A HYBRID DSP AND FPGA SYSTEM FOR SOFTWARE DEFINED RADIO APPLICATIONS

Vladimir Podosinov (Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, US; v_podosinov@vt.edu); Majid Manteghi (Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, US; manteghi@vt.edu);

ABSTRACT

There are multiple systems on the market for the software defined radio (SDR) research. The most commonly used platforms are USRP and USRP2. Lyrtech is one company that offers a commercial system, while Rice University has a research-minded platform called WARP based on the FPGA. In addition, there are other SDRs on the market, but they are for more specific applications. Lyrtech and WARP SDRs are costly, while USRP platforms are not very flexible. This paper presents an SDR built for the purpose of testing antennas that give an alternative system design at a competitive price for the SDR research. This SDR system is flexible, great for educating, and can be modified for more advanced applications if needed. The following SDR was built and tested under two different conditions, and the results are shown at the end.

1. INTRODUCTION

A hybrid system for the software defined radio applications began with the intention of building an SDR for the purpose of testing the frequency hopping antenna in [1]. The system needed to have the ability to quickly switch receiving and transmitting frequencies using an external trigger. The system needed to process up to 10 MHz of bandwidth and support multiple modulations so that the non-linear effects of the antenna could be observed. The main goal of the system was to perform bit error rate (BER) testing.

There are professional devices available to test BER performance, but they are usually very expensive. It is also possible to use a hardware defined radio to test antenna BER performance. However, the goal was to test multiple modulations, as well as the ability for the radio to work with particular sub-bands from the 10 MHz baseband spectrum. These sub-bands could be modulated at different symbol rates using different modulations. SDR was desirable because any future modifications to the BER testing platform could be performed easily.

The system implemented a total of 11 different modulations at the transmitter side, which are BPSK, BASK, BFSK, MSK, QPSK, 4-ASK, 4-FSK, 8-PSK, 8-PAM, 16-QAM, and 16-APSK. At the receiver, only linear modulations (ASKs, PSKs, and QAM) were implemented, as non-linear modulations require a different demodulator structure, and would not fit. The data rate was fixed at 1 Megasymbol/second (Msys/s), and the received signal was placed at 10 MHz digital IF frequency by setting the receiving oscillator 10 MHz lower than transmitter oscillator.

2. HARDWARE ARCHITECTURE

In order for the system hardware design to be rapid, commercial components were used. The overall system block diagram is shown in Figure 1.

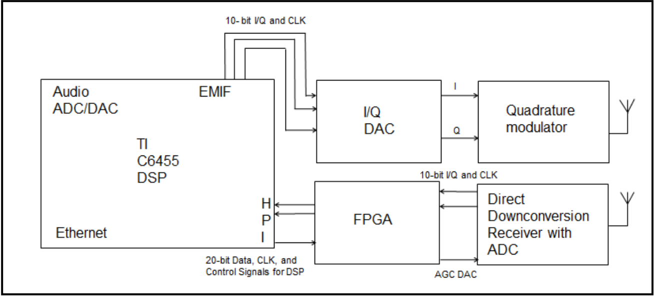


Figure 1

The main processor performing the modulation, demodulation, bit error rate calculations, and information communication with the PC was a Texas Instruments C6455 DSP on a Spectrum Digital C6455 DSK. The Spectrum Digital DSK board also provides a gigabit Ethernet port, a 16-bit audio analog-to-digital (ADC) and digital-to-analog (DAC), LEDs and switches that can be accessed by the user. The board also provides 3 peripheral connectors: external memory interface (EMIF), host port interface (HPI), and additional connectors with timer outputs and serial port (McBSP) connections.

DAC used for the signal modulation was a rapid prototyping module from the ComBlock COM-2001. COM-2001 has 2 10-bit parallel DACs with the maximum clock rate of 125 MHz for a complex analog baseband output. DAC outputs are band-limited to 13 MHz. Analog complex baseband passes to the ComBlock COM-4002D module. COM-4002D module is a quadrature modulator, with the frequency range of 950-1450 MHz, and a maximum output power of about 0 dBm. Receiver is based on the ComBlock COM-3002B module which is a direct quadrature down-converter where analog baseband signal is sampled at 40 MHz. Down-converted signal passes through a 20 MHz anti-aliasing filter, which is more than was needed, before being sampled by the ADC. COM-3002B works in 900-1575 MHz range, and can receive signals as low as -54 dBm for a full range ADC reading. COM-3002B also provides a 70 dB dynamic range AGC. A 40 MHz sampling rate is too fast for the direct input to the DSP and HPI port through which ADC is connected, and can only be controlled by an outside device. Therefore an FPGA was introduced between a DSP and a receiver ADC that performs sample rate conversion, matched filtering, and data transfer to the DSP. FPGA is also responsible for the AGC control of the receiver. FPGA is a Xilinx Spartan-3 FPGA on a ComBlock COM-1400 module. The COM-1400 module provides the ability to store an FPGA program on a flash memory and a sampled signal monitoring through ComScope, which is similar to the ChipScope.

The connection of COM-2001 and COM-1400 with the DSP are done using a ribbon cable via breadboard for the Spectrum Digital DSK shown in Figure 2. To reduce cost, a ribbon cable for PC to Laptop IDE bus conversion was used with one side soldered to the breadboard. The IDE ribbon cable was sufficient as the DAC clock rate was set at 12 MHz.

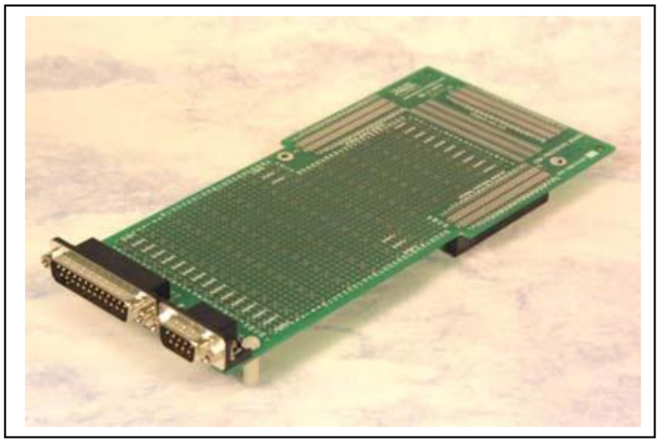
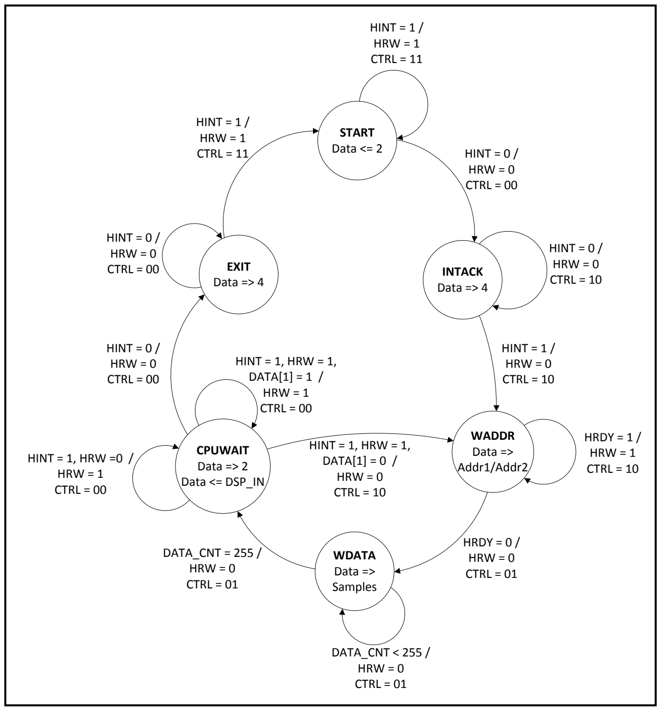


Figure 2

An external memory interface was used to connect DAC to the DSP, and an HPI port was used to connect ADC to the DSP. EMIF port was used with the DAC, because EMIF port provides the parallel data output needed by DAC and a reference clock. The only drawback is that the maximum DAC clock achievable without DSK modification is 19.2 MHz. The ADC was connected to the HPI port through the FPGA. FPGA reduced sample rate, as an HPI port can work at a maximum clock speed of about 60 MHz, and filtered the signal before the DSP demodulated it. As the HPI port is designed to be controlled by an external device, FPGA implements a finite state machine that writes 256 samples to the DSP memory before sending an interrupt to the DSP. A state transition diagram for the HPI port control is shown in Figure 3.


HINT = 1 / HRW = 1 CTRL = 11
START Data <= 2
HINT = 1 / HRW = 1 CTRL = 11
HINT = 0 / HRW = 0 CTRL = 00
HINT = 0 / HRW = 0 CTRL = 00
EXIT Data => 4
INTACK Data => 4
HINT = 0 / HRW = 0 CTRL = 10
HINT = 0 / HRW = 0 CTRL = 00
HINT = 1, HRW = 1, DATA[1] = 1 / HRW = 1 CTRL = 00
HINT = 1 / HRW = 0 CTRL = 10
HINT = 1, HRW =0 / HRW = 1 CTRL = 00
CPUWAIT Data => 2 Data <= DSP_IN
HINT = 1, HRW = 1, DATA[1] = 0 / HRW = 0 CTRL = 10
WADDR Data => Addr1/Addr2
HRDY = 1 / HRW = 1 CTRL = 10
DATA_CNT = 255 / HRW = 0 CTRL = 01
WDATA Data => Samples
HRDY = 0 / HRW = 0 CTRL = 01
DATA_CNT < 255 / HRW = 0 CTRL = 01
Figure 3

The finite state machine for the FPGA to DSP communication is a combo of Mealy and Moore state machines, because the HPI port latches data and control commands at different clock edges.

In addition to the transfer of data from the ADC to the DSP, the FPGA also performs additional signal processing such as AGC control, decimation, matched filtering, and sample timing correction. Overall FPGA signal processing blocks are shown in Figure 4. AGC block uses a technique similar to the binary search to set the gain level, and keep the signal within ADC limits. The AGC block also contains block memory which stores an ideally sampled version of the incoming signal generated in MATLAB and converted to the ADC samples format. Block memory in the AGC block was needed so that BER performance of the demodulator could be tested without the effects of the ADC and receiver clock offsets.

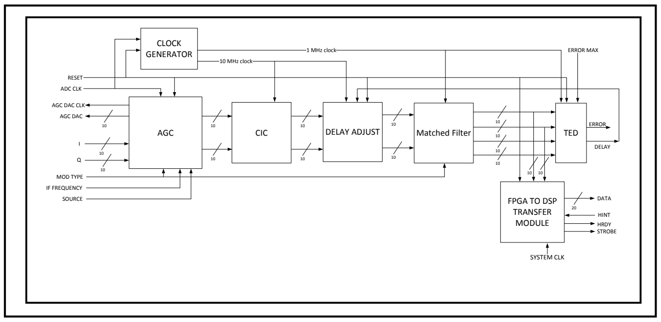


Figure 4

The signal stored in the block memory contained frequency and phase offset, and contained integer number of sinusoid cycles so that it could be cycled continuously. A detailed block diagram of the AGC is shown in Figure 5.

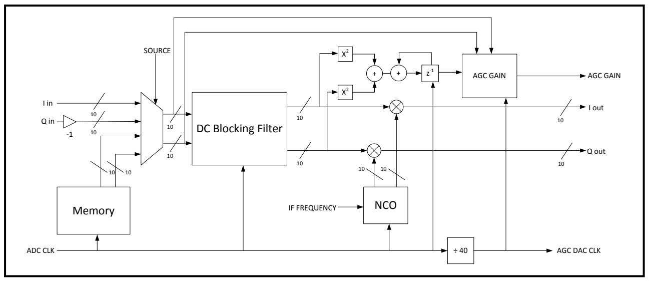


Figure 5

Also note that in Figure 5, the AGC block contains NCO and DC blocking filters. Even though COM-3002B is a direct quadrature down-converter, it was decided that a digital IF frequency was still needed. Digital IF frequency is needed so that the DC filter can remove any unwanted DC offset before the signal is sent for demodulation. DC Filter was implemented using a technique described in [2].

After the AGC stage signal is decimated using a 1-stage CIC filter, it is filtered using a matched filter. The matched filter is a running average filter because square pulses are used for the communication. There were a total of 5 samples per symbol. Due to the ADC clock jitter, matched filter sampling instant was constantly changing. Sampling error caused additional bit errors that were not expected. To reduce those errors an early-late gate timing error detector was used. An error was calculated every 8 symbols, and sampling time was adjusted based on error threshold. Sampling instant was adjusted using a delay FIFO through which samples pass before being match-filtered.

3. SOFTWARE ARCHITECTURE

All of the modulating and demodulating functions were performed in the DSP. Software was written in C language using a real time operating system from the Texas Instruments BIOS. TI BIOS was chosen because it allows the processing of multiple interrupts at the same time more efficiently and also allows software design to be more flexible for future code expansions. In addition to the modulating and demodulating functions, DSP also performs network communication to the PC. Software can be divided into 3 categories: transmitter software, receiver software, and network communication software. A block diagram of the transmitter software is shown in Figure 6.

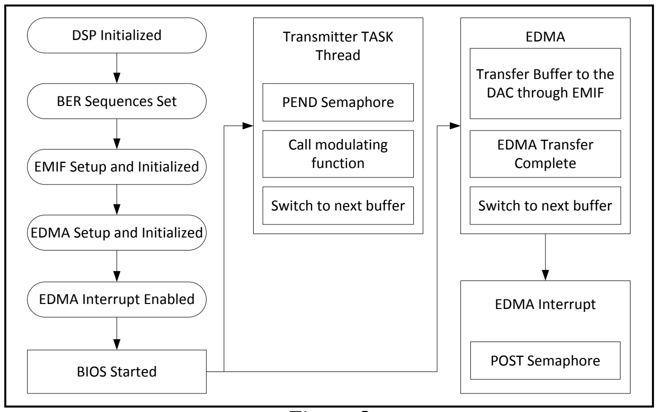


Figure 6

Enhanced Direct Memory Access (EDMA) is used to transfer data to the DAC using a double-buffering scheme. As DSP is working at much higher speeds than DAC, it fills one of the buffers while the other one is in the process of being transmitted. A timer is used to trigger DMA transfers such that the DAC receives a continuous stream of samples. When the DMA finishes transferring, another buffer is filled. For the transmitter a TASK is used, which is a BIOS thread with its own stack. TASKs are lower in priority than hardware and software interrupts.

A block diagram of the receiving software is shown in Figure 7. The receiver is implemented as a software interrupt thread.

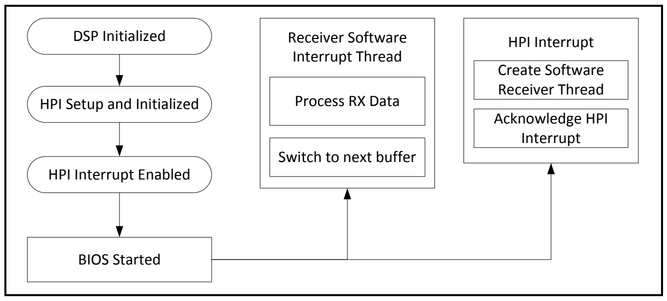

DSP Initialized
HPI Setup and Initialized
HPI Interrupt Enabled
BIOS Started
Receiver Software Interrupt Thread
Process RX Data
Switch to next buffer
HPI Interrupt
Create Software Receiver Thread
Acknowledge HPI Interrupt

Figure 7

The HPI port has its own DMA mechanism, so EDMA is not used in the receiving software. The FPGA finite state machine shown in Figure 3 transfers a total of 256 samples to the DSP internal memory before triggering an interrupt, which starts a receiver thread. The receiver performs frequency and phase recovery using decision-directed software PLL, the structure of which is shown in Figure 8.

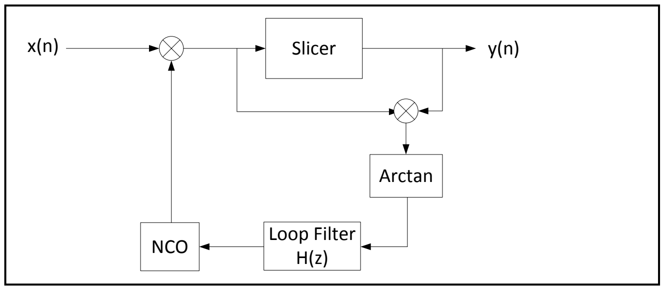


Figure 8

The loop filter is a second order proportional-integral filter with a loop bandwidth of 7500 Hz. This bandwidth is sufficient to synchronize onto signals with a frequency offset of 25 kHz or so. For 8-PSK modulation, an 8th power PLL was used instead due to 22.5 degree phase ambiguity. In order to design the loop filter in digital domain, an analog filter was designed using MATLAB's PI controller optimizer in SISO toolbox. After analog design was done, it was converted to the digital domain using a bilinear transform. The digital loop filter was implemented in the Direct Form in C code. A couple filter iterations were done, until a minimum loop bandwidth was found that still synchronizes with a frequency offset of 25 kHz.

In order to calculate the bit error rate between transmitted and received signals, an m-sequence of length 63 bits was used. M-sequence was chosen for BER testing because it is easy to synchronize to, and its pseudo-random nature makes the output spectrum valid. For larger constellations such as QPSK and QAM, each symbol modulated multiple m-sequences at the same time. Each modulated m-sequence used the same generator polynomial, but had a different phase. This structure allowed all symbols in the constellation to be tested, while simultaneously simplifying synchronization. At the receiver side, the synchronization loop was looking for the correct sequence phase offset before beginning error calculation. To avoid excess errors caused by phase ambiguity in case the phase flips, errors were only accepted as valid if the correlation peak was still there after a 63-bit correlation. Pseudo-code for this process is shown in the following box.

```
Correlate receiving bits against replica m-sequence
If Correlation == 0 or 63 ± Threshold
Compare every next received bit against current m-sequence value
Else
    Reset correlation value
    Increment initial condition
    Repeat
```

For the cases where internal FPGA block memory is used to create an input signal, an additional Gaussian generator was created. Gaussian generator was made using a Box-Muller transformation and a uniform variable generator consisting of 3 m-sequences for each uniform variable. For larger constellations such as 16-QAM and 16-APSK, too many resources were used on the DSP, and a large number of white noise samples generated in MATLAB were saved in the DSP memory.

To communicate error information between DSP and a PC, a network protocol was written. A UDP network protocol was utilized in order to save resources, and because it is simpler to use. A user GUI for DSP communication is shown in Figure 9.

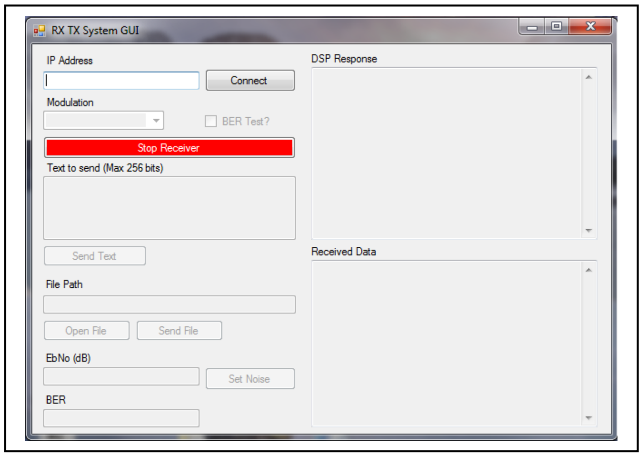


Figure 9

BER graphical user interface (GUI) was written in Visual Basic .NET language. A network protocol was created consisting of 9 commands that were 2-3 integers in length. As an additive feature to BER testing a data transfer capability was also added.

For FPGA settings control a ComBlock Control Interface was used.

4. TESTING RESULTS

There were four different tests performed on the constructed system. First DAC outputs were measured with the oscilloscope to make sure pulses were formed correctly. Secondly, an output spectrum was measured using a spectrum analyzer to see if it was correct as well as if there were any extra spectrum components. After that a model signal was generated in MATLAB and stored in the FPGA block memory and played back to DSP emulating received signal. DSP generated white noise was added to each sample before PLL, such that BER performance can be analyzed. For the last test, two monopole antennas were connected to the system and a wireless link was measured under line-of-sight and no line-of-sight conditions with high signal-to-noise ratio.

Examples of the DAC output and a spectrum output for the BFSK and QPSK modulation are shown in Figure 10, 11, 12, and 13 respectively. Spectrum output for those modulations is correct and so is the baseband DAC output. FSK modulations were implemented so that they could be demodulated non-coherently. A BER plot of the BASK, BPSK, and QPSK modulation is shown in Figure 14. BER curves were calculated using a signal that was read from block memory, passed through the whole chain, and had a noise added before the PLL. Performance of the SDR is close to theoretical when there is no ADC clock jitter present. For larger constellation modulations such as 16-QAM and 16-APSK, performance was about 0.5 – 1 dB off theory, which is still a good result.

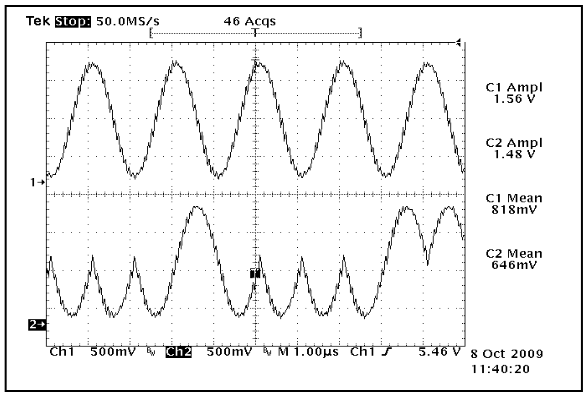


Figure 10

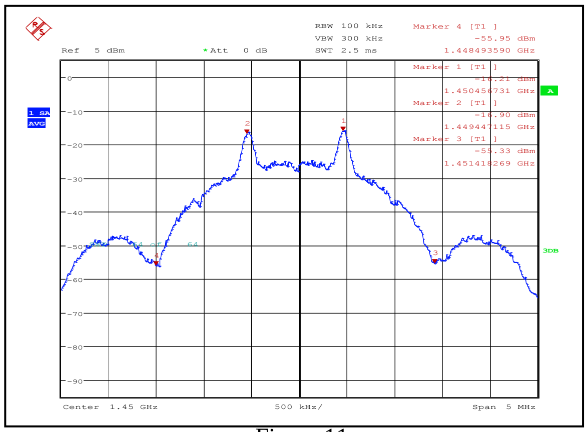


Figure 11

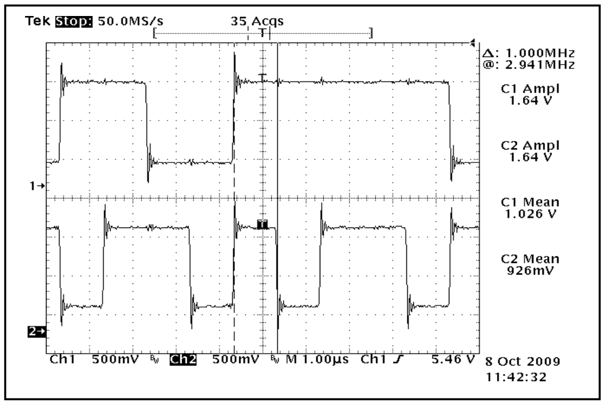


Figure 12

BER test curves for 8-PSK, 16-QAM, and 16-APSK are shown in Figure 15.

For the wireless link tests monopole antennas were separated by a distance of about five meters. The monopole antennas were hand-made antennas from thick copper wire and a metal plate so as to create a good ground plane. Center frequency was 1 GHz, and bandwidth was about 4 MHz. Figure 16 and 17 show samples of the received constellations for 16-QAM and 8-PSK modulations.

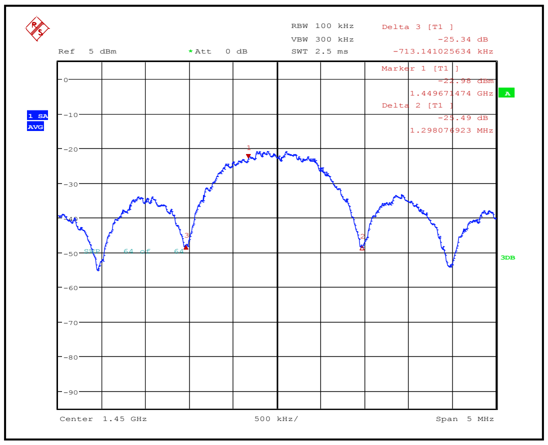


Figure 13

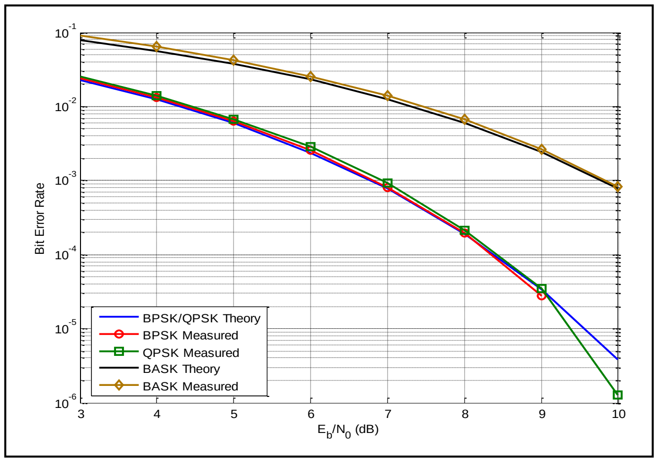


Figure 14

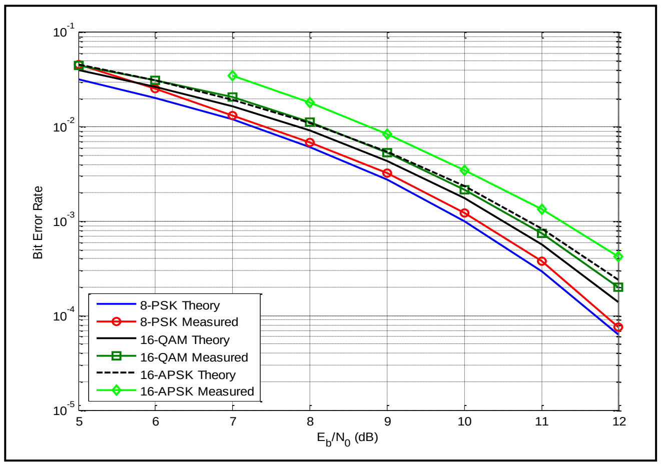


Figure 15

As it can be seen the constellations are correct and appear accurate. These conditions are valid when PLL and timing detector are synchronized. Table 1 and Table 2 provide results of the BER testing under high SNR with line-of-sight and no line-of-sight conditions respectively.

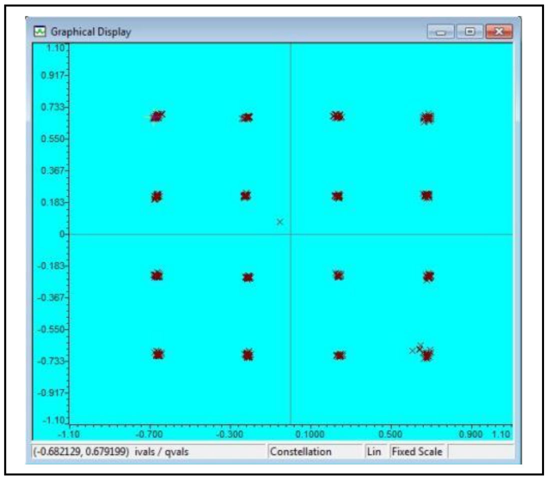


Figure 16

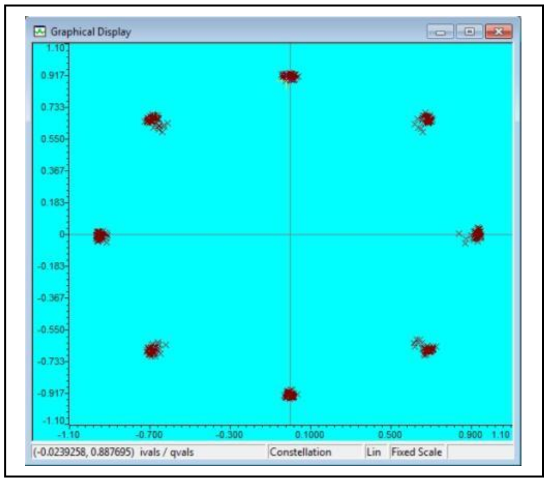


Figure 17

Table 1

Modulation	Error Test 1	Error Test 2
BASK	0.01643164	0.0122165
BPSK	0.0152784	0.01331759
QPSK	0.0072194	0.0188586
4-ASK	0.014609	0.014479
8-PAM	0.032588	0.032715
8-PSK	0.047863	0.0271668
16-QAM	0.0323732	0.024467
16-APSK	0.0248244	0.025308

Table 2

Modulation	Error Test 1	Error Test 2
BASK	0.0165107	0.0164654
BPSK	0.0138805	0.0130997
QPSK	0.021261	0.0235587
4-ASK	0.023284	0.020410
8-PAM	0.0367070	0.03299138
8-PSK	0.030490	0.0347004
16-QAM	0.0252258	0.02521068
16-APSK	0.027396438	0.02745781

It is notable that even with high SNR conditions, performance is still distant from the theoretical values, which should be around zero. However there seems to be no apparent multipath conditions with either case. Also results from two tests show that under stationary conditions overall BER values remain relatively constant.

5. CONCLUSION

This paper presented the structure of the alternative SDR system for research and education. Overall SDR cost is about $1400, which is comparable to the USRP2. The introduced SDR system is very easy to modify as it is built from simple blocks. Transmitter and receiver blocks can be purchased for different frequency bands, and a larger FPGA can be used as well. The DSP board contains audio ADC/DAC and an Ethernet port which can be used for future projects, such as audio codec testing. The DSP has special commands for communication processing, such as the Viterbi decoder, which can be used as well. In addition to that, the serial port can be implemented on the DSP board. The transmitter and receiver provide clock input so that frequency hopping can be implemented as well.

Power consumption by the system was not of importance, because the system was designed to work from a power supply. However, DSK's power supply is 5V and can supply up to 2A of current. FPGA, ADCs, DACs, and transceiver boards are powered from the DSK's connector, which is capable of supply up to 1A. At the moment of writing this paper, all non-DSK modules consumed around 600 mA of current. This means the overall system power drain is about 8-10W. As DSK's connections can provide more power, additional modules can be installed as well. Overall system power consumption can be improved by using a more energy efficient family of DSP processors, and using different kind of transceivers, as well other components.

The system is a great educational tool, because the signal can be observed at almost every single stage of the transceiver. As such this SDR system can be used in labs to show how each modulation actually looks in frequency domain, and it can also be used to show different RF impairments and how to solve them.

The SDR system was designed originally as a BER tester, and with additional modifications it can be effectively used as such. Based on no clock-jitter results, this SDR operates close to or at theoretical values. Under real conditions this system can be used as a BER tester only by relative comparison. A calibration antenna needs to be used before comparing it to the test antenna.

Additional improvements to this system that are still needed are better clock recovery methods to improve BER results, non-linear modulations demodulator, and possibly a better ADC clock to reduce jitter.

The source code for the system including C code for the DSP and a Verilog/VHDL code for the FPGA is freely available for download from the authors of this paper.

6. ACKNOWLEDGEMENTS

This work has been supported by the Federal Bureau of Investigation (FBI) under contract J-FBI-10-276.

[1] M. Manteghi, "A Switch-Band Antenna for Software-Defined Radio Applications," *Antennas and Wireless Propagation Letters, IEEE,* vol. 8, pp. 3-5, 2009.

[2] R. Yates and R. Lyons, "DC blocker algorithms," *IEEE Signal Processing Magazine,* vol. 25, pp. 132-134, Mar 2008.