From Pixeldata to COFDM – DVB inside

Thomas Kleffel (dg5ngi@maintech.de), Christian Daniel (dg2ndk@maintech.de)

15.11.2006

Abstract

Analog PAL based TV transmission is no more – DVB has taken over in Germany. This should be legitimate reason to take a really deep look into how the new technology works and what needs to be done to get your own transmission into the ether. The base for our lecture is a standard CVBS signal coming from a video camera. From there we will go with it through all the necessary stages of encoding, framing, multiplexing and modulating.

1 Introduction

With the shutdown of the classic anolog TV transmission in Bavaria at the end of May 2006 and with the foreseeable conversion to a transmission using the Digital Video Broadcasting standard [1] in Germany and Europe, the question to be asked is: Is it still possible to handle this new technology as an amateur with a non-enterprise budget?

Of course this question doesn't refer to the acquisition of a new TV set along with a DVB decoder in the next shopping mall, but to the issue of creating and sending your own DVB signal without the need for a chip prototyping setup...

In our paper we're going to deliver an answer to that question – and also an overview, how DVB works at all. The individual steps to be taken are no more complicated than they were with analog TV — but now there are much more things to be taken care of.

We used a lot of readily available components like the MPEG2 encoder, but our work lies in the creation of a FPGA[1] firmware for the modulation of the DVB-T COFDM signal. When that worked properly, we started including DVB-C as well and now DVB-S is the next item on the list.

2 Encoding

In the following we're going to give you an introduction to DVB and what has to be done in order to bring for example the video and audio signal of a camera into the ether. Luckily the major part of the DVB standard is independent of the actual distribution of the signal, i.e. DVB-S, DVB-C, DVB-T, and with some exceptions DVB-H use the same algorithms for coding, compression and packetizing.

Figure 1 on the next page shows the basic composition of our transmitter along with the most important signal paths.

2.1 Video

The lion's share of the transmission is taken up by the picture data – two to four MBit/s when using DVB-T, up to eight MBit/s on DVB-S, depending on the financial power of the TV station. Accordingly, the video codec is the critical subsystem for an effective utilisation of the available bandwith.

[1] Field Programmable Gate Array

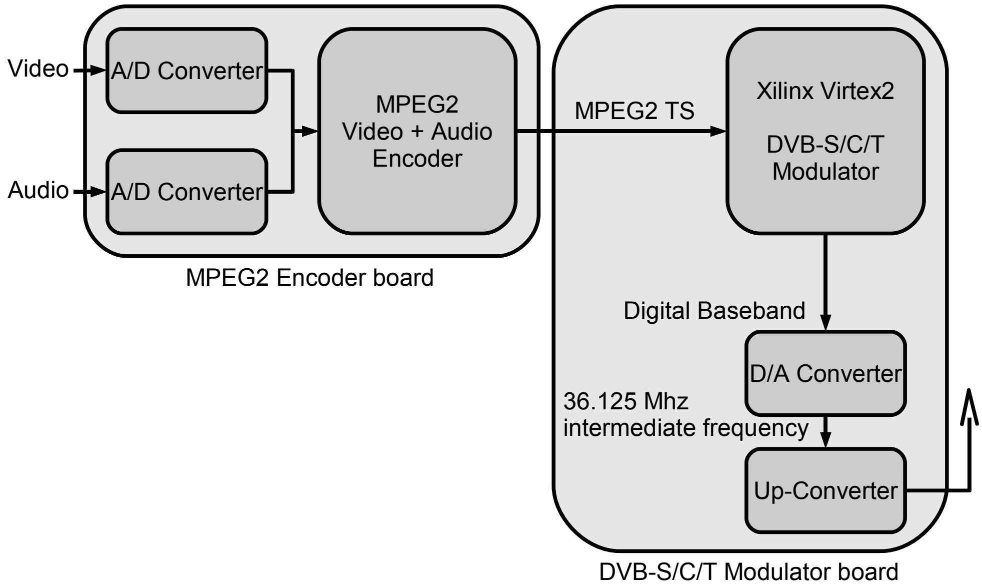

Figure 1: DVB modulator/transmitter overview

2.1.1 Input format

Base of operations is an analog CVBS[2] as it is delivered by any consumer video camera or a TV-out port of a computer graphics card. An A/D converter digitizes this signal and sends it to the MPEG2 video encoder via an ITU-656 interface [2]. ITU-656 is the most common denominator for delivering digital video pixeldata: The picture data is sent over an eight bit bus accompanied by the 27MHz pixel clock on a seperate signal line. Synchronisation is done using a known start-of-active-video bitpattern.

As colorspace not RGB[3] is used, but the YUV-format [3], which is common for TV transmission technology: Y is the luminance or brightness of a pixel, U and V contain the color value (chrominance). This format is one result of the invention of color TV: U and V were added to the old black-and-white signal in such a way that an old TV would just ignore them while still being compatible to the new TV standard. Since the brightness is already contained in the black-and-white signal, there is no need to transmit it again in the three RGB components thus YUV saves bandwith.

For the reception in the human eye another aspect is interesting: The eye distinguishes brightness differences with much greater resultion than different colourings. As a result of that, that it makes sense to transmit the Y-component with more bandwith than U and V.

In total the ITU-656 interface delivers about 206 MBit/s of data corresponding to uncompressed PAL video at an resolution of 720x565 pixels plus blanking interval.

2.1.2 Video-Compression

The actual video compression is defined in the ISO standard ISO/IEC-13818-2 [4] and shown in figure 2. Another very comprehensible description of the algorith can be found in [5].

The most crucial part of the codec is the prediction of future picture contents. This prediction tries to create an prognosis of the next frame by analyzing current and past data. The better this works the less the difference to reality is. As the encoder uses the same prediction algorith (or

[2]Colour Video Baseband Signal

[3]Red, green and blue as components

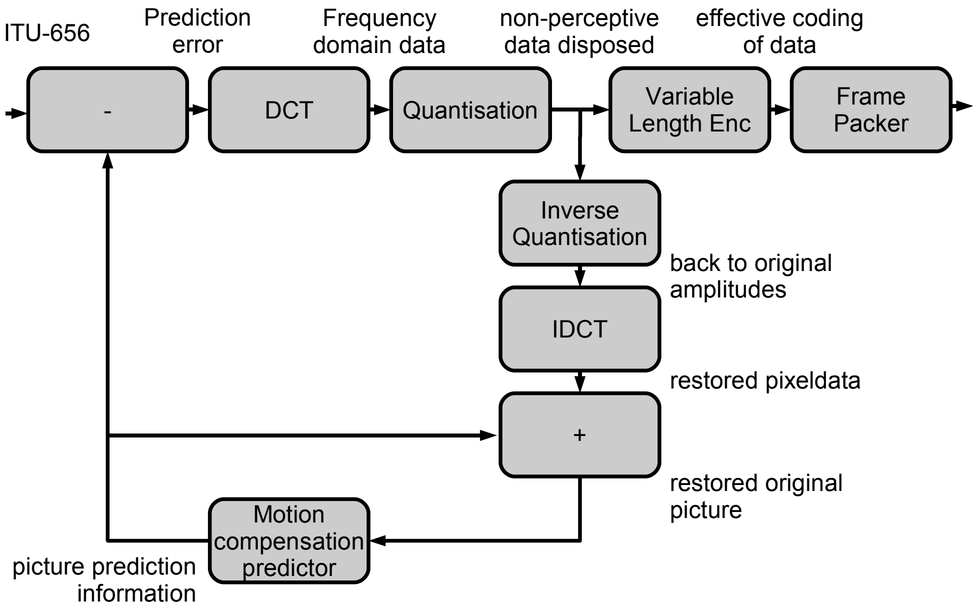

Figure 2: ISO/IEC-13818-2 video encoder

predictor) as the decoder, it can calculate what the receiver will predict. This information is used to finally encode and transmit only the corrections to this prediction.

As every prediction has to be initialized at some point, there are several kinds of picture encoding defined for the datastream:

Encoding	Kind of prediction
I (Intra-Frame)	no prediction – the frame is complete and is comparable to a JPEG image. This is a point where the decoder can newly start decoding the stream.
P (predictive coded picture)	This frame uses the predictor but only has references to frames in the past.
B (Bidirectionally predictive coded picture)	This frame uses the predictor and has references to frames in the past and to future frames.

With the existence of the B-frames it becomes clear that MPEG2 video doesn't send pictures mandatorily in the order they are displayed on the screen, but that the encoder may decide to send a frame earlier in order to allow for better predictions, thus saving bandwith in the end. The decoder has to have enough RAM to keep images available until no further references are possible. The encoder on the other side has to keep track of how full the decoder's memory is at any given moment.

In detail the prediction works by the codec trying to detect the movement of image areas. For example a ball could roll through the picture: The background will not change much but occupy the major part of the scene. The ball is being moved over the image, but also changes hardly – especially not if it is single coloured. Thus the predictor can assume that the object, which shifted ten macroblocks to the right from the picture before last to the last, will do the same thing in the next picture. Apart from that the background from the first I-frame can be reinserted without any need for retransmission of that image area.

Using this method, the codec reduces spatial redundancy (the same ball is shown at different locations) and temporal redudancy (background is first shown, then masked by the ball, then displayed again). Additionally there is a psychovisual redudancy: The eye cannot see all details of the picture with the same resolution. For a viewer it is immediately apparent if the edge of a sheet of paper on a dark desk is blurred, but he won't detect a light spot on the paper. According to that phenomenon the codec has to invest bandwith into edges but it can save on gradients.

To effectively find these edges, the codec doesn't work with the pixels the way they are delivered by the A/D converter, but the data is transformed into the frequency domain using the discrete Cosine transformation [6]. To be able to use the DCT, the picture is first divided into blocks of eight by eight pixels.

The worst case szenario is a complete random distribution of values inside a block, because then the frequency spectrum is completly used. But since this only happens rarely, it is another possibility to save bits: Most of the blocks contain only a light colour gradient or mostly one edge. Figure 3 shows two blocks and the DCT transformed equivalents.

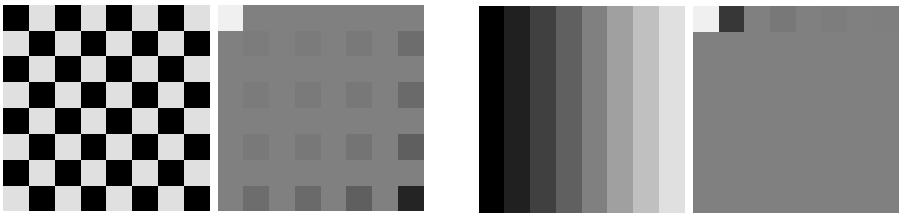

Figure 3: Checker and gradient pattern and their DCTed equivalents

Most of the bits are put into the DC part and the lower frequencies (shown at the top left area of the examples). There are several profiles defining the exact distribution of the bandwith; from these the encoder will chose one which serves a good compromise between data falsification and configured bandwith. This step is called quantizing – here the effective reduction of data is done.

After the quantizing step, only the coding of the prediction errors, the motion vectors and the newly transmitted blocks into a bitstream needs to be done. Here a variable-length-code is used, thus representing frequent symbols by short bitpatterns and rare symbols by longer ones.

Apart from the pure image data, the encoder also sends a collection of meta information which is important for the decoder. Alongside the size of the screen in pixels this meta information also contains data about the aspect ratio of the original data. Additionally a sync word is added to allow the decoder to lock onto the bitstream.

2.2 Audio

The audio codec is more similar to the video codec than one might suspect at the first glance. But the perception in the ear works very similar to seeing. This becomes clear while plotting the spectrum of a sound like it is shown in figure 4.

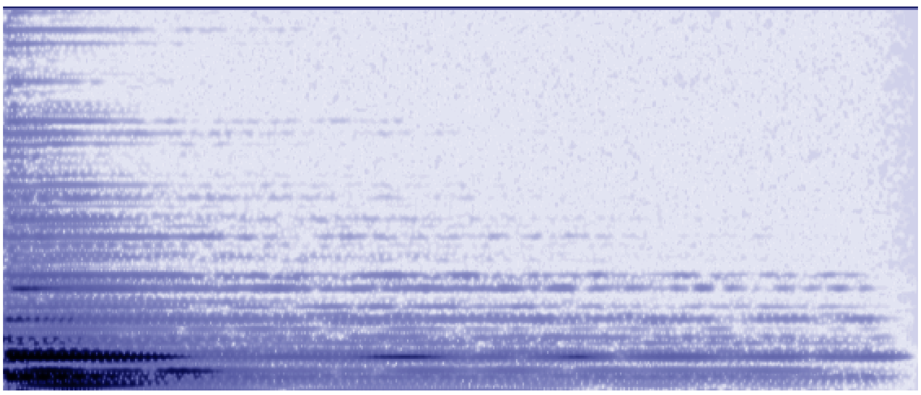

Figure 4: Windows asterisk sound spectrogram

The human ear cannot hear sounds that are directly adjacent to a frequency carrying a much stronger second noise. It follows from the above that this is another possibility to reduce the data after the transformation into the frequency domain. Also there are other effects: E.g. that humans are able to distinguish the volume of gentle sounds much more precisely than that of loud noises. The A/D converter samples every loudness with the same amount of bits, which isn't necessary.

The DVB standard uses MPEG1 audio layer 2, defined in ISO/IEC-13818-3. Sadly this standard could not be found freely available on the internet.

The audio codec itself is diagrammed in figure 5. Other than to the video codec, it doesn't make use of any predictor – this functionality appears only in layer 3, bettern known as MP3.

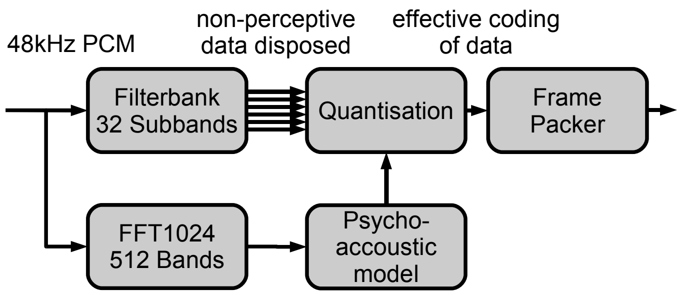

Figure 5: ISO/IEC-13818-3 audio encoder

The first step on the one hand divides the signal into 32 subbands and on the other hand transforms it into 512 frequency bands for further inspection. The 512 frequency bands are fed into the psychoaccoustic model which uses them to decide how strongly each of the 32 subbands is perceived and how many bits this perception is worth. In the following quantizing step the available bit budget is distributed into the subbands, which results in the independence of MPEG1-L2 audio frames: Each frame can be decoded without any knowledge of past or future frames. Apart from that all frames have the same length, i.e. they represent always the same number of original samples.

Finally, the frame packer adds a sync word to the compressed data preparing it for transmission.

3 Streamformat

Both video and audio codec were not developed with a special transport mechanism in mind, which is why they deliver a dataformat unfit for transmission: The video codec doesn't guarantee a fixed data rate, but only stays under a given maximum and it doesn't produce packets with a fixed length. The audio codec produces packets with a fixed length, but this length depends on the used datarate.

Additionally a single transmitter delivers datarates between 16 MBit/s (DVB-T) to over 50 MBit/s (DVB-C). This means that data coming from the codecs needs to be repacketized to gain a fixed packet size and to allow for multiplexing of serveral programs into a single common stream.

3.1 Packetizing

DVB uses a constant packet size of 188 Bytes for all data – a value that arises from using four ATM cells of 54 Bytes size together as a container, subtracting ATM header and ATM-adaption-layer. ATM is usually used as feeding network for DVB transmitters.

Figure 6 shows the frame format of DVB transport stream packets.

Every packet starts with the sync byte 0x47, which is needed for locking onto the stream when no additional synchronisation information is available.

The sync byte is followed by the actual transport stream header, which carries some flags bits and the payload ID (PID). Every service inside a transport stream is assigned a PID by the packetizer, whereas the PIDs 0x0000 to 0x003f serve special purposes. As a typical TS contains at least three

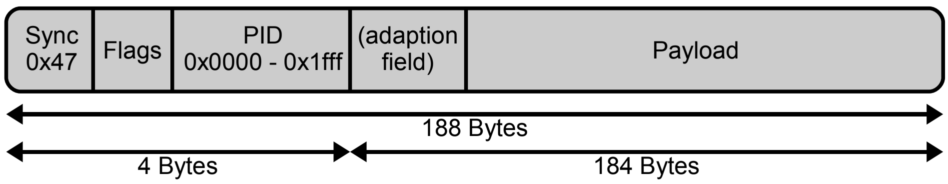

Figure 6: ISO/IEC-13818-1 Transport Stream packet header

or more TV programs, the demultiplexer in the receiver e.g. can use the two PIDs to find video and audio data of a specific program within the transport stream.

In addition to the normal TS header there can be an adaption-field in front of the actual payload. The existence of that adaption-field is signalled by one of the flag bits. Inside the adaption-field there are more flags which tell the receiver when there is a good point to start decoding after a program switch. Typically this is true when an I-frame is being sent.

Besides of that there is the continuity-counter, which allows the receiver to detect if a packet was lost or duplicated.

3.2 Contents of a stream

A MPEG2 transport stream carries a lot of ancillary data, too. These tell the receiver which programs are available with what PIDs, what names they have and what additional services like Teletext there are.

The following table shows the most important additional data streams:

Short name	Long name	Description
PAT (PID 0x0000)	Program Association Table	The PAT carries a list of all programs in this stream and a reference to the NIT
CAT (PID 0x0001)	Conditional Access Table	In the CAT data controlling pay-TV decryption is delivered.
NIT (PID 0x0010)	DVB Network Information Table	The NIT has information about the current transmitter network, additional frequencies and locations, where the same stream is transmitted
SDT, BAT (PID 0x0011)	Service Descriptor Table, Bouquet Association Table	Table with the program names and languages available in this stream
EIT (PID 0x0012)	Event Information Table	Contains the electronic program guide (EPG)
PMT (PID via PAT)	Program Map Table	Delivers information about a specific program; e.g. which PID is used for video, which for audio, etc. Every program has its own PMT.

These tables carry static data that is retransmitted every few hundred ms. If the receiver wants to get one of the tables, it has to wayt for a moment until a new copy can be found in the transport stream. Correspondingly the multiplexer has the job to ensure that every table is sent often enough.

3.3 Synchronisation

A last problem has to be solved before the transport stream can be sent on its journey: Since the audio encoder works with a constant bitrate and the video-codec is very variable and even can chose to send pictures earlier, a mechanism to synchronize the two is needed. The decoder has to know when to display which frame in order to be able to restore the correct order.

A possible solution for that problem is the one used by most software based MPEG2 players: A big data buffer is filled with stream data before decoding starts. When that buffer is full, a clock starts (commonly the soundcard clock) and is used to synchronize the rest of the system. At 48KHz audio samplerate every 1920 samples a new PAL frame has to be put onto the screen. That the sample clock is certainly not in sync with the transmitter can be compensated by delaying a frame once in a while or by dropping frames respectivly.

For set-top-boxes, which must not be more expensive than 20 Euros in manufacturing, this doesn't work: The needed buffer memory is so expensive alone that the only solution is to synchronize the box's crystal with the encoder.

To achieve that, a lot of information about the local encoder clock state is embedded into the stream: The program clock reference (PCR) helps adjusting the receiver's 27MHz-crystal to the transmitter – which in the first STB designs was done by literally adding a varactor diode parallel to the crystal.

Even more information is added to the stream: The packets carry decoding timestamps, which tell the decoder when a frame needs to be decoded to serve as a dependency for other frames (e.g. a P-frame that was sent earlier to be referenced by B-frames). Also a presentation timestamp is included which controls when a frame is actually displayed.

3.4 Multiplexer

Finally the multiplexer has the job to combine the different data streams, e.g. video and audio, but also the PSI tables as well as other services into a single transport stream so that all packets are available at the receiver when they are needed.

The multiplexer works according to the round-robin principle: Several inputs fill FIFO buffers and from every FIFO that has at least one packet waiting, one is taken at a time and sent out. But as the transmitter continues to run even when no data is available, these gaps are filled with NULL-packets which carry the PID 0x1fff and have a payload of all zeros.

Beyond that one small thing has to be managed by the multiplexer: When a packet is put into a FIFO, this means that the packet is going to be delayed a bit and the timestamps inside the packet will not be correct anymore when the packet reaches the decoder. Because of that the multiplexer has to add the time the packet stays in the FIFO to these timestamps.

When used for DVB-H the multiplexer also has to make sure that all streams are grouped together in a way that guarantees that the decoder has to enable it's receiver as rarely as possible. This safes power in the mobile or handheld device. But that is another complicated story which would go well beyond the scope of this paper.

4 Channel coding

The first step in preparing the multiplexed stream of 188 byte packets for transmission is the energy dispersion. Modulation and transmission works best if the transmitted data looks like random data. Regularities in the transmitted datastream lead to some symbols appearing more frequently than others in the modulation process, this in turn leads to an unbalanced spectrum where the energy is distributed unevenly. This is undesirable because an unbalanced spectrum can't utilize the whole dynamic range of the system. A DVB datastream is likely to contain a lot of regularities like zeroed padding packets. For that reason the packets' contents (not the sync bytes – 0x47 at the beginning of each packet) are xored with data from a pseudo random number generator (PRNG). Every eighth packet the sync byte is inverted (to 0xB8) and the PRNG is reset. This allows the decoder to synchronize it's PRNG with the encoder.

After the energy dispersal, a Reed-Solomon code is applied to each packet, enlarging it by 16 to a total of 204 bytes. This means that up to eight corrupted bytes can be corrected in each packet. This step may also be called the "outer coder". In DVB-T and DVB-S, there is a second layer of protection called FEC or "inner coder" which we will explain later. The Reed-Solomon code works on bytes – a corrupted byte counts as one error, no matter how many of it's bits are erroneous. Hence, the RS code is best suited to correct the block errors which are produced by the Viterbi decoder encountering too many errors on the inner protection layer (FEC). If – for example – the Viterbi (the inner) decoder goes out of sync and emits sixteen bits, those sixteen bits count as only two or three errors for the Reed-Solomon Â (the outer) decoder.

To make the data stream even more robust, it is then interleaved. This part may also be called the "outer interleaver" because in DVB-T there is a second interleaver, called the "inner interleaver". Each packet is interleaved with the preceding 12 packets so that the packet's first byte is not modified, it's second byte is assigned the value of the last packet's second byte, it's third byte is assigned the value of the third byte of the packet before the last one, and so on... The packet's thirteenth byte again is not modified and the interleaving procedure begins anew at this point. This goes for 17 times as 204 divided by 12 is exactly 17. Please note that in this procedure, the sync bytes (0x47 or 0xB8 at the beginning of packets) are not modified, so that the data still looks like a sequence of 204 byte packets afterwards. The interleaving in combination with the RS code enables the decoder to correct up to 8 times 12 times 8 (=704) corrupted bits in a row.

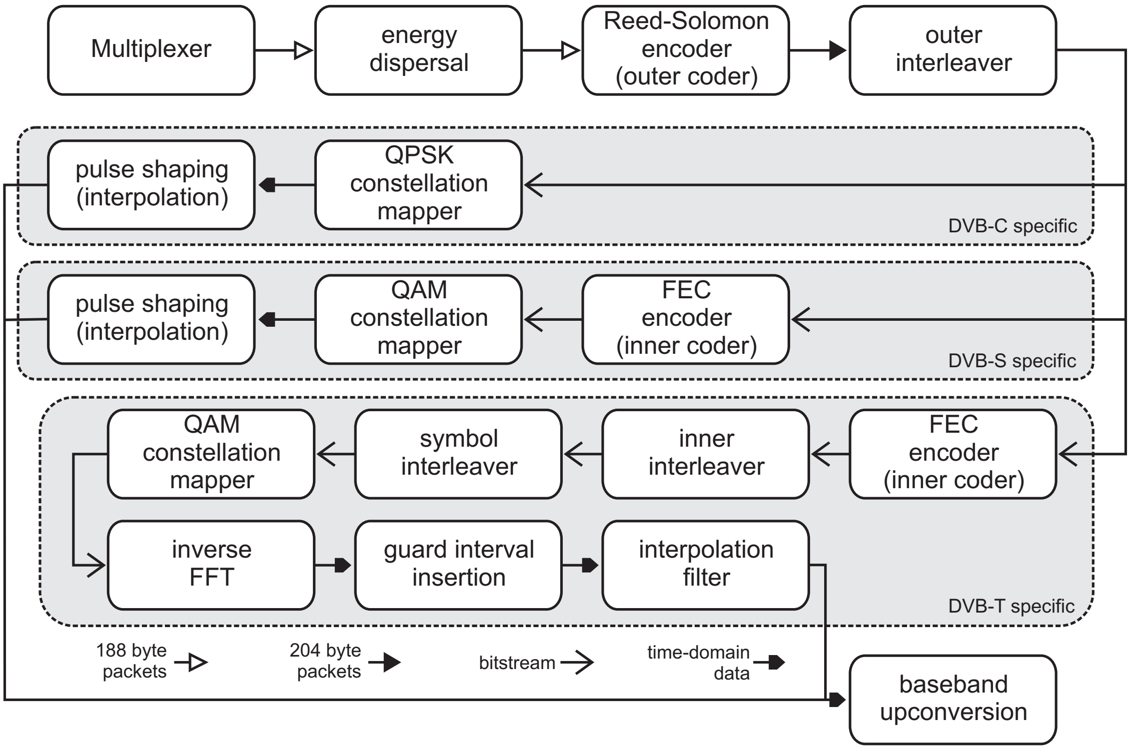

Figure 7: DVB modulator data flow

From this point onward, the datastream is seen as bitstream – all following operations are applied to single bits or symbols being composed of an arbitrary number of bits. This is also where we have to split up DVB-C – as a DVB-C signal is meant to be transmitted over a cable where noise and distortion is very low, the bitstream we have at this point can be mapped to QAM-symbols and transmitted without further treatment.

For DVB-S and DVB-T, which are to be transmitted over the air with noise, distortion and interference, another layer of error protection is added. A convolutional coder – also called the "inner coder" – outputs 2 bits for every bit put into it and blows the bitstream up by a factor of two. This bitstream can be decoded by a Viterbi decoder which works best if the corrupted bits are randomly distributed. This is convenient as the inner coder's main task is to remove noise (which is more or less random) from the signal in order to lower the necessary signal to noise ratio required for proper reception. This process is also called "forward error correction" (FEC). In this mode, the FEC eats up half of the available bandwidth – it's called a "1/2 FEC" because for every one bit of data, two bits are transmitted. If you have good reception and noise is low, you might not want to waste that much bandwidth for a FEC you don't need. In this case, you might just throw away some bits. If every fourth bit is thrown away, three bits are actually transmitted for every two data bits. This would be called a "2/3 FEC". The process of throwing away unwanted bits is called "punctuation". The receiver needs to know the pattern used to throw away bits and simply regards those bits as unknown or corrupted in the decoding process. Of course, this limits the receivers ability to correct real errors so that there is a trade-off between bandwidth spent on FEC and the signal level required for reception. The DVB standard allows 1/2, 2/3, 3/4, 5/6 and 7/8 FECs.

At this point, if you'd want to transmit a DVB-S signal, you could map the bitstream to symbols and transmit those symbols. For DVB-T the process is much more complex. Where for a DVB-S signal – transmitted from a satellite through lots of nothing and thin air – the only real problem is noise, a DVB-T signal suffers from reflections on houses and hills. A DVB-T receiver might in fact see multiple copies of the same signal, each copy delayed by a slightly different amount of time.

In DVB-T, a lot of bits are transmitted simultaneously on a great number of adjoined small carriers (1705 in 2k-mode or 6817 in 8k-mode) forming a signal about 8MHz wide. This whole block of carriers is also called an OFDM-symbol. To lower the effects of small banded interferences and fading, , the signal is interleaved a second time. This block is also called the inner interleaver. For this matter, the bitstream is split up into n separate bitstreams where n is the number of bits that each carrier in one OFDM-symbol carries (2 in QPSK-mode, 4 in QAM16-mode and 6 in QAM64-mode). Each of those bistreams is then interleaved separately in blocks of 126 bits.

Not all carriers available in an OFDM-symbol are loaded with a QAM-symbol to carry data. As the OFDM-symbol is 8MHz wide, transmission properties like phase shift, fading and distortion might vary greatly over that frequency range. To allow the receiver to compensate for those effects, some carriers are loaded with predefined QAMsymbols. Those carriers are called "pilots" and come in two varieties: continual and scattered. Continual pilots are at the same place with the same symbol in each and every OFDM-symbol. There are 45 or 177 continual pilots in each symbol in 2k or 8k mode. In addition, every twelfth carrier becomes a scattered pilot. This pattern is shifted by three carriers, with each OFDM-symbol. This means that in the first symbol carriers 0, 12, 24, 36,... become scattered pilots. In the second frame, carriers 3, 15, 27, 39 are scattered pilots, and so on... This pattern repeats itself at the fifth symbol. The pilots are boosted, which means that they are transmitted with slightly more energy than the other carriers. This makes them stand out in the spectrum and helps the receiver to lock onto the signal.

To decode the signal, the receiver needs to know a couple of parameters about the signal. However, the user wants to tune into a channel without knowing details like FEC-factor, constellation and transmission mode. For that reason, all important modulation properties are encoded into a 37 bit long structure. This structure is then padded with sync bits and error protection resulting in a total of 68 bits which are then called a TPS block. In each OFDM-symbol, a couple of carriers are reserved for the TPS (17 in 2k-mode an 68 in 8k-mode). In every OFDM-symbol, only one bit of the TPS block is transmitted by modulating this same bit on each of the TPS-carriers. This way, a whole TPS block is transmitted every 68 OFDM-symbols. Such a block of symbols containing exactly one TPS block is called an OFDM frame.

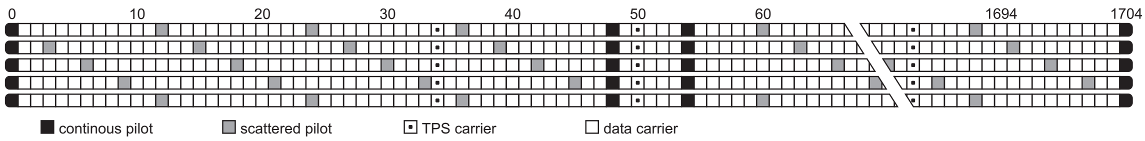

Figure 8: Carrier allocation for five consecutive DVB-T symbols (2k-mode)

After the insertion of pilots and TPS bits, 1512 of the 1705 carriers are left for the symbol's payload (6048 of 6817 in 8k-mode). The symbol interleaver then maps the n-tuples coming out of the inner interleaver onto those carriers. After this step, the time domain signal is generated by applying an inverse furier transformation. A IFFT algorithm with 2048 or 8096 points is used for that purpose. The output of a DVB-T transmitter is in fact the output of those inverse Fourier transformations stringed together, symbol by symbol. To protect each symbol from distortion which happens if the symbol gets mixed with delayed copies of the previous symbol reaching the receiver by reflection, a guard interval is inserted after each symbol. The length of the guard interval may be 1/4, 1/8, 1/16 or 1/32 of the symbol's length. During that time, the modulator starts over transmitting the symbol's signal from the beginning.

Four of the OFDM-frames, each containing 68 OFDM-symbols, are grouped together to build a superframe. In 2k mode with QPSK modulation and 1/2 FEC, such a superframe contains exactly 411264 data bits (this is the number of bits before the FEC is applied). Those bits make 51408 bytes, which make exactly 252 Reed-Solomon protected DVB-packets with a size of 204 each. The size of the OFDM frames and superframes is carefully chosen, so that the number of bytes in any allowed superframe always is a multiple of 204. The modulator has to make sure that the first byte in each superframe is the sync byte of a DVB packet, so that the packets are aligned within the superframes. In addition, all the pseudo random number generators, patterns and interleavers are reset with the beginning of each superframe. A superframe is therefore the biggest connected structure in a DVB-T signal. Each superframe can be decoded by itself, without knowledge of the previous or following superframes. This allows the receiver to start decoding the signal at the beginning of each superframe.

5 Modulation

If you want to impress information into an electromagnetic carrier wave, there are quite a lot of ways to accomplish this goal. The most obvious way would be to simply switch the carrier wave on or off. This is also what the first men who built transmitters did to modulate their information on the signal. They used a code known as the Morse code to encode their messages into a switched carrier wave. Since then, lots of different modulation techniques have been developed to satisfy a broad variety of requirements of different applications. However, it is important to know that in frequency domain, any modulation results in a spectrum that is more or less broader than the spectrum of the unmodulated carrier wave. This is a result of the fact that if you change any property of the carrier wave (phase, frequency, amplitude), you have to add additional frequencies to the spectrum in order to represent the changed signal. In time domain, modulation results in a signal that can be described by writing down it's amplitude and phase shift compared to the unmodulated carrier over time. This is what you see, when you look at a so called "I/Q diagram" or "constellation plot". At any given moment, the signal can be described by a point in that diagram. The amplitude of the signal is represented by the point's distance from the center while the phase is represented by the point's angle of rotation around the center of the diagram. If you'd watch such a signal in slow motion, you could see that point wandering around in the diagram.

You can use this technique to pack information on your carrier wave by encoding bits in the position of the point at a predefined time. You could for example say that in your signal, the point is either top left, top right, bottom left or bottom right in the diagram. This diagram with the point somewhere in it is called symbol. This way, you could transmit 2 bits per symbol. This is how DVB-S works – it is called Quadrature Phase Shift Keying. "Quadrature" because the process of using phase and amplitude to encode information is called "Quadrature Modulation", "Phase Shift Keying" because in this case only the phase is changed – all four points used to encode data have the same distance from the center, only their angle is changed. The amount of information you could transmit per second of course depends on how often you change that symbol. In DVB-S, common symbol changing frequencies are 22 MegaSymbols/s or 27.5 MegaSymbols/s. The resulting spectrum has a bandwidth of about 35 MHz.

In German cable networks, the bandwidth available for each carrier is limited to 8 MHz. Therefore, the symbol rate for DVB-C is limited to about 7 MSymbols/s. To archive a data rate similar to DVB-S, more information needs to be carried by each symbol. To archive that, more different points on the I/Q diagram are used to encode bits. DVB-C uses 16, 32, 64, 128 or even 256 different points, to encode up to eight bits per symbol. This kind of modulation is called Quadrature Amplitude Modulation (QAM) – the number of points used is often given as number after the acronym like in "QAM128". The QPSK modulation which is used for DVB-S could also be called QAM4. Of course, using more positions makes the signal more vulnerable to noise as is becomes much easier for a point to be moved to a different position by a bit of noise.

A problem with that kind of modulation is, that the individual symbols are extremely short.

If multipath interference occurs, a copy of the signal which took a detour of 10 km (perhaps by bouncing of a small hill nearby) is delayed enough to mix each symbol with the previous one at the receiver which would make the signal completely un-useable. While this is fine for DVB-S and DVB-C, it is unacceptable for terrestrial transmission, especially in urban areas. One way to make the symbols longer would be to use even more points per symbol – something like QAM1024 or QAM4096. In practice, this would not work - the signal would be too vulnerable to noise.

DVB-T is using a rate of only about 4000 symbols per second in 2k-mode. But instead of one single carrier, 1705 carriers are used simultanously, each carrier having it's own I/Q diagram carrying a couple of bits. Because the symbol rate is low, the indiviudal carriers are small, so that all 1705 of them fit into a band only 8 MHz wide. The spacing between the carriers needs to be choosen carefully to prevent them from interferring whith each other – they need to be orthogonal. This method of modulation is called Orthogonal Frequency Division Multiplexing or Discrete Multitone Modulation.

6 Conclusion

To answer the question from the beginning: Yes, it is possible to build a homebrew DVB transmitter – but it needs parts and knowhow that are not commonly available in any home workshop. Luckily now that the hardware is available for less than 1000 Euros, more experimentation on the protocol level should be possible. One application of this could be the replacement of the outdated Packet Radio network run by the radio amateurs. One sad thing remains: We cannot open the FPGA firmware since chinese companies are already on the verge of building rogue copies – but we're looking for a way to enable people to add more stuff to the FPGA without the need for the modulator source. The solution here could be the IPcore supporting tools Xilinx has embedded into its development environment.

7 Thanks and Greetings

- Stefan Reimann, DG8FAC, who gave us a new an all new problem to chew on – and who had trust in us that we wouldn't break his expensive equipment.
- Andreas Koch, DG5FX, who brought us into contact with Stefan and hoped, we could help him with the FPGA stuff.
- Harald Welte, who, once again, encouraged us to do a lecture at the Chaos Communication Congress.
- Sabine Pichler, who in her capacity as English teacher, helped us catch the typos.
- Thanks also to our beloved girlfriends Teresa and Sabine, who make sure that we have a warm bed when finally finishing work at 4am...

References

[1] `http://www.dvb.org`

[2] `http://inst.eecs.berkeley.edu/~cs150/Documents/ITU656.PDF`

[3] `http://en.wikipedia.org/wiki/YUV`

[4] `http://le-hacker.org/hacks/mpeg-drafts/is138182.pdf`

[5] `http://www.bbc.co.uk/rd/pubs/papers/paper_14/paper_14.shtml`

[6] `http://en.wikipedia.org/wiki/Discrete_cosine_transform`