Implementing DSP-based algorithms in digital modulation tasks

Emil E. Vladkov

The specific computational architecture of a Digital Signal Processor is the preferred candidate for solving various algorithmic subtasks related to high speed digital modulation. Utilizing a versatile modulator board interconnected to an DSP evaluation system a general state diagram and particular algorithms for QAM symbol coordinates remapping, generating pseudo-random sequences for testing purposes and adding pseudo-random sequence based noise of different amounts to the QAM constellation points/symbols are proposed. The algorithms are implemented in assembly language programs to achieve maximum efficiency and speed in the typical real-time environment of the proposed design. The concept of Direct Memory Access (DMA) is proposed and implemented for the indispensable high speed communication channel between internal DSP-memory and the converter hardware. The capacity and efficiency of the presented firmware working on the custom-build hardware is examined and proved in experiments with different noise-free and noisy QAM constellation plots.

Прилагане на основани на цифрова обработка на сигналите алгоритми в задачи по цифрова модулация (Емил Владков). *Специфичната изчислителна архитектура на Цифровите Сигнални Процесори е предпочитания кандидат за решаване на различни алгоритмични подзадачи, свързани с високоскоростната цифрова модулация. Базирано на използването на гъвкава платформа на модулатор, свързан към развойна система на цифров сигнален процесор, са предложени обща диаграма на състоянията и конкретни алгоритми за определяне на координатите на QAM-сивмоли, генериране на тестови псевдо-случайни последователности и добавяне на различно количество шум с псевдослучаен характер към точките на QAM констелационната диаграма. Алгоритмите са реализирани като програми на асемблерен език за осигуряване на максимална ефективност и скорост в типичната, изискваща работа в реално време, среда на предложената система. За осъществяване на необходимия високоскоростен комуникационен канал между вътрешната памет на сигналния процесор и преобразувателите е предложена и използвана концепцията на директния достъп до паметта (DMA). Възможностите и ефективността на предложеното програмно осигуряване, работещо върху специално разработения за целта хардуер, са изследвани и доказани посредством експерименти с различни чисти или с добавен шум QAM констелационни диаграми.*

I. Introduction

Based on the general concept of a DSP-based Quadrature Amplitude Modulation (QAM) hardware, presented in a separate article [1], different algorithms for controlling the DM9753 evaluation board through the ADSP21061 EZ-KIT LITE evaluation system [2], reformatting user data and generating test sequences are proposed and implemented in the form of assembly language programs. The algorithms make extensive use of the computational and programming features, specific to a Digital Signal Processor, so the different subtasks can be executed with maximum efficiency and in real-time.

II. The State Diagram of the firmware

The algorithms put into practice by the firmware software running on the ADSP-21061 digital signal processor [3] are designed to support different modes of digital modulation with the concept in mind that other new modulation schemes shall be easily added. The different tasks of the DSP-program are summarized in Fig.1, which represents the state diagram of the coordinating algorithm. The program cycles between 6 states and the change to the next state is initiated by asserting an interrupt by pressing the Interrupt Request button IRQ1 on the EZ-KIT LITE SHARC Evaluation board. The other control button on the

board (FLAG1) should be depressed to switch between the 6 states. The 0-State is the initial start-up state of the modulator. The output is a QAM-16 coded PRBS (Pseudo-Random-Bit-Sequence) data, which can be used for system evaluation purposes in a QAM-signal generator. The "State 1" is the same QAM-16 coded PRBS data, except that the output of the QAM-mapper (the two I- and Q-outputs) is Root Raised Cosine (RRC) filtered to assist in removing the Inter-Symbol-Interference (ISI) at the QAM-receiver (this feature will be discussed separately) [4]. The next state "State 2" increments the data carrying capacity of the QAM scheme through a 64 symbols constellation diagram (QAM-64). Again PRBS data is transmitted in a test generator mode of operation. The next "State 3" brings the PRBS-generator idea further to the QAM-256 modulation scheme. "State 4" is the true user mode of operation of the DM9753 digital modulator – user data in a 1000-locations 48-bit wide buffer (*data_buffer*) is QAM-16 modulated and transmitted in sequential bursts (one burst for each reloading of the buffer with new data). The user program working together with the modulator firmware should load the buffer and set a flag (*new_data_flag*), signaling that data is ready to be transmitted. The last "State 5" is again an user mode of operation state providing QAM-16 mapping and Trellis Coded Modulation (TCM) for a better error performance in noisy environments [5].

The QAM-16 constellation is subdivided in sets and not all transitions between symbols in the sets are valid ones – it is an easy task for the receiver to determine in following dedicated allowed paths in the data history which was the actual data transmitted despite of errors occurred on the channel. In "State 4" every 48-bits wide data location is subdivided (delimited) in 12 4-bits symbols for QAM-16 remapping. As the Trellis Coded Modulation is a Convolutional type of encoding it adds 1 bit to every 3 user bits to provide for the added error prone performance. So the 48-bits wide word with user data is subdivided in "State 5" in 16 3-bits wide symbols. This symbol delimiting process is depicted in Fig.2. The QAM remapping for the "State 4" user mode results in an 6000 location QAM buffer filled with DAC-ready data for the 1000 48-bit wide words of user data. Due to the 1-bit overhead added by TCM in "State 5" the same amount of user data produces an 8000 locations QAM buffer with remapped DAC-ready data.

In every state of the QAM modulator algorithm additive white noise can be added to the quadrature modulated symbols. The noise is added on both the I- and Q-axes in equal amounts. The addition of this real world impairment to the test signal is accomplished by pressing the other button on the SHARC DSP evaluation board – the FLAG1-button. As long as this button is pressed noise is added to the output of the modulator. Noise can be added on both the PRBS-data and the user data for compliance testing of receivers for example. The noise amplitude can be changed by pressing the FLAG1-button and simultaneously pressing the interrupt IRQ1-button. As a noise source another PRBS-generator is used and its output is added to the least significant bits of the I- and Q-data loaded into the DACs of the modulator board. So an easy way of controlling the amount of noise added (the blurriness of the constellation diagram) is to pick different number of bits from the serial stream of the noise PRBS-generator. The noise settings cycle through 10 states, gradually incrementing the noise-PRBS bits from 1 to 10 and then cycling from the beginning again.

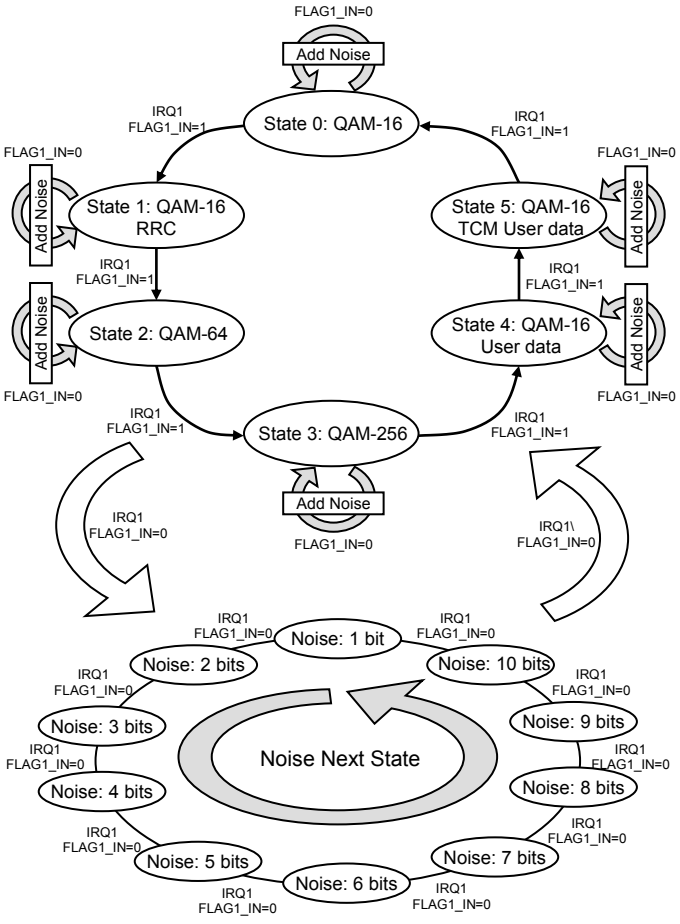

Fig.1. State Diagram of the digital signal processor software for controlling the digital modulation process.

As data from the the PRBS-generator (in QAM test generator mode) alternately fills (after remapping) two buffers (*PRBS_data* and *PRBS_data1*), which are

swapped during the implemented DMA mode of operation when transmitting the I- and Q-data to the modulator board, a visual indication of the PRBS data buffer reloading process is obtained through the blinking of the FLAG0 LED on the DSP board (one cycle blink for one reload operation).

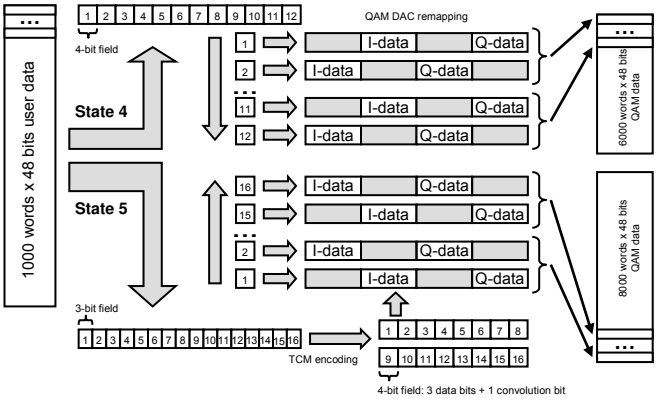

Fig.2. Delimited data structure of user data for "State 4" and "State 5" of the algorithm.

III. DSP algorithm of the PRBS-generator

The source of data for the generator mode states of the digital modulator (States 0, 1, 2 and 3) is a software implementation of a Pseudo Random Bit Sequence (PRBS) generator. It uses the classical polynomial generator $1+x^{18}+x^{23}$ of the length $2^{23}-1$ as specified in ITU recommendations O.151 and O.152 and used for scrambling implementations in DSL-modems [6], [7]. The functional diagram of the PRBS-generator algorithm is shown in Fig.3 and it's software implementation in the Fig.4 listing. The algorithm makes extensive use of the R11 register from the register file of the ADSP-21061 signal processor. This register is 32-bit wide when not used in Extended Precision Floating Point Mode (in this case the register is 40-bit wide). The size of R11 proves absolutely adequate as for the implementation of O.151 and O.152 actually only a 23-stage shift register is needed (or a 23-bit delay line as depicted in the functional diagram in Fig.3). The bits at the boundary between D18 and D19 (bit 18) and after D23 (bit 23) are easily extracted with the FEXT (field extract) assembly language instruction of the SHARC DSP. The bits are then XOR-ed through a dedicated XOR-instruction and then put back at the beginning of the shift register through the use of the FDEP (field deposit) instruction. Then the whole R11 register is shifted left by one bit to emulate a hardware shift register. The whole operation is organized in a DO-UNTIL loop with the LCNTR (loop counter) expired condition for exiting the loop. So if 8 new PRBS bits are needed to supply the QAM-remapper with 2 new 4-bit symbols (for QAM-16) the loop should be executed 8 times. It is important to note that at start-up the R11 register should be initialized with a non-all-zero value, as the opposite will stick the PRBS-generator to a non-changing state. The actual value used for the R11 is 0x0FFF FFFF.

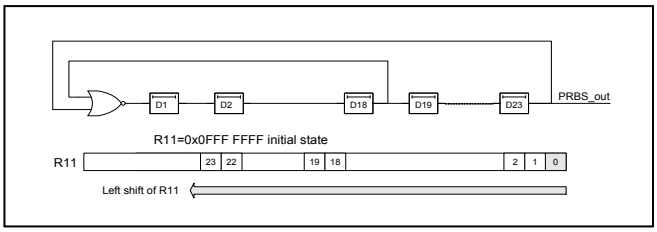

Fig.3. PRBS generator algorithm functional diagram and register implementation.

```
        LCNTR = 8, DO PRBS_loop_16 UNTIL LCE;    /* PRBS generator loop */
                R11 = LSHIFT R11 BY 1;           /* performs 2^23 -1 polynomial */
                R10 = FEXT R11 BY 18:1;
                R9 = FEXT R11 BY 23:1;
                R8 = R9 XOR R10;
                R11 = BCLR R11 BY 0;
PRBS_loop_16:   R11 = R11 OR FDEP R8 BY 0:1;
```

Fig.4. Software implementation of the PRBS algorithm.

IV. DSP algorithm of the QAM-remapper

The QAM-remapper is the kernel of the DM9753 digital modulator architecture and algorithm (Fig.5). The DSP code for the remapping process is shown in the Fig.6 listing. The first goal is to determine the maximum and minimum deviations of the symbols in the constellation used on both I- and Q-axes. These deviations have to be converted into valid 12-bit words, as the two DACs used in the DM9753 evaluation board are 12-bit devices. There shall be headroom provided in the DAC-output for the addition of noise later, so the actual maximum deviations in both positive and negative directions on both axes are set to 75% of the dynamic range of the DAC. As the DACs are differential output devices, their "zero" corresponds to the midscale value on both outputs (no difference between outputs) and this is exactly the code 1000 0000 0000 in binary representation. To obtain the maximum deviations in both directions the half-dynamic range value multiplied by 0.75 is added/subtracted from this midscale value. The DAC-values for the points in-between the whole dynamic range boundaries are obtained by calculating the Δ-parameter for the respective QAM – this should present no difficulty as the maximum deviation is clearly associated with an nΔ value (for QAM-16 it is 3Δ for example) [8]. So the constellation point coordinates are calculated independently for the different QAM-mappings. How these DAC-values are output on the data-bus of the

SHARC evaluation board to the QAM-daughter board depends on the hardware wiring. As the DACs used (AD9753) clock out converted output voltages twice the rate they are clocked, two symbols at a time have to be supplied. So in one remapped DAC-word they are actually two QAM-symbols coded. These two symbols (actually their I- and Q-coordinates) are shifted by 12-bit to be concatenated to one 24-bit value for every DAC, so that the two ports of the DAC are fed with data simultaneously. Because they are two DACs on the modulator board the resulting remapped word occupies the whole external data bus of the SHARC EZ-KIT LITE from DATA0 to DATA47. To accomplish easier routing of the PCB the MSB and LSB bits of every port are flipped (the whole DAC-value is flipped), so the data in the remapping Look-up tables (which is the 12-bit data calculated as described above) is flipped too. For example the value 1010 0000 0000 for the Δ-coordinate on both axes in the QAM-16 case becomes 0000 0000 0101 (005hex).

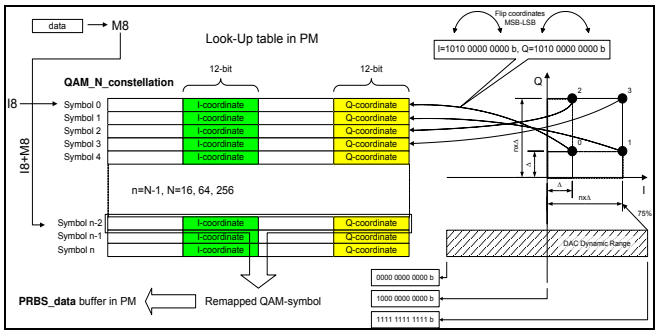

Fig.5. The process of association of symbols with I- and Q-coordinates – the QAM-remapping.

The whole set of symbols for the respective QAM (16 for QAM-16, 64 for QAM-64, 256 for QAM-256) is ordered sequentially (in incrementing number order) and their coordinates on the I- and Q-axes are evaluated. A 48-bit value is associated with every symbol, but only two 12-bit fields of this 48-bit word are occupied – one for the I- and one for the Q-coordinate, the other bits are zeroed to be filled with the second symbol coordinates by concatenating the two symbols in one 48-bit word. The whole look-up table is stored in the Program Memory (PM) space of the DSP (only the PM space is 48-bit wide to accommodate instructions for the DSP). So the remapping look-up table is constructed – there is one look-up table for QAM-16, one for QAM-64 and one for QAM-256. The downright process of remapping is simple – the symbol value is an index to the remapped value in the table. The procedure is illustrated with the code in Fig.6. The 4-bit pattern (16-QAM) from the PRBS-generator output is extracted and resides in R12. With this value the modifier register M8 of the Data Address Generator 2 (DAG2) of the DSP is loaded. Then an indirect (index addressing) program memory fetch into the 48-bit wide PX-register is accomplished. The index register I8 is initially loaded with the start of the remapping look-up table. The PM(M8,I8) structure performs the memory fetch with premodify (a specific addressing mode of the DSP architecture). It means the M8 is first added to I8 and then the new address I8+M8 is used as the PM address of the location to fetch data from. I8 is not updated with any new value – it remains pointing to the start of the look-up table. The I- and Q-coordinates for the first symbol are now loaded in the PX register. Next the coordinates of a second QAM symbol are to be extracted and concatenated to the first one coordinates. As PX will be needed next the R7 and R6 registers are utilized as temporary storage for the both parts of PX – PX1 and PX2. Again the fetch from the look-up table through index addressing is performed for the second symbol with another set of 4 PRBS bits used as the source of the random data. The process of concatenating the two sets of coordinates follows. PX1 and PX2-data is shifted to the correct locations in registers R2 and R3. Then the shifted result is OR-ed with the previous set in R7 and R6. The partial results are returned to the PX-register and its content is written to the *PRBS_data* buffer in program memory through index addressing (I9 is used to point to *PRBS_data* buffer). This whole procedure is performed in a loop until the whole *PRBS_data* buffer is filled up with remapped data and is ready for DMA-transfer to the modulator board.

```
R12 = FEXT R11 BY 20:4;          /* extracts 1-st 4-bits from PRBS */
M8 = R12;                        /* and remaps them to QAM16 constellation */
PX = PM (M8,I8);
R7 = PX2;
R6 = PX1;
R12 = FEXT R11 BY 16:4;          /* extracts 2-nd 4-bits from PRBS */
M8 = R12;                        /* and remaps them to QAM16 constellation */
PX = PM (M8,I8);
R1 = PX2;
R2 = LSHIFT R1 BY 12;
R3 = PX1;
R2 = R2 OR LSHIFT R3 BY -4;
R2 = R2 OR R7;     /            * combines QAM-data for 1-st and 2-nd nibble */
PX2 = R2;                        /* because the I and Q DACs work at 2x speed */
R3 = PX1;                        /* so for one PRBS_data location actually   */
R6 = R6 OR LSHIFT R3 BY 12;      /* two symbols are transmitted */
PX1 = R6;
PM(I9,M9) = PX;
```

Fig.6. DSP-code for QAM-remapping of the symbols.

V. Adding Noise to the output symbols

One of the applications of the proposed design is to be used as a real world source for testing QAM-receivers, so some kind of impairments (usually supplemented by the transmission medium) to the QAM-signal must be provided. One such impairment is the addition of noise to the I- and Q-channels resulting in both amplitude and phase noise superimposed on the QAM-signal. The blurred QAM-constellation (a simple model of it) is shown in Fig.7 and the functional algorithm for implementing a noise

impairment to the signal is represented in Fig.8.

Like the PRBS-source for the test data generator another PRBS-sequence is used here for simulating an additive white noise source. The generator polynomial is $1+x^{28}+x^{31}$, which gives much greater lengths of the pseudo-random period compared to the useful test PRBS-signal. So a real world noise-like behavior is simulated. The DSP-implementation of the algorithm shown in the Fig.9 listing is actually very similar to the Fig.4 program with the difference that here the whole length of another register from the ADSP-21061 register file – R15 is used as a shift register. In correspondence the feedbacks are taken from bits 28 and 31 of the R15, which is initially loaded with 0x0FFF FFFF to overcome the all-zeros condition. As the option is provided in the state diagram for changing the magnitude of the noise through the extraction of different number of bits from the PRB sequence and adding them to the QAM-remapper output, the loop counter (LCNTR) of the noise PRBS generator is not loaded with a fixed count value but is taken from a parameter variable (*noise_cntr*). This parameter variable is updated by the part of the software, which changes the noise parameters when pressing FLAG1 and IRQ1 simultaneously.

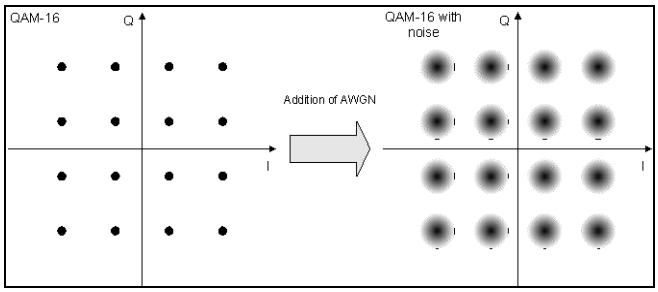

Fig.7. The effect of noise on the constellation diagram.

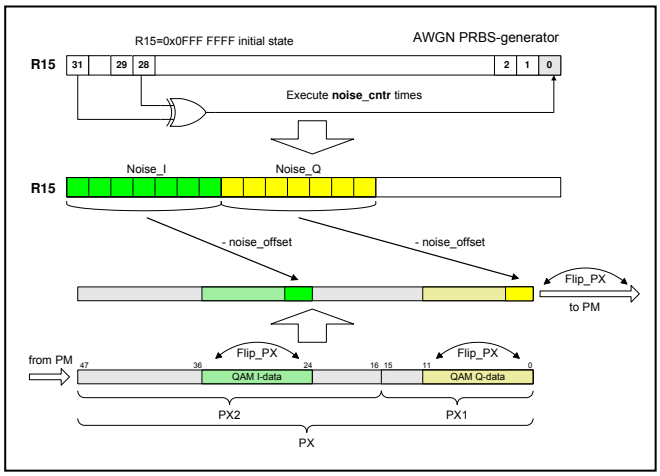

Fig.8. Functional diagram of the algorithm for adding noise to the QAM-symbols.

Part of the DSP noise code performs the addition of the PRBS noise-like data to the 48-bit long QAM-mapped data, which is read from the program memory of the DSP and resides in PX2 and PX1-registers (both parts of the big 48-bit PX-register). Due to the hardware implementation which maps the most significant bits of the SHARC data bus to the least significant bits of the DACs on the add-on board and the fact that the remapping software considers this feature, the swapping of the QAM-data in the PX-register is mandatory. This swapping is accomplished by a dedicated *Flip_PX* routine. The PRBS noise-like data stream takes only positive values, so before adding the noise data to the QAM-data the noise data has to be offset by one half of its magnitude. The noise offset is stored in the noise parameter variable *noise_offset*. It is convenient not to offset the noise but to subtract the offset from the QAM-data before adding the noise. Considering the I-data only following happens: The I-data resides in PX2, but it needs shifting by 8 locations to the right to align with the LSB-s of R14 for correct arithmetical manipulation. The I-noise-offset resides in R13 and is subtracted from R14, then the noise data occupying R12 is added to R14 and the result is back-shifted to its correct position and then returned to the PX-register (PX2). The same procedure is followed for the Q-data, but this time the R2-register is used for the arithmetic. Finally after the noisy QAM-data is ready in PX the *Flip_PX* operation is performed again and the data is transferred to the DACs on the daughter board.

```
Add_Noise:      R13 = DM (noise_cntr);                  /* Generation of white noise subroutine */
                LCNTR = R13, DO Noise_PRBS_loop UNTIL LCE;
                                R15 = LSHIFT R15 BY 1;  /* uses the 1 + x^28 + x^31 polynomial */
                                R10 = FEXT R15 BY 28:1;
                                R9 = FEXT R15 BY 31:1;
                                R8 = R9 XOR R10;
                                R15 = BCLR R15 BY 0;
Noise_PRBS_loop:                R15 = R15 OR FDEP R8 BY 0:1;
                CALL Flip_PX;           /* Flips the I,Q pair as LSBs and MSBs are interchanged */
                R13 = DM (noise_offset);
                R14 = DM (noise_I);     /* extracts noise value from the PRBS */
                R12 = FEXT R15 BY R14;  /* according to the noise parameters */
                R3 = PX2;
                R14 = LSHIFT R3 BY -8;
                R14 = R14 - R13;        /* adds noise offset to the I value */
                R14 = R14 + R12;        /* adds noise value to the I value */
                R3 = LSHIFT R14 BY 8;
                PX2 = R3;
                R14 = DM (noise_Q);     /* extracts noise value from the PRBS */
                R12 = FEXT R15 BY R14;  /* according to the noise parameters */
                R2 = PX1;
                R2 = R2 - R13;          /* adds noise offset to the Q value */
                R2 = R2 + R12;          /* adds noise value to the Q value */
                PX1 = R2;
                CALL Flip_PX;           /* Flips the I,Q pair to restore the DAC data bits arrangement */
                RTS;
```

Fig.9. DSP-code for adding noise to the QAM-symbols.

As the noise magnitude changes from 1 to 10 bits in 1-bit steps, so change not only the noise offset value but also the positions in the noise-PRBS shift register R15, where from the different bit length field shall be extracted. These location parameters are stored in the parameter variables *noise_I* and *noise_Q*. It should be noted that I- and Q-data are not blurred by one and the same noise samples at a time, as this will

obstruct the true random statistical behavior of the simulated system. So consecutive noise samples from the PRBS shift register are taken – one for I-noise and one for Q-noise. This makes both I- and Q-noise components not absolutely statistically independent, but with the respect to the simple (and fast) implementation of only one noise generator this proves to be a justifiable approximation of a real world noise source.

VI. Transmitting data using DMA

The process of computing the QAM-mapped data coordinates to fill the *PRBS_data* buffer takes more time to complete than the procedure of sending the data to the DACs. Because the converters are working at the full speed of the DSP external bus (40MHz) and actually two symbols are supplied to them every clock cycle (making the maximum allowable by the design symbol rate equal to 80Msymbols/sec) it is not reasonable to expect the ADSP-21061 Digital Signal Processor to prepare the data in real time (this preparation includes remapping, adding noise as an option, optional RRC filtering and TCM encoding – all time consuming and computation intensive tasks) [2], [3]. Therefore the logical structure depicted in Fig.10 is implemented, which utilizes two 4000 locations long buffers – *PRBS_data* and *PRBS_data1* – one of them being output to the DACs on the daughter board and the other filled with QAM-data from the signal processor in the same time. After the current buffer loaded with data is full and ready for transmission the two buffers are logically interchanged (swapped) and this alternating process continues until another state of the digital modulator is chosen. The process of emptying the active buffer is much faster than the process of filling the inactive one, so the proposed solution is to stop the transmission after the active buffer is send out to the DACs in the case of user data to be modulated, waiting for new user data to arrive, or sending repeatedly the PRBS test data stored in the active buffer as long as needed to fill the other buffer with data in the case of PRBS test generator. The last concept is illustrated in the second diagram in Fig.10.

Obviously the act of sending data to the DACs cannot be accomplished with a DM/PM (data/program memory) write instruction to the auxiliary memory space of the signal processor, where the converters reside. To achieve this at the 40MHz clock (and instruction) rate will mean to take up the whole processing power of the DSP for data transfers only every cycle and no time will be left for computing the new QAM-data for the inactive buffer. The ADSP-21061 architecture has many I/O features available, the most valuable in such a case being the Direct Memory Access (DMA) channel feature [2]. The design implements DMA Channel 6 to transfer data from internal memory (the PRBS data buffers) to external memory (the data converters) without any intervention from the DSP core. The only preparation to be done is to set up the DMA transfer before beginning the computation of QAM-data for the inactive buffer and to enable it.

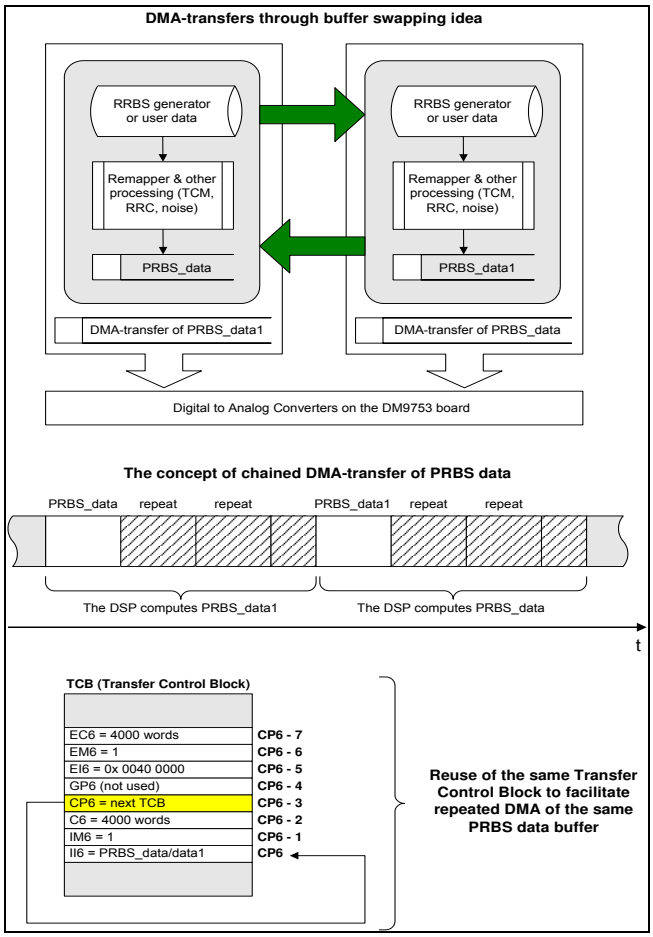

Fig.10. Transferring QAM-data to the DACs through Direct Memory Access.

The DMA from internal to external memory is prepared through the loading of 8 DMA Channel 6 registers with data. This configuration data specifies the starting addresses of data to be transferred in the source and the destination spaces of the DMA, the number of words to be transferred and if these words are transferred from/to sequential locations or locations are skipped (through the modifier register). The register names, their function and the initialization data for the actual design are presented in Table 1. The general purpose register GP6 and the Chain pointer register CP6 are not loaded with data in this case (a simple and not a chained DMA is

implemented). If the active buffer content has to be retransmitted until the other alternate buffer is ready, the process of "Chain Loading" shall be implemented. The Chain Loading mechanism implies that the content of all DMA configuration registers is loaded as a whole from a buffer with the parameters in data memory, called TCB (Transfer Control Block). One of these registers (the chain pointer) will point to the next set of parameters in data memory to be loaded after completion of the current DMA-transfer. So every DMA registers setup will prepare the next DMA registers setup. If the chain pointer (CP6) points to the end of the same (to whom it belongs) parameter set buffer then the DMA will repeat itself from the same and to the same location until stopped by disabling the DMA through the DMA Control Register 6 (DMAC6), which is implemented in the design with the process of chaining being presented in the last diagram of Fig.10. Setting up the TCB is not enough to start the DMA with chaining. The Chain pointer register (CP6) must be explicitly loaded with the pointer to the TCB.

Table 1

DMA Channel 6 parameter registers

Register	Register Function	Design initialization values
EC6	External count (number of words to transfer)	4000
EM6	External modifier (external address space increment)	1
EI6	External index (starting address for the external data buffer)	0x 0040 0000 (the beginning of the external memory space)
GP6	General purpose register	Not initialized
CP6	Chain pointer (address of the next TCB)	Points to TCB for chaining only
C6	Internal count (number of words to transfer)	4000
IM6	Internal modifier (address increment)	1
II6	Internal index (starting address for the internal data buffer)	Pointer to *PRBS_data* or *PRBS_data1*

Fig.11. The experimental setup used to test the DM9753.

VII. Experimental results with the prototype of the digital modulator

The measurement setup used to obtain the presented experimental results is shown in Fig.11 and consists of the DSP evaluation board and the modulator prototype board DM9753, connected to a scope set in X-Y mode of operation at the DACs output monitoring points. The symbol rate of the modulator was reduced from the maximum possible by design 80MSymbols/sec to 20MSymbols/sec. The whole system is controlled and the firmware code is loaded to the DSP through the VisualDSP++ software running on a standard PC [9].

The experimental results for different constellation plots are presented on the photos in Fig.12 – Fig.16. The scope screenshots show not only the constellation points but even the transitions between the separate states – in this way they constitute a form of two-dimensional eye-diagram. Fig.12 represents the QAM-16 constellation, on Fig.13 and Fig.14 the bit-rate is increased by transmitting symbols of the QAM-64 and QAM-256 constellations. Due to the momentary exposure and the source of the data being a PRBS-generator some of the points in the QAM-256 constellation are missing, but this is a presentation artifact only – actually all points are generated over the time. Fig.15 and Fig.16 show the influence of 9-bit and 8-bit noise on the constellations for QAM-16 and QAM-64 respectively. As expected the diagrams get blurred by the amount of noise applied. The output RF-power measured was -1dBm without any modulation, -7.11dBm for QAM-16, -8.00dBm for QAM-64 and -8.5dBm for QAM-256.

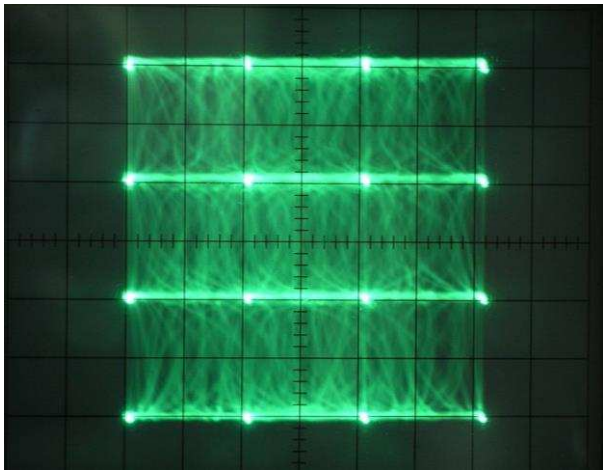

Fig.12. QAM-16 constellation diagram screenshot.

VIII. Conclusion

The experimental verification of the proposed algorithms for controlling a specialized digital modulator hardware through digital signal processing firmware programs in real-time confirms the usefulness of the unique DSP architecture in such computation-

intensive tasks, where large amount of data have to be both preprocessed and transferred at high speed to the analog front-end of a communication system. Future investigations on the implementation of error-correcting algorithms and algorithms to compensate for the transmission medium introduced signal impairments are to be performed.

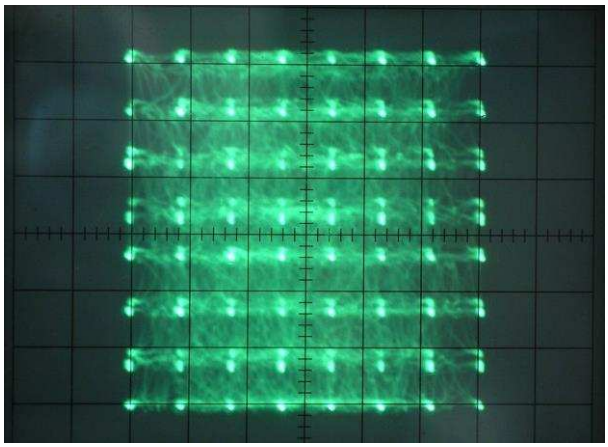

Fig.13. QAM-64 constellation diagram screenshot.

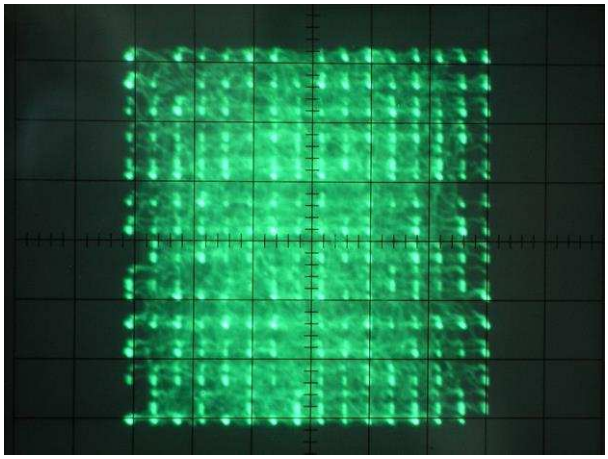

Fig.14. QAM-256 constellation diagram screenshot.

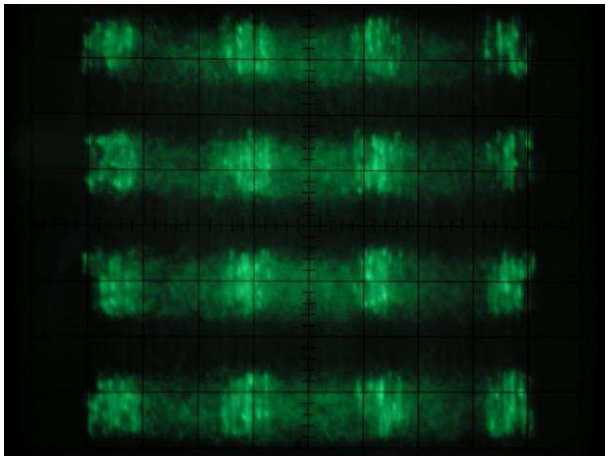

Fig.15. Noisy QAM-16 with added 9-bit white noise.

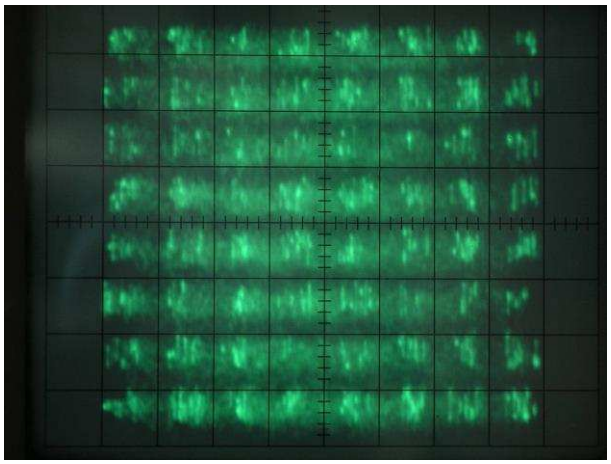

Fig.16. Noisy QAM-64 with added 8-bit white noise.

REFERENCES

[1] Vladkov, E.E. A general concept and implementation of a DSP-based QAM digital modulator. Electrotechnica & Elektronica E+E, Vol. 50. No 1-2 / 2015, pp.18-25.

[2] Analog Devices Inc., ADSP-21061 EZ-KIT LiteTM Evaluation System Manual. Revision 3.0, January 2003.

[3] Analog Devices Inc., ADSP-2106x SHARCTM User's Manual, Second Edition. May 1997.

[4] Langton Charan. Inter Symbol Interference (ISI) and raised cosine filtering. Intuitive Guide to Principles of Communications., www.compextoreal.com, 2002.

[5] Gitlin, R. D., Communications Theory: Trellis Coded Modulation [TCM]. Department of Electrical Engineering, Columbia University, 2001.

[6] ITU-T Recommendation O.151, Error performance measuring equipment operating at the primary rate and above.

[7] ITU-T Recommendation O.152, Error performance measuring equipment for 64 kbit/s paths.

[8] Colby, L. Modulation Error Ratio Specifications for QPSK and QAM Transmitters. Hewlett-Packard Corporation, Interactive Broadband Products, IEEE PROJECT 802.14, PHY Working Group, July 1996.

[9] Analog Devices Inc. VisualDSP++ 3.5 User's Guide for 32-Bit Processors. Revision 1.0, March 2004.

Assoc. Prof. Dr. Emil E. Vladkov *is currently a lecturer at Sofia University, Faculty of Physics, Dept. of Radiophysics and Electronics, he graduated from Sofia University in 1996 and became a PhD in 2001. His scientific interests cover the area of communication networks and protocols, Wireless Sensor Networks (WSN) and Digital Signal Processing Algorithms and hardware implementations.*

tel.: +359 888 099199 e-mail: evladkov@phys.uni-sofia.bg

Received on: 30.06.2015