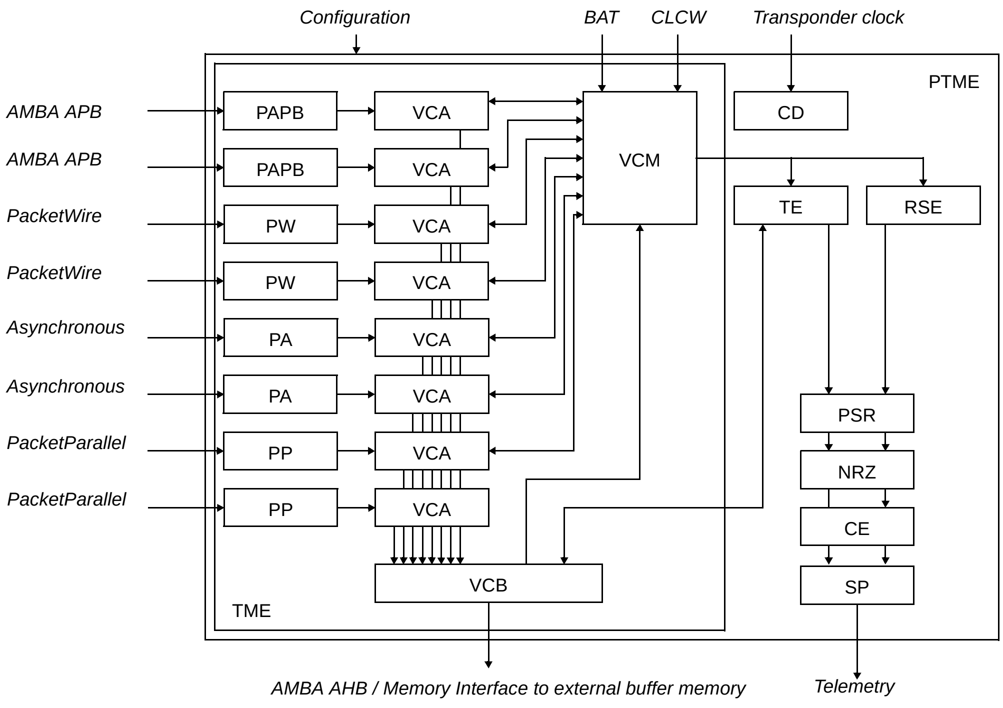
PTME

Packet Telemetry Encoder

Synthesizable VHDL Model - Product Sheet

The Packet Telemetry Encoder (PTME) synthesizable VHDL model comprises several encoders and modulators implementing the Consultative Committee for Space Data Systems (CCSDS) recommendations and the European Space Agency (ESA) Procedures, Standards and Specifications (PSS) for telemetry and channel coding. The European Cooperation for Space Standardization (ECSS) documents will be based on the CCSDS recommendations, permitting the PTME to be used in future projects. The PTME is being used in two ongoing ESA satellite projects and in several microelectronics developments. The PTME model is based on the knowledge and experiences from the preceding VCA, VCM and RESCUE standard devices.



Features:

- Packet Telemetry Encoder (PTME)
- Telemetry Encoder (TME)
- Reed-Solomon Encoder (RSE)
- Turbo Encoder (TE)
- Pseudo-Randomiser (PSR)
- Non-Return-to-Zero Mark encoder (NRZ)
- Convolutional Encoder (CE)
- Split-Phase Level modulator (SP)
- Clock Divider (CD)

Compatibility:

- Packet Telemetry, CCSDS 102.0-B-5
- Packet Telemetry Standard, ESA PSS-4-106
- Telemetry Channel Coding, CCSDS 101.0-B-5
- Telemetry Channel Coding Standard, ESA PSS-04-103
- AMBA^{TM} Specification, Rev 2.0, ARM IHI 0011A

PTME-003-01
Version 0.0
March 2002

Stora Nygatan 13
411 08 Göteborg
Sweden

tel +46 31 802405
fax +46 31 802407
www.gaisler.com

1 PACKET TELEMETRY ENCODER (PTME)

The Packet Telemetry Encoder (PTME) VHDL model can be configured at compile or synthesis time to include various encoding functions. The resulting instantiation of the design can then be configured during operation to enable the use of implemented encoders. The PTME model instantiates all encoder and modulator modules described hereafter. It propagates all interface signals to the relevant embedded modules, with few signal modifications, allowing control and observability of the embedded encoders and modulators. It also provides the interconnection between the encoders and supports all permissible coding chains.

1.1 Implementation options at compile time

The number of Virtual Channels to be implemented is selectable between one and eight. The Virtual Channel Identifiers are assigned automatically. It is possible to generate Idle telemetry Transfer Frames on any of the implemented channels or as a separate Virtual Channel on which no user data can be transmitted.

The following parameters can be set collectively for all Virtual Channels:
- support for alternative Attached Synchronisation Marker
- support for Secondary Header generation, ESA PSS Version-1
- support for Operation Control Field (OPCF) generation
- support for Frame Error Control Word (FECW) generation
- support for time strobe generation

Several parameters control the implementation of each individual Virtual Channel and their interfaces. The following parameters are configurable at compile time:
- packet handling, generating the First Header Pointer dynamically
- ready-to-receive a segment/packet indication
- Idle Source Packet insertion

The different Virtual Channels share a common external buffer memory in which data received on the user interfaces are temporarily stored. The sizing and splitting of the external buffer memory between the Virtual Channels is fixed to some extent at compile time. The following parameters are set: the overall memory size, the number of areas into which the memory is split, the number of areas that can be allocated to a Virtual Channel.

1.2 Telemetry Encoder (TME)

The Telemetry Encoder (TME) can be configured to support any number of Virtual Channels. All Transfer Frame sizes are supported, with the inclusion of all optional fields. The encoder supports all permissible packet formats:
- Source Packet (Version Number 000_b - Version-1)
- CCSDS Network Protocol (NP) Datagram
- Internet Protocol Datagram (IPv4)
- Encapsulation Packet
- Source Packet (Version Number 100_b - Version-2)
- Telemetry Packet (Version Number 100_b - Version-2)

The encoder can optionally generate and insert Idle Source Packets for each Virtual Channel to fill up an incomplete Transfer Frame Data Field. This ensures that user data do not remain inaccessible due to an incomplete Transfer Frame being resident in the external buffer memory. When there is insufficient data available from the implemented Virtual Channels to complete a Data Field, the encoder generates an Idle Transfer Frame on the fly. The Idle Transfer Frame Data Field need not to contain any useful data and is filled with a pseudo-random bit sequence.

The encoder provides two built-in algorithms for selecting which Virtual Channel to output in the next Transfer Frame. The Bandwidth Allocation algorithm guarantees a minimum bandwidth for each channel. In this algorithm the encoder uses adaptive channel ordering to utilize the available bandwidth as efficiently as possible. In the Priority Selection algorithm the encoder selects the highest priority Virtual Channel that has data ready for transmission.

The encoder provides several different input interfaces to the Virtual Channels. The PacketWire (PW) interface is a simple bit synchronous protocol. Data can either be packets or a bit stream. The PacketParallel (PP) interface is an asynchronous memory type interface. Data are written by means of a write strobe. The PacketAsynchronous (PA) interface is a bit asynchronous protocol used for receiving data only. The PacketAPB (PABP) interface is compliant with the AMBA APB interface specification.

All Virtual Channels share a common external buffer memory. The access bandwidth to the memory is dynamically allocated, giving an equal amount of guaranteed bandwidth to the different channels. The arbiter scans requests from all channels in parallel to decide which channel will be granted an access in the next slot. Since an unused slot is assigned to the next entry in a fixed round robin table, no bandwidth is wasted. This also allows sharing of the bandwidth between different channels. The interface to the external buffer memory can either be an asynchronous memory interface or an AMBA AHB master interface that can be used for system-on-a-chip solutions. The memory interface supports a single external memory device for data storage and provides Error Detection And Correction (EDAC) capabilities.

1.3 Reed-Solomon Encoder (RSE)

The Reed-Solomon Encoder implements two encoding schemes. The ESA standard specifies the E=16 (255, 233) code. This code is also specified in the CCSDS recommendation, which in addition specifies the E=8 (255, 239) code.

1.4 Turbo Encoder (TE)

The Turbo Encoder encodes data from the preceding Telemetry Encoder according to CCSDS recommendations. It implements all possible information block lengths and code rates.

1.5 Pseudo-Randomiser (PSR)

The Pseudo-Randomiser generates a bit sequence which is combined with the data output of preceding encoders. This function allows the required bit transition density to be obtained on a physical channel in order to permit the receiver on ground to maintain bit synchronisation.

1.6 Non-Return-to-Zero Mark encoder (NRZ)

The Non-Return-to-Zero Mark encoder differentially codes the data from preceding encoders.

1.7 Convolutional Encoder (CE)

The Convolutional Encoder implements two convolutional encoding schemes. The ESA standard specifies a basic convolutional rate 1/2 code without puncturing. This basic convolutional code is also specified in the CCSDS recommendation, which in addition specifies a punctured convolutional code. All puncturing rates are supported.

1.8 Split-Phase Level modulator (SP)

The Split-Phase Level modulator modulates the data bit stream from preceding encoders.

1.9 Clock Divider (CD)

The Clock Divider provides clock enable signals for the telemetry and channel encoding chain in Packet Telemetry Encoder (PTME), generating the desired bit and symbol rates.

2 PTME VHDL SOURCE CODE

The Packet Telemetry Encoder (PTME) model is written in synthesizable VHDL, currently targeted towards the Synplify synthesis tool from Synplicity, but is also compatible with the Synopsys Design Compiler. The PTME is an almost fully synchronous design based on a single system clock strategy. The asynchronous parts are related to the interfaces. The model and testbenches are written according ESA recommendations. The code complies to VHDL'93.

2.1 Packages and libraries, interface port and generic types

The following VHDL packages are used in the PTME VHDL model: *Std.Standard*, *IEEE.Std_Logic_1164* and *IEEE.Std_Logic_Arith*. Most PTME model interfaces use the *UnSigned* array type. The complete PTME model is configured by means of a package containing all configuration constants used in the design. There are no generics used for the top entity in the PTME model.

2.2 Simulation

A testbench is provided with the PTME VHDL model which can be used to set up verification campaigns. The output is in ASCII text format and can be processed off line to verify the output of the encoders. The testbench is configured with the same configuration package as the PTME VHDL model. The testbench will thus adapt itself to the selected PTME configuration.

3 AVAILABILITY

The PTME VHDL model has been developed by the European Space Agency (ESA) and Gaisler Research. Development and integration support can be obtained from Gaisler Research. The source code can be purchased from the ESA, at *www.estec.esa.nl/microelectronics/core*