IMPLEMENTING THE TETRA PHYSICAL LAYER ON LYRTECH'S SFF SDR DEVELOPMENT PLATFORM

Stefan Nagel, Dennis Epple, Friedrich K. Jondral
(Institut für Nachrichtentechnik, Universität Karlsruhe (TH), Germany,
{nagel;epple;fj}@int.uni-karlsruhe.de)

ABSTRACT

This paper presents the implementation of the TETRA physical layer on an FPGA/DSP based development platform. The objective is to implement a highly portable waveform and to determine the optimal boundary between FPGA-functions and DSP-functions. Due to the portability aspect, we tried to implement as much functionality as possible on the DSP and followed a design process proposed from the Model Driven Architecture. The Lyrtech Small Form Factor SDR Development Platform is used as the hardware platform. It is equipped with a Virtex-4 SX35 FPGA from Xilinx and a TMS320DM6446 DSP system-on-chip from Texas Instruments.

TETRA SYSTEM OVERVIEW

Terrestrial Trunked Radio (TETRA) is an open digital standard defined by the European Telecommunication Standards Institute (ETSI). Its infrastructure is primarily targeted at the mobile radio needs of public safety groups like police and fire departments [2].

For modulation a π/4-Differential Quaternary Phase Shift Keying (DQPSK) scheme is used which provides a symbol rate of 18 kbaud/s. TETRA uses different forward error correction schemes applied to the different logical channels. For the Traffic Channel (TCH) a Rate Compatible Punctured Code (RCPC) is employed while the Access Assignment Channel (AACH) uses a shortened Reed Muller (RM) code for example. The channel access mode is a combination of FDMA and TDMA. Each channel of bandwidth 25 kHz is divided into 4 timeslots. For Uplink/Downlink separation an FDD/TDD mode is implemented. The time delay between uplink and downlink is about two time slots, so the mobile station does not have to send and receive data simultaneously. Within one time slot of each carrier frequency a normal burst is transmitted. It is used for the transmission of voice or data and contains 432 data bits and 78 bits for training, synchronization, guard period, etc. So the overall length of a burst is 510 bits that are transmitted within 14.17 ms. Four bursts fit into one TDMA frame which results in a frame duration of 56.67 ms. [1]

PLATFORM OVERVIEW

Lyrtech's Small Form Factor SDR Development Platform is a heterogeneous platform for signal processing that consists of three modules: the RF Module, the Data Conversion Module and the Digital Processing Module. The platform has further the possibility to interconnect to MatLab/Simulink and to develop waveforms on a high level view.

RF Module

The RF Module is the RF frontend of the SDR. The receive path is realized with a super heterodyne receiver, that mixes the signal from a range of 20… 928 MHz to an intermediate frequency of 30 MHz. The analogue receive filter can be chosen to support a bandwidth of 5 MHz or 20 MHz. The complex signal is transported to the Data Conversion Module via coax cable. The transmitting path of the RF Module expects two inputs from the Data Conversion Module: the inphase and quadrature component. These are directly mixed to the desired carrier frequency from 200 MHz to 930 MHz. The RF Module is further equipped with a 10 MHz oscillator which provides the reference clock for the whole platform.

Data Conversion Module

The Data Conversion Module is the interface between the analogue and the digital side of the platform. It is equipped with two DACs and two ADCs, with only one ADC being connected to the RF Module. The DAC works with a sample rate of 500 MSPS and a resolution of 16 bit. The two DACs are used to convert the inphase and quadrature component independently from one another. The ADC works with a sample rate of 125 MSPS and a resolution of 14 bit.

Digital Processing Module

The Digital Processing Module is equipped with an FPGA, a DSP and a GPP. The FPGA is the Virtex-4SX35 with

34,560 Logic Cells, 30,720 Flip Flops and 192 DSP Slices. The GPP and the DSP are both on a common chip which is the DaVinci SoC from Texas Instruments. The core of the DSP is a C64x+ with 594 MHz and the core of the GPP is the ARM926 with 297 MHz.

Design Flow:
The design flow starts with building a Simulink model of the FPGA and the DSP [4]. This can be executed on PC until the simulation works properly. For executing the software on the hardware further development tools are needed. For the DSP, the Code Composer Studio transforms the Simulink model into a binary file for the processor (.out file). To build the FPGA model in Simulink there is need for special Simulink blocks which are equipped with the System Generator for DSP. This software builds the interface between Simulink and the Xilinx Development Environment and transforms the model in the bitfile. The design flow and the three modules are shown in Figure 1.

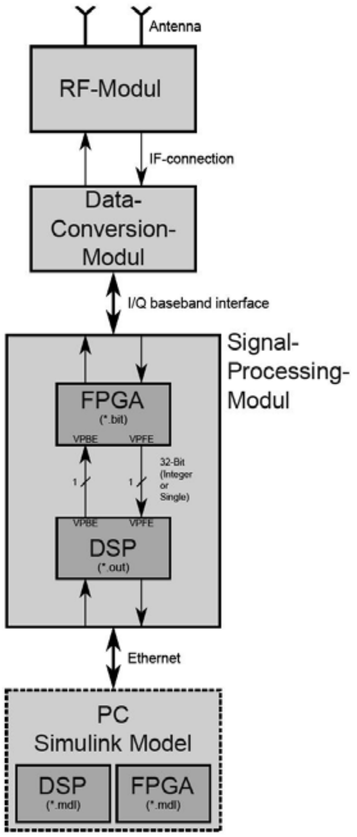

Figure 1: Structure of the SDR platform and the design flow

WAVEFORM DEVELOPMENT

One objective of this work was to get a high portability of the TETRA-waveform and to evaluate if modules could be reused. We followed a waveform based development process based on the Model Driven Architecture (MDA) which is an initiative of the Object Management Group (OMG) [3]. It introduces four different types of models:

1. The Computation Independent Model
2. The Platform Independent Model
3. The Platform Specific Model
4. The Code

These models are levels of abstraction of a waveform. By implementing all these models there is an evolution from a very generic waveform specification to code which can be executed on hardware.

The Computation Independent Model (CIM) is just a description of the functionality. In our case, this is the specification of the TETRA air interface, i.e., the ETSI standard.

The next model is a Platform Independent Model (PIM) that provides the functionality without platform specific aspects. The PIM was created by implementing a MatLab/Simulink model of the TETRA physical layer.

The subsequent step towards a waveform running on a platform is the design of a Platform Specific Model (PSM). This includes platform specific aspects like the connection from the FPGA to the DSP, human machine interfaces, etc. To transform the PIM to the PSM we followed the model based design flow proposed by Lyrtech. Due to that, we replaced the generic blocks in the model by target specific blocks. We additionally added platform specific blocks that are interfacing the I/Os on the board.

The final step in the MDA is the implementation of Code that runs on the specific platform. The development environment generates code from the PSM and sends it to the processor's Integrated Development Environment (IDE) that will compile the generated code and load the executables to the processors.

Development of the CIM:
The CIM actually exists as the ETSI standard for the TETRA AIR interface. Unfortunately this is a very large and extensive standard so we decided to simplify it for our development.

There are seven different bursts defined in the specification so we decided to implement just the Normal Continuous Downlink Burst. This burst includes a Traffic Channel for the data supporting a bit rate of 4.8 kbit/s (TCH/4.8) and a Control Channel (CCH) that manages the allocation of the next up- and downlink slots.

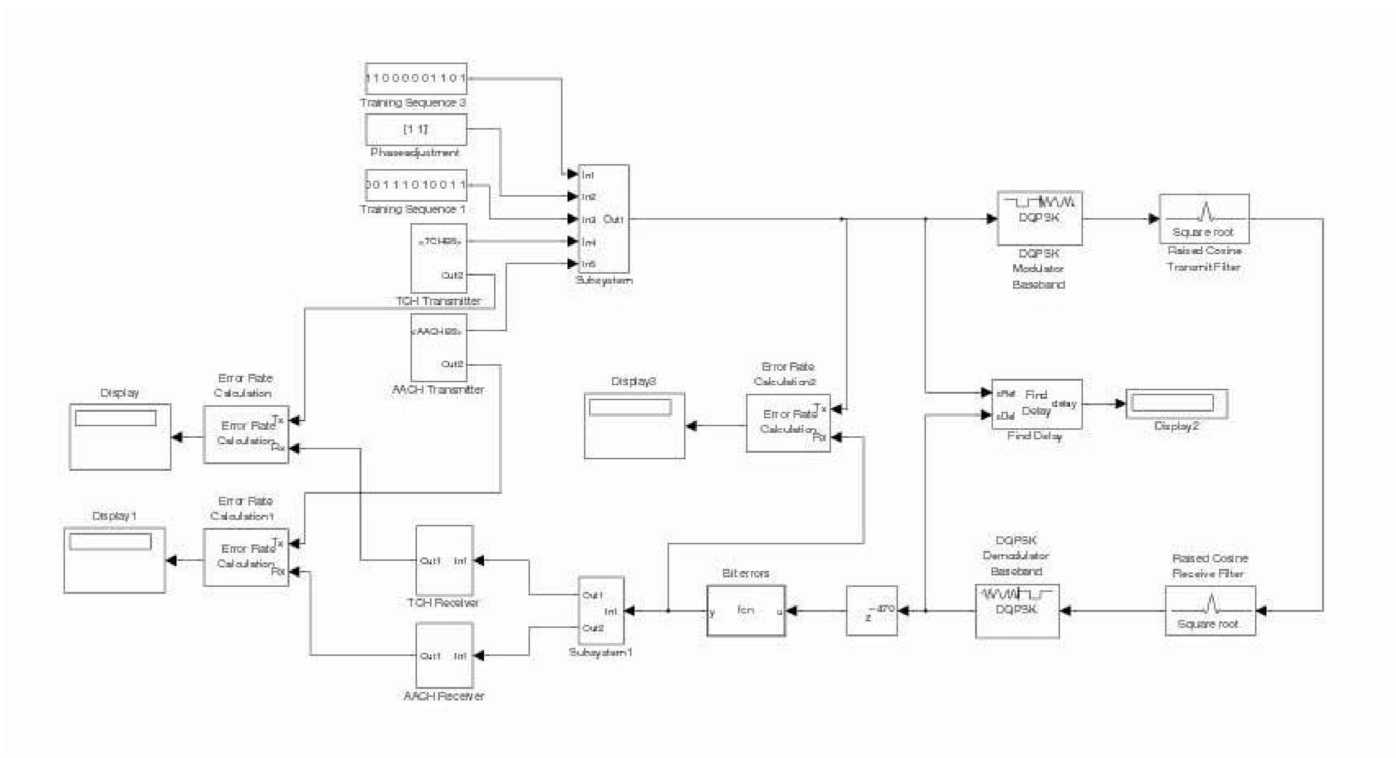

Figure 2: Platform Independent Model

Development of the PIM:

The TETRA transmit burst we implemented consists of the Normal Continuous Downlink Burst. This burst is based of 510 bits that are represented in Table 1.

Table 1: Normal Continuous Downlink Burst

Bit Number	Field length	Field content
1 to 12	12	normal training sequence 3
13 to 14	2	phase adjustment bits
15 to 230	216	scrambled block 1 bits
231 to 244	14	scrambled broadcast bits
245 to 266	22	normal training sequence
267 to 282	16	scrambled broadcast bits
283 to 498	216	scrambled block 2 bits
499 to 500	2	phase adjustment bits
501 to 510	10	normal training sequence 3

The 30 broadcast bits come from the Control Channel which manages the allocation of the next up- and downlink slot. They consist of 14 information bits that are coded with a Reed Muller code and afterwards scrambled.

The 432 scrambled bits are the bits from the Traffic Channel which transfers the user data. The two blocks consist of 288 information bits plus 4 tail bits that are encoded using a convolution code. The puncturing after the encoding provides the 432 coded bits that are afterwards scrambled.

The remaining 48 bits are training sequences and bits to adjust the phase.

The frame is sent to the DQPSK modulator that converts the bits to symbols and afterwards filtered by a root raised cosine transmit filter.

The receiver filters the signal with a root raised cosine filter and demodulates the incoming symbols to bits. The 432 coded bits are separated from the frame by dropping the trainings bits. Decoding is done with a Viterbi Decoder for the Transmit Channel and a Reed Muller Decoder for the Control Channel.

By developing the PIM we provide a starting point from which any waveform developer could port the TETRA-waveform on any Software Defined Radio platform.

Development of the PSM:

Configuration:

In the transformation from the PIM to the PSM the clock and sample times have to be set. A time slot of the TETRA system has a duration of 14.167 ms in which 255 symbols are transmitted. By supporting just the DQPSK modulation this leads to a bit rate of 36 kHz as the fastest rate in the DSP. The ADC clock can be varied from 80 to 125 MHz. So we chose an ADC clock of 108 MHz. By providing a clock divider of 3000 the DSP bit rate can be provided.

After setting the clock rate the partition between the DSP and FPGA has to be set. Our intention was to get a highly portable waveform, so there was the intention to bring as much as functionalities onto the DSP as possible. The optimum would be to develop the whole waveform on the DSP and just send the I/Q components to the DAC and vice versa. We decided to stay very close at this optimum. So the signal processing from the data bits to the generation of the phase is done in the DSP and the further signal processing like the generation of the inphase and quadrature component and the up-sampling is done in the FPGA.

DSP Transmit Side:

The TETRA transmit burst is generated as described in the Platform Independent Model. The symbol generation maps two bits to a symbol which is afterwards used by the Differential Preencoding to generate the proper phase. The actual symbol S(k) is obtained by rotating the phase of the last symbol S(k-1) by the phase Dθ(k):

$$S(k) = S(k-1)e^{jD\theta(k)}$$

The phase shift is related to the modulation bits as described in Table 2.

Table 2: Phase shift in π/4 DQPSK modulation

S(k)	Dθ(k)
00	+ 1/4 π
01	+ 3/4 π
10	- 1/4 π
11	- 3/4 π

The output of the Differential Preencoding block is the phase of the symbol in the range from 0 to 2π. The phase has to be converted in the proper way by:

$$\varphi_{FPGA} = \varphi_{DSP}\frac{2^B}{2\pi}$$

B is the amount of bits provided by the Direct Digital Synthesizer (DDS) in the FPGA. Before transferring this value to the FPGA it has to be converted in the proper representation due to the fact that the VPBE bus requires an

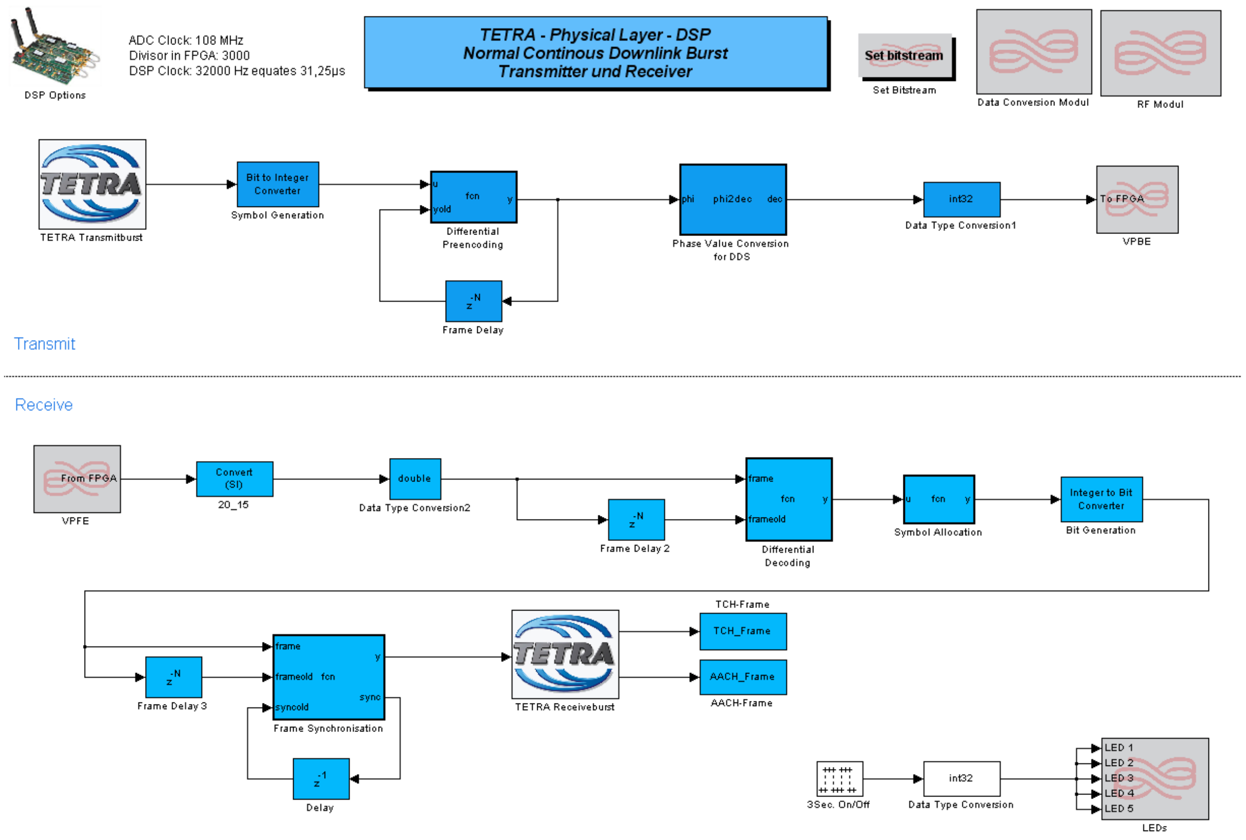

Figure 3: DSP part of the Platform Specific Model

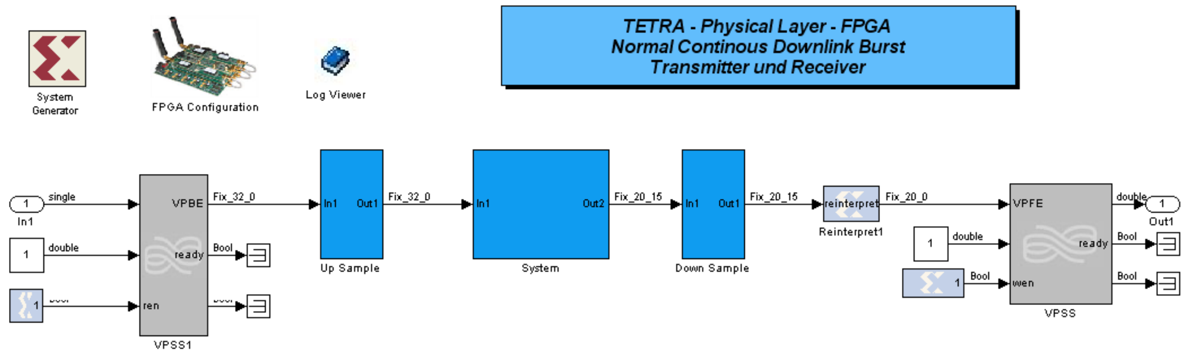

Figure 4: FPGA part of the Platform Specific Model

integer format (int32).

DSP Receive Side
The data coming from the VPFE bus are represented by an integer format (int32), although the FPGA works with a fixpoint representation. To get the information back to floating point representation there is need to drop the last 20 bits from the received value and set the point to the 15th bit. After a conversion to the double representation, the well known work with Simulink can be continued.
The differential decoding is done by subtracting the former phase from the previous one with help of a delay block. To map the phase to a symbol Table 3 is used:

Table 3: Mapping the phase to the symbols

Range	Phase	Symbol
[0...π/2]	+ 1/4 π	00
[π/2...π]	+ 3/4 π	01
[0…-π/2]	- 1/4 π	10
[-π/2…-π]	- 3/4 π	11

Before dividing the burst into the coded bits and the training sequence there is need to find the head and the tail of the frame. So the first 12 bits from the burst are used for frame synchronization. This algorithm is:

- Take two consecutive frames
- Connect them to a frame with 1020 bits
- Search for the training sequence
- The synchronized frame consists of the training sequence and the adjacent 498 bits
- Drop the first frame and take a new one
- Start over

The TETRA receive burst block performs like the PIM in Figure 2. The whole DSP part of the PSM is shown in Figure 3.

FPGA High Level Structure
Figure 4 shows the high level view of the FPGA structure. The input is the VPSS block which is the bus from the DSP. The signals are acquired with a rate of 36 kHz and then up-sampled by a factor of 3000 to a sampling rate of 108 MHz. The same is done on the receiving side accordingly. The signals from the system block are down-sampled with a factor of 3000 and sent to the DSP via the VPSS block with a sample rate of 36 kHz.

FPGA System
The FPGA System is shown in Figure 5. The modulation block requires phase information coming from the DSP and generates a complex signal with the DDS. The frequency in the DDS is 18 MHz that is used as an intermediate frequency.
The channel filtering is performed by FIR-filters extracting the channel of 25 kHz. The pulse shaping is done by a root raised cosine filter with a roll-off-factor of 0.35. This is predefined in the TETRA standard. This filter was implemented as an FIR filter with 5 taps.
In order to transfer the data to the FPGA, the fixpoint representation with 20 bits has to be converted to 16 bits due to the DAC's resolution.
Further data processing is not shown in Simulink because the DAC is the last block for the FPGA. Nevertheless, data

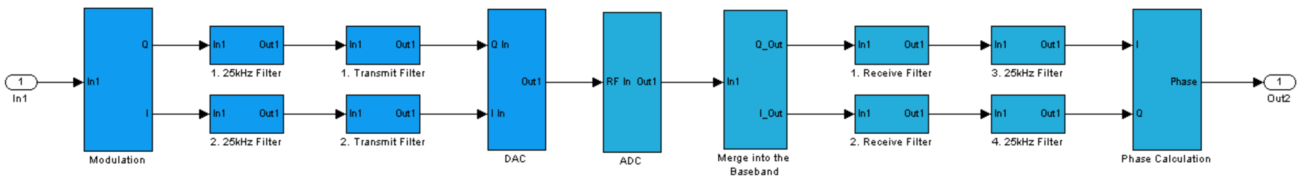

Figure 5: FPGA system

processing can be configured by APIs in the DSP. In Figure 3 in the upper right corner there are the blocks that represent the APIs to configure the ADCs or the RF-Frontend.

The RF Module receives the signal at 400 MHz and translates it to an intermediate frequency of 30 MHz. This signal leaves the ADC. The mixer-block brings the signal down to the complex baseband by multiplying the signal with a sine and a cosine at a frequency of 30 MHz. The sine and cosine is generated with the DDS-block. After filtering the signal with a bandpass filter the phase has to be calculated. This is done in the Phase Calculation block with the CORDIC algorithm. It requires the inphase and quadrature components of a signal and calculates phase and magnitude. Due to the fact that the whole information is incorporated in the signal phase, the magnitude is dropped and only the phase is provided to the DSP.

DEVELOPMENT OF THE CODE:

The Code was generated automatically by using the Code Composer Studio for the DSP code and the System Generator for DSP with the ISE for the FPGA bit file We utilized 3,935 Slice Flip Flops that made 12% of the total amount and 4375 Look Up Tables that made 14% of he total amount, available on the FPGA. The utilization of DSP48 blocks was 19% of the total amount which made a usage of 37.

CONCLUSION:

We developed the Traffic Channel of the TETRA physical layer on Lyrtech's SFF SDR Development Platform with the design flow of the Model Driven Architecture. In order to simplify our model, we chose representative parts of the standard in order to build a Computation Independent Model. The development of the Platform Independent Model was done in MatLab/Simulink. This model can serve as a starting point for any waveform developer to get the TETRA Traffic Channel on any Software Defined Radio platform. By developing the Platform Specific Model for our platform we added special interfaces and the partition between DSP and FPGA functionality. Due to the platform architecture it was not possible to implement the whole waveform on the DSP. The FPGA had to provide inphase and quadrature components, up- and down-conversion and filtering. Finally the development of the code was done automatically by the tools provided by the processor vendors.

The advantage of such a design flow is time. The whole implementation was done in about three months. So this is a good way for rapid prototyping. Another advantage is the reusage of our models. We can implement this model on any other DSP with minor changes in the I/O.

The disadvantage of this design is the insufficient comprehension of the Simulink blocks. We never knew what is really going on in the blocks, how they are implemented and how much performance is lost. To evaluate this, we are going to implement a new PSM in C without code generation and compare this to our TETRA waveform.

REFERENCES

[1] ETSI, *"Terrestrial Trunked Radio (TETRA); Voice plus Data (V+D); Part 2: Air Interface (AI),"* EN 300 392-2, 2007.

[2] B. Walke, „*Mobilfunknetze und ihre Protokolle 2*", Teubner, Stuttgart, 2003.

[3] T. Langguth and D. Schober, "SDR based Waveform Development," *5th Karlsruhe Workshop on Software Radios*, 2008.

[4] R. Sathappan, M. Dumas and M. Uhm, "A new architecture for development platforms targeted to portable radio applications," *Lyrtech Technical Paper*, 2007.