LATVIAN JOURNAL OF PHYSICS AND TECHNICAL SCIENCES
2013, N 1
DOI: 10.2478/lpts-2013-0005

SATELLITE COMMUNICATIONS

SDPSK MODULATOR FOR RADIATION TOLERANT SATELLITE MODEM

A. Skorodumovs

Engineering Research Institute
"Ventspils International Radio Astronomy Centre"
of Ventspils University College,
101 Inženieru Str., Ventspils LV-3601, LATVIA

After a satellite has been launched, it is impossible to reprogram the hardware modulator. Therefore, a nanoRTU FPGA-based controller (*AAC Microtec)* in a modified modulation scheme may be programmed as a modem backend for software-defined radio (SDR) in the satellite communication system in order to adjust the balance of data rate *vs*. link reliability, enable coding or encryption system, change communication protocols, etc. For this purpose, a low-power radiation-tolerant reprogrammable satellite modem back-end has been realized for the satellite-to-Earth communication. This paper describes realization of a low data rate modulator based on a robust and reliable symmetric differential phase shift keying (SDPSK) modulation scheme combined with a raised cosine pulse shaping filter in nanoRTU, and presents results of its testing.

Key words: *SDR, SDPSK modulation, nanoRTU, FPGA based modulator.*

1. INTRODUCTION

The purpose of this work was to implement a low data rate modulator based on a robust and reliable symmetric differential phase shift keying (SDPSK) modulation scheme combined with a raised cosine pulse shaping filter. The modulator meant for nanoRTU – an FPGA-based radiation-tolerant and space-approved board for rapid cubesat deployment – would then be used for transmitting critical satellite data from a master RTU board or any other microcontroller-based circuit. The SDPSK modulation scheme used by Orbcomm [1] is tolerant to the frequency changes and Rician fading.

2. THEORY

SDPSK is a differential modulation scheme, which uses positive and negative 90° phase transitions for sending data (Fig. 1). Its main advantage over other DPSK modulation schemes is phase changing with every symbol. This means that the bit stuffing is no longer necessary, and each next received symbol helps to restore the symbol timing. Besides, the phase offset is no longer important since the data are encoded using the phase alteration rather than the phase itself. The phase rotation caused by transmitter and receiver frequency inequality can be

effectively averaged out, thus compensating for a low Earth orbit satellite's Doppler shift in a definite frequency range. An 11 symbol raised cosine finite impulse response (FIR) filter is used to smooth the phase transitions because of its nearly rectangular frequency spectrum. Filtering is needed to remove the fast phase transitions thus reducing noise [2]. Still, a good band-pass filter is necessary to completely reduce the digital noise.

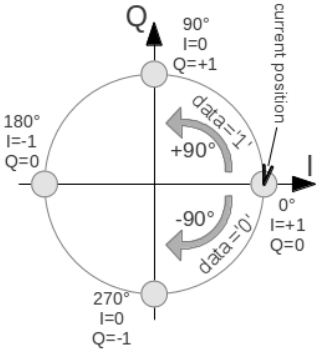

Fig. 1. SDPSK constellation.

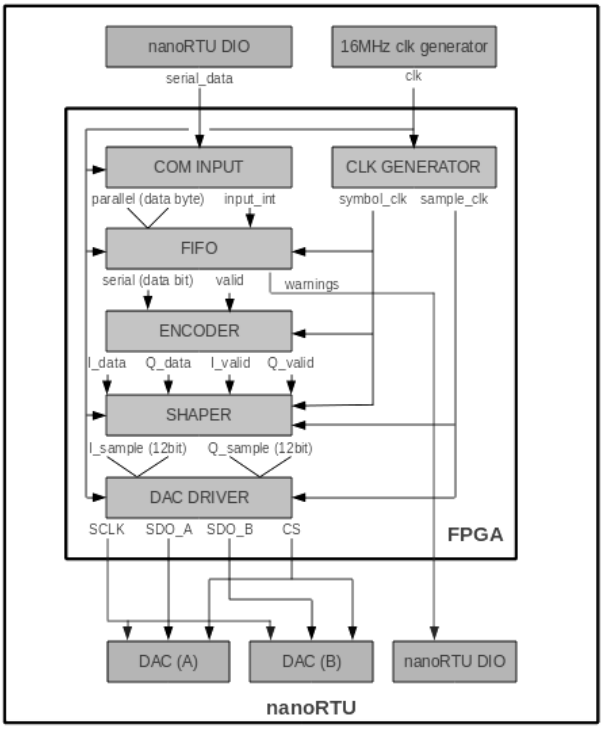

Fig. 2. Modulator structure.

The logic which underlies the modulator back-end is simple: the data are received from the host, their being buffered, encoded and split into I and Q streams, then filtered using a shaping filter, and, finally, sent to a DAC microchip for the analogue output. This whole process can logically be split into few blocks shown in Fig. 2 (detailed later in the text).

3.1. Clock

The clock block coordinates the whole modulator system except the receiver. It generates a short pulse in sym_clk line every 1666 clk cycles, indicating that a next symbol should be transmitted, thus setting the modulator output data rate to 9600 symbols per second. Since the modulator is designed to output 14 samples per symbol, the clock block generates a samp_clk pulse every 119 clk cycles. The data output and sample rates can be boosted by adjusting this block.

3.2. Receiver

This block is a simple asynchronous receiver that is hard-coded to 19200 8N1 settings. It receives eight data bits, outputs them to a serial bus and generates an interrupt for a buffer. The receiver is not controlled by the clock block.

3.3. Buffer

This block is an incoming data buffer with 5 byte slots. The incoming byte is stored on the falling edge of the interrupt generated by receiver (if there is at least one free slot in memory). The memory is a FIFO organized as a circular buffer with parallel 8 bit input and serial output. On sym_clk rising edge this block outputs a next symbol. Two signals – “serial” and “valid” – are used to describe a symbol at this stage. If the data are in the buffer, the “serial” would contain the next data bit, and “valid” would be set to one, indicating that real data are being sent. When the buffer is empty – “valid” is set to zero, indicating that the “serial” line contains a false value. In this way the encoder can determine what symbols should be treated as data, and what should not. Since the input is a 19200 baud per second and the output – 9600 symbols per second, the buffer is necessary in order to equalize the data flow. To prevent the buffer over- and underflows, the warning signals are used as feedback for the host. The following warning codes are sent: "100" – memory is empty, "101" – transmitting last byte, "110" – last free slot available, "111" - memory is empty, "0XX" – no warnings. These signals are used by the host device to determine if it is necessary or possible to send another byte to the modulator. In the case when the host latency is high, the buffer size can be easily increased.

3.4. Encoder

The encoder processes the incoming symbol on the sym_clk falling edge. The basic idea is that when a valid data bit is received, the phase rotates by 90° according to the SDPSK modulation principle. Then the resulting phase is split into I and Q channels and put out for symbol shaping. This block is coded in such a manner that the I and Q channel outputs are calculated from the incoming and previous values (since they represent a previous phase). The encoder output consists of two signal pairs which describe the phase: I_data and I_valid for the X coordinate, Q_data and Q_valid for the Y coordinate (Table 1). An unshaped SDPSK signal can be in any of the following five states: (+1,0), (–1,0), (0,+1), (0, –1) and (0,0) when no data are sent.

Table 1

SDPSK constellation states

#	I_valid	I_data	I-channel	Q_valid	Q_data	Q-channel	Phase
1	0	X	0	0	X	0	–
2	0	X	0	1	1	+1	90°
3	0	X	0	1	0	–1	270°
4	1	1	+1	0	X	0	0°
5	1	0	–1	0	X	0	180°

3.5. Shaper

The shaper block is an FPGA implementation of a FIR shaping filter. Normally, such a filter is very resource-demanding. A standard implementation of an 11-symbol filter with 14 samples per symbol and a 12-bit resolution per sample would consume a huge amount of resources. Since this modulator design has 119 clk's per sample, it is not necessary to calculate everything at once. For this particular FIR implementation it takes only 24 clocks to read the incoming data, calculate next I and Q samples using a shaping lookup table, and output the result. The I and Q symbol data are being read on the sym_clk rising edge, and the sample output is done at the samp_clk falling edge. It is possible to boost the symbol rate, the sample rate or the filter length if needed, since the sample calculation requires only ~ 20% of the available time.

3.6. Driver

The digital-to-analogue converter (DAC) SPI driver feeds two double-channel DACs with sample values. Each sample is fed to its own DAC device. Therefore, less time is needed to output the samples, which doubles the maximum sample output rate. After a reset, DAC microchip needs some time to be powered up before the use. A dd_ready feedback signal is used to tell the clock block that the driver is ready for work. The nanoRTU analogue channels 1 and 3 are employed to output the modulated signal, while channels 2 and 4 are set to a floating state during startup sequence, since they are intended for receiving the incoming signal for demodulation. The driver block is hard-coded to work with nanoRTU dual DACs and will not function correctly on other systems without heavy modification.

3.7. Input format

The nanoRTU data input is a standard 19200 8N1 asynchronous serial data stream (used in UARTs). In this way the nanoRTU modulator can be fed with data from almost any source – a PC with COM-port, a PC with USB port and USB-to-UART extension, any microcontroller with a built-in or software-simulated UART, or any of the RTU boards produced by *AAC Microtec*, provided that the nanoRTU digital input voltage requirements (3V TTL) are met.

3.8. Output format

The nanoRTU modulator output consists of two analogue signals in the 0–3 V range, centring at 1.5 V. A small +0.05 V bias in the signal should be compensated before mixing the channels to IF (intermediate frequency). The output

symbol rate is 9600 symbols per second, and the signal is formed using SDPSK modulation scheme with a raised cosine symbol shaping filter. The buffer feedback consisting of three warning signals serves as the output to the nanoRTU digital pins to prevent buffer over- and underflow.

Since it is very hard to troubleshoot the whole design, three test designs were made to incrementally test the modulator blocks:

1) DAC driver test scheme (Fig. 3);

2) encoder and shaper test scheme (Fig. 4);

3) buffer test scheme (Fig. 5).

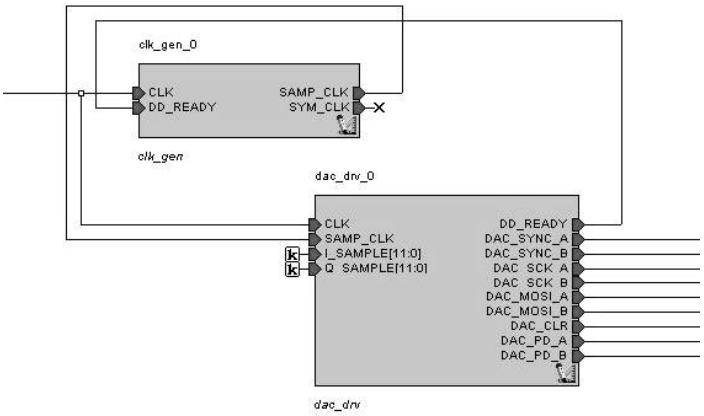

Fig. 3. Test design 1: DAC driver test scheme.

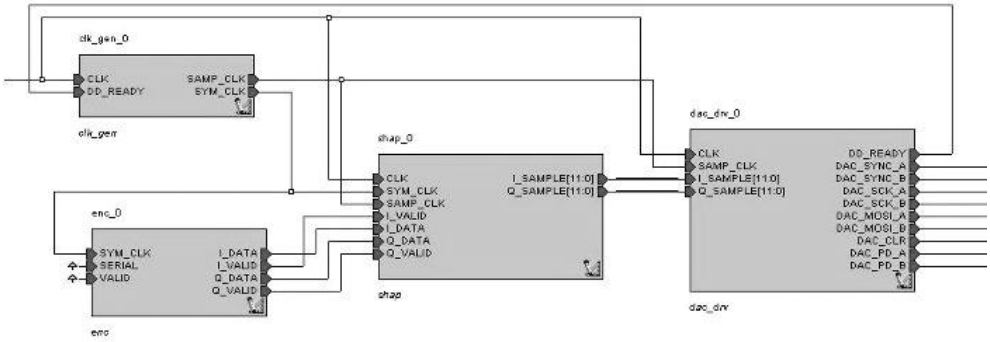

Fig. 4. Test design 2: encoder and shaper block test.

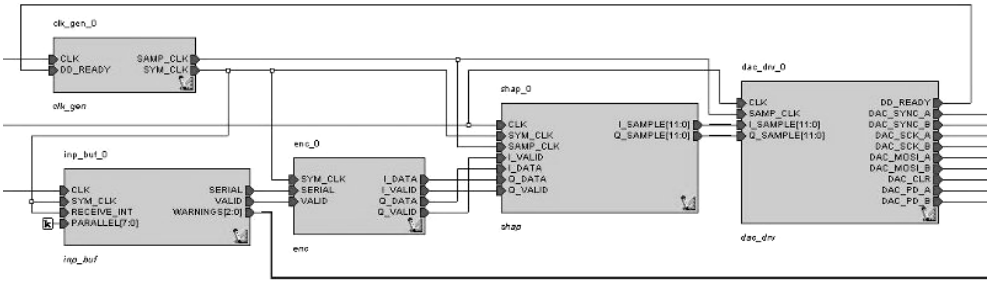

Fig. 5. Test design 3: buffer block test scheme.

This order was chosen to ease the interpretation of test results. An oscilloscope connected to nanoRTU analogue outputs 1 and 3 will provide all the data necessary to determine if the design is working properly. Each test was successfully passed. The only block which was not covered in this test series is receiver. This was simulated in Libero IDE and tested on nanoRTU board (apart from other blocks) by looping the received data back to the sender.

3.9. Test 1: DAC driver test scheme

In Test 1 (Fig. 3), I_sample and Q_sample signals are tied to constant 0x555 and 0xAAA values, so after startup analogue pins 2 and 4 should remain floating, while analogue pin 1 would be set to 1/3 of the reference voltage (1 V + 0.05 V bias), and analogue pin 3 – to 2/3 of the reference voltage (2 V + 0.05 V bias). If this works, the DAC driver block is functioning correctly. To ensure the result, other values were tested. Also floating analogue pins should be tested, for example by pulling them to VCC and GND with a 1K resistor.

3.10. Test 2: encoder and shaper test scheme

In Test 2 (Fig. 4), encoder and shaper blocks are added to the already tested DAC driver. Both the data and valid encoder inputs are tied to the logical one. In this way the encoder perceives them as constantly incoming "1" data bits (the same result is obtained sending a 0xFF byte). The result should be that the output phase would constantly rotate counter-clockwise. When symbols are properly shaped, the oscilloscope would show two identical smooth (14 samples per symbol) nearly sinusoidal waves with a 90° phase offset regarding each other (Fig. 6). The expected result can easily be calculated and compared with the one captured by oscilloscope.

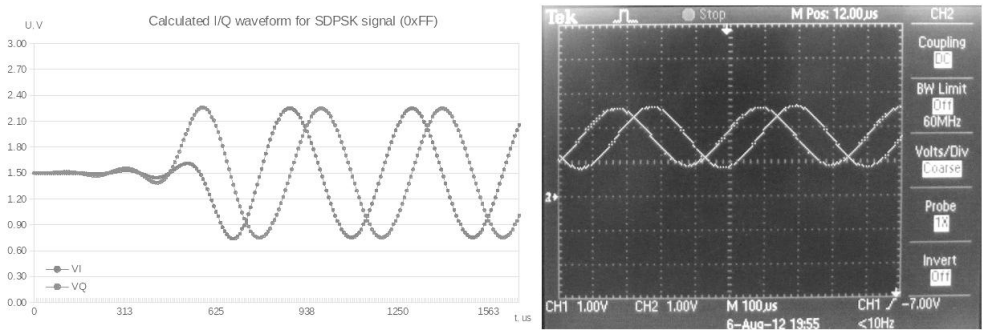

Fig. 6. Calculated and measured results of test 2.

3.11. Test 3: buffer test scheme

In Test 3 (Fig. 5), the buffer input is tied to a constant value 0x9D, and its interrupt pin – to a sym_clk. In this way with every sym_clk pulse the buffer will output a bit and store a new 0x9D byte. This will also test the buffer block for overflow handling (results in "111" warning). This also demonstrates that the receiver could be configured for any data rate over 12000 baud per second. A lower data input rate would result in constant buffer underflows. As in the previous

case, the resulting waveform was calculated and compared with the one captured by oscilloscope (Fig. 7).

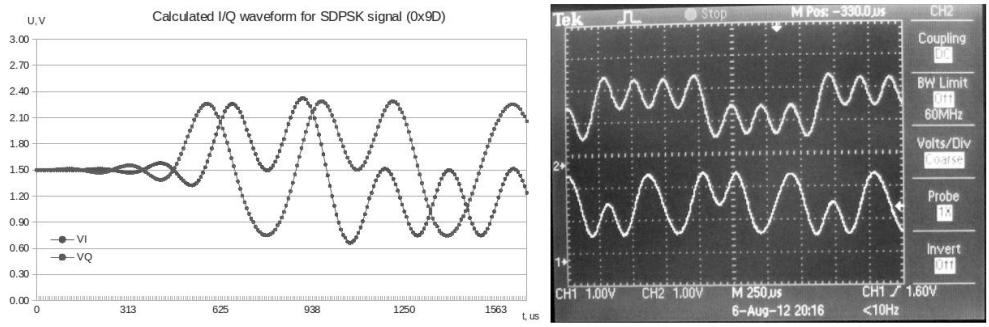

Fig. 7. Calculated and measured results of test 3.

4. CONCLUSIONS

The realized modulator scheme makes it possible to provide nanoRTU board modulator back-end functionality for communication.

Three test designs used to incrementally test modulator blocks and the modulator scheme have been proved to be fully functional.

The complete programmed SDPSK modulator uses only ~ 1/6 of the available nanoRTU FPGA resource, thus leaving much space for realization of demodulator.

ACKNOWLEDGEMENTS

The research was supported by the European Regional Development Fund within the project "Development of the Software defined radio (SDR) satellite communication model" (No. 2010/ 0266/ 2DP/ 2.1.1.1.0/ 10/ APIA/ VIAA/ 117).

The hardware and documentation were provided by ÅAC Microtec staff, namely Fredrick Bruhn and Jan Schulte.

REFERENCES

1. Prösch, R., & Daskalaki-Prösch, A. (2011). *Technical Handbook for Radio Monitoring VHF/UHF*. Books on Demand GmbH, pp. 245–246.
2. Harold, P.E., Stern, & Samy A. Mahmoud (2006). *Communication Systems: Analysis and Design*, *Pulse Shaping to Improve Spectral Efficiency.* Prentice Hall.

SDPSK MODULATORS RADIĀCIJAS NOTURĪGAM SATELĪTA MODĒMAM

A. Skorodumovs

K o p s a v i l k u m s

Kosmiskajās misijās iespējamas situācijas, kad rodas nepieciešamība pārkonfigurēt satelīta komunikācijas kanāla parametrus – izmantojamo modulācijas shēmu, pārraides protokolu, dekodēšanas algoritmu utt. Šajā rakstā izklāstīta SDPSK modulatora struktūra, kas optimizēta izmantošanai nanoRTU radiācijas noturīgajam aparatūras risinājumam, kā arī apskatītas modulatora darbības pārbaudes shēmas un izklāstīti iegūtie rezultāti.