from a PC through the interface program. As soon as enabled, decryption begins using the same procedure as encryption, except that instead of generating the frame count, the frame count is detected and used to decrypt the blocks of data in each frame.

PROTOCOL LAYER AND PROCESSING PIPELINE

A. TM STREAM GENERATION

The approach of using a DSP connected to a FPGA through a high speed synchronous serial interface provides a good balance between programmability and performance. This allowed the design

[1]The AES module was based on the avs_AES open-source code developed by Thomas Ruschival at www.opencores.org.

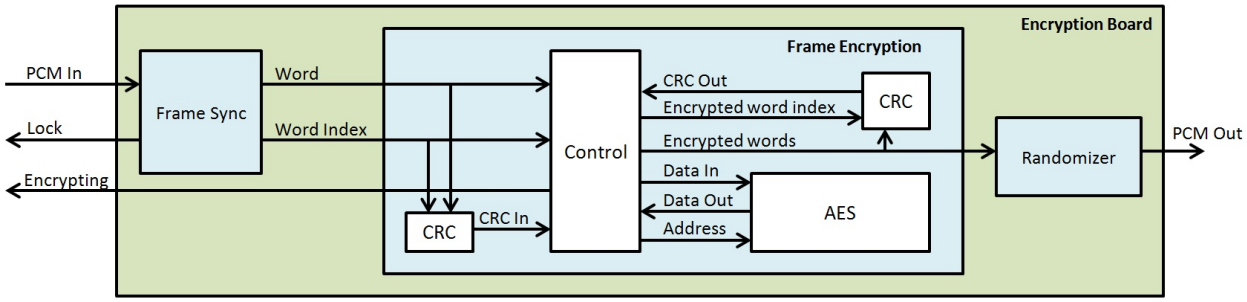

Figure 7: Encryption board block diagram.

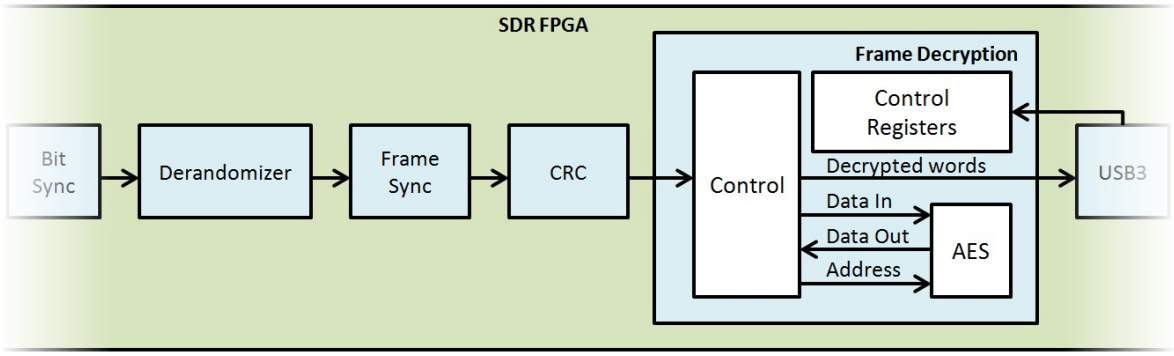

Figure 8: Decryption block diagram.

of all data handling and interfacing to external sensors through standard electrical interfaces. The DSP also allows for significant signal processing and estimation algorithms for real-time computations in the projectile for data compression and other data processing functions. It was determined the offloading of the AES encryption to the FPGA would allow for much higher throughputs of the telemetry data stream.

The DSP was then responsible for framing the data and processing incoming data payloads to generate the PCM stream. According to the IRIG standard [7], the PCM data is required to be placed into fixed length frames and embedded serial streams had to comply to a limiting set of requirements. Theses limitations are often burdensome and require significant frame and data planning to address the available channel bandwidth. Therefore, it is proposed to utilize a layered protocol where the top most protocol is the IRIG framing layer and subsequent protocols are embedded into this fixed length block. Figure 9 shows the structure of the telemetry stream. Each IRIG frame has a sync pattern as described in the previous sections with a Sub-Frame Identifyer (SFID) to identify the payload type, a frame counter, and checksum calculated using a CRC algorithm. This structure allows for creating unique SFIDs for specifically identifying fixed length frames which could comply with traditional standards. Utilizing a SFID also provides a universal framework for communicating with the SDR where standard fixed length frames may be added to the TM Stream. Each payload section is created using an asynchronous stream of messages.

The DSP creates the variable length message payloads through a First In First Out (FIFO) system

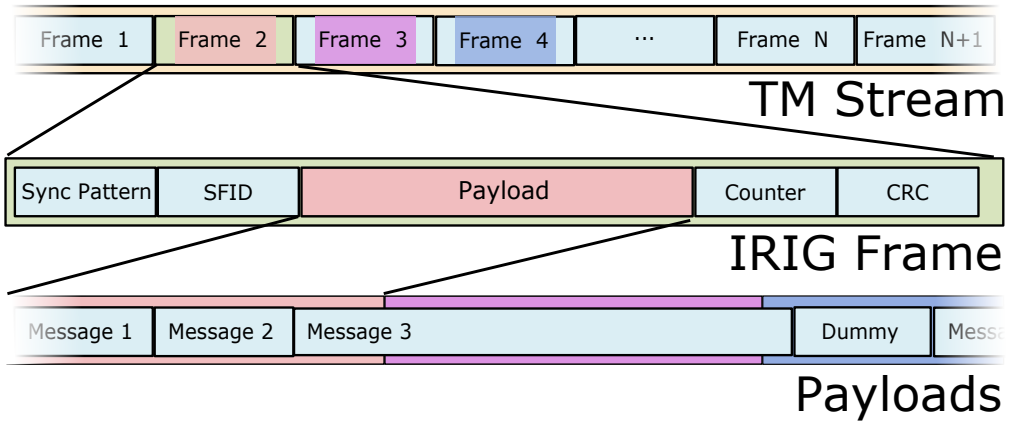

Figure 9: Protocol layer diagram.

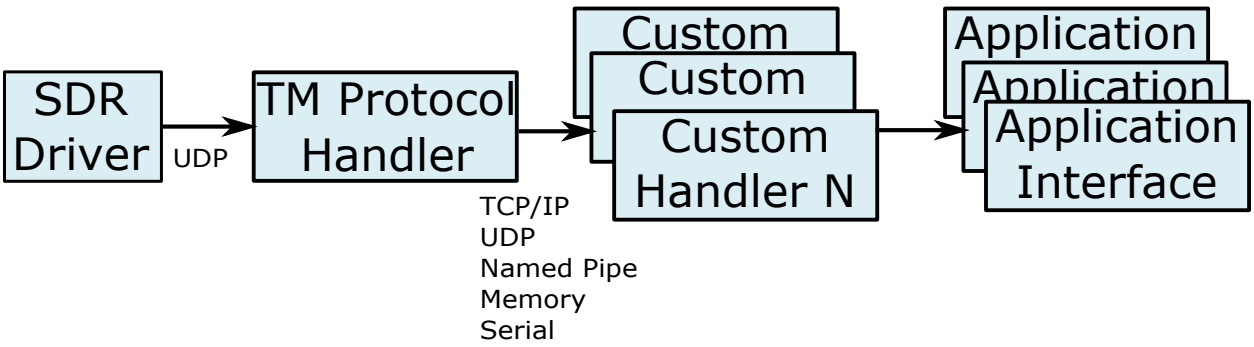

Figure 10: TM stream decoding.

and preserves the bitrate of the telemetry system through the injection of "Dummy" payloads when the FIFO is empty. Control of the payload is handled through a Direct Memory Access (DMA) channel to minimize the impact to the main CPU. Each message may have their own protocol and format. For example, a GPS message may have a uBlox protocol as described in the protocol specification [12]. Messages are thus only required to be defined as having a length and byte payload. For decoding the data, it is recommended to use higher level structuring of the data packet to ensure proper decoding.

B. TM STREAM DECODING

Figure 10 illustrates the Telemetry (TM) stream decoding process. The telemetry from the SDR first arrives through the SDR main driver to a UDP endpoint. This endpoint provides packets consisting of the fixed frames of data in Figure 9. The SDR ensures each frame has passed any decryption and CRC checks. A TM protocol handler then parses the standard IRIG frame and converts the remaining payload sections into a byte stream. The byte stream is then available on a variety of interfaces including Transmission Control Protocol/Internet Protocol (TCP/IP), UDP, Serial, Named Pipes, Memory, etc. These interfaces are then fed into a custom protocol handling software for decrypting to any application.

CONCLUSIONS

In this paper, we presented a SDR based telemetry for encrypted data transmission across a standard IRIG compliant S-Band channel. The telemetry link utilized multiple protocols for transmitting a variety of data sets using variable length messaging. The system was proven in a field experiment using range equipment at Yuma Proving Ground, AZ in March of 2017 where data was successfully encrypted, transmitted, decoded, and processed in real-time for data display.
-

REFERENCES

[1] H.-C. Chen, T.-H. Lin, H. Kung, C.-K. Lin, and Y. Gwon, "Determining rf angle of arrival using cots antenna arrays: a field evaluation," in *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*, pp. 1–6, IEEE, 2012.

[2] B. A. Fette, *Cognitive radio technology*. Academic Press, 2009.

[3] B. Davis, "Telemetry-based instrumentation for mortar systems," in *Proc. 2003 Mortars Conf*, pp. 4–8, 2003.

[4] M. L. Don, "Advances in telemetry capability as demonstrated on an affordable precision mortar," tech. rep., ARMY RESEARCH LAB ABERDEEN PROVING GROUND MD WEAPONS AND MATERIALS RESEARCH DIRECTORATE, 2012.

[5] Xilinx, "Vivado high-level synthesis," 2017.

[6] M. Iedema, *Getting Started with OpenBTS: Build Open Source Mobile Networks*. " O'Reilly Media, Inc.", 2014.

[7] Range Commanders Council Telemetry Group, Range Commanders Council, White Sands Missile Range, New Mexico, *IRIG Standard 106-04: Telemetry Standards*, 2004. (Available on-line at http://www.ntia.doc.gov/osmhome/106.pdf).

[8] M. L. Don, "A low-cost software-defined telemetry receiver," in *International Telemetering Conference Proceedings*, International Foundation for Telemetering, 2015.

[9] United States National Institute of Standards and Technology (NIST), "Announcing the ADVANCED ENCRYPTION STANDARD (AES)," *Federal Information Processing Standards Publication 197*, Nov. 2012.

[10] Committee on National Security Systems, "CNSS Policy No. 15, Fact Sheet No. 1: National policy on the use of the advanced encryption standard (AES) to protect national security systems and national security information," 2003.

[11] W. Diffie and M. E. Hellman, "Privacy and authentication: An introduction to cryptography," *Proceedings of the IEEE*, vol. 67, no. 3, pp. 397–427, 1979.

[12] *GPS.G6-SW-10018-F: μ-blox 6 Receiver Description Including Protocol Specification*.