CAN-CTRL

CAN Controller Core

For CAN 2.0B and CAN FD

Test Verification Document

February 2019

IP Product Version
7x06N00S00

Document Signature
CAN-CTRL–VES–7x06N00S00–111

CAST, Inc.

CONFIDENTIAL

This core Engineered by:
Fraunhofer IPMS

Document Version

Table 1-1 Document Version History

Version	Date	Person	Changes from Previous Version
100	2016/04/26	R.H.	First release
101	2016/04/27	R.H.	Wishbone Interface added
102	2016/05/17	R.H.	List oft tested transceivers extended, CANoe test description extended
103	2016/06/08	R.H.	List oft tested transceivers extended,
104	2016/06/10	R.H.	AMBA APB testbench added to the release package
105	2016/08/26	R.H.	Wording improvements
106	2016/09/19	R.H.	AMBA AHB testbench added to the release package
107	2016/10/17	R.H.	Chapter for Avery VIP added
108	2017/02/03	M.Z.	Code coverage added
109	2017/12/20	M.Z.	Linux test system added
110	2018/03/26	R.H.	LINT documentation
110	2018/10/24	R.H.	Synopsys DC removed from supported tools
111	2019/02/06	R.H.	Note about input synchronizers and the VIP added

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Purpose of the document

This document describes the methods that are used to verify the correctness of the CAN / CAN FD protocol IP core CAN-CTRL. In addition it provides an architectural overview of the verification environment.

1.2 Functional Overview

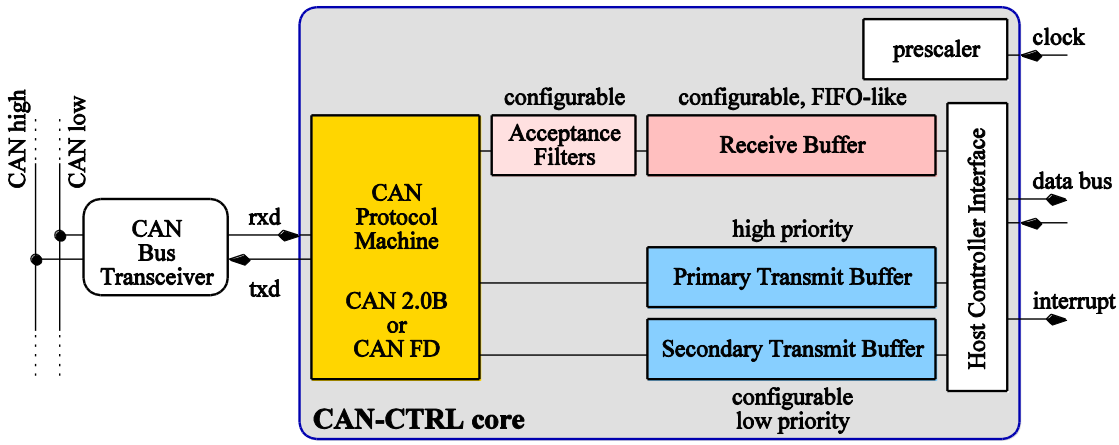

Figure 1-1 Connection to CAN Bus and Main Features of the CAN-CTRL Core

Figure 1-1 gives an schematic overview. The following list gives details, that are relevant for the verification:

- Fully synchronous design. All signals are clocked with the rising edge.
- 3 clock domains (can_clk, host_clk timer_clk). Clock domain crossing (CDC) used.
- All flip-flops have an asynchronous reset. For each clock domain there is a separate asynchronous reset input (host_rst_b, can_rst_b, timer_rst_b). There is no reset synchronizer inside CAN-CTRL. Reset synchronization needs to be done outside.
- For CAN 2.0B and CAN FD. Reduced source code for only CAN 2.0B available as well as full source code both. Features of CAN FD can be disabled by a generic parameter for the full source code.
- Configurable size of receive buffer (RB), transmit buffer (TB) and acceptance filters (ACF). All of those can be implemented as synchronous memories (BRAM in FPGAs) of array of flip-flops.
- Full handling of the CAN / CAN FD protocol
  - Additional reporting (e.g. for error observation)
  - Additional operating modes (e.g. for bus observation)
- Host controller interface is a 32 bit synchronous interface. Additional wrappers available:
  - Wrapper for 8 bit hosts
  - AMBA APB and AHB
  - Wishbone 32 bit
- Configurable interrupt sources. One interrupt output signal.
- Verilog 2001, VHDL 93

A description of the functional blocks of the CAN / CAN FD protocol machine can be found in the User's Guide (chap. "Functional Description"). The I/O pins and generic parameters are also described in the User's Guide (chap. "Interfaces").

2. Verification Methodology and Tools

2.1 Simulation and Verification Tools

Table 2-1 Simulation and Verification Tools

Task	Tool
Behavioral simulation	Cadence Incisive NCSim
LINT check	Cadence Incisive HAL
Code Coverage	Cadence Incisive IMC
ASIC synthesis	Cadence RC
FPGA synthesis	Altera Quartus Xilinx ISE Xilinx Vivado

Static timing analysis is done within the synthesis tool itself. For ASIC synthesis this is sufficient, because a standalone STA tool is not necessary after synthesis. For FPGA synthesis the mapping step is also included and therefore STA for the final implementation is included.

Scripts and project files for the tools are included in the release package.

2.2 Simulation Test Environment Description

2.2.1 Bosch Reference Model

The Bosch C Reference Model has been used to generate VHDL / Verilog stimuli for tests of the for CAN 2.0B operation. This is a behavioral test. It verifies compatibility to the CAN protocol.

The Bosch tests cover the CAN 2.0B protocol. The other testbenches are used to verify the CAN FD specific topics and the features that are accessible using the host controller interface.

2.2.2 The Bus Model

All the testbenches, except for the Bosch Reference Model, include a simulation of the CAN network which includes bus delays, transceiver models and error stimulation components. These simulation components are included in the release package. The used bus topologies are described in the appropriate chapters of the User's Guide.

2.2.3 The 3-Node Behavioral Testbench

The architecture of the 3-node environment is described in the User's Guide, chapter "Testbenches". In short it contains of three CAN / CAN FD nodes connected to a CAN network.

The 3-node environment is included into the release package, but only a few basic tests are included too. The majority of the tests are not included into the release package. The basic tests verify that all main protocol features are running and all included memories are useable. The full set of tests is used for internal verification.

Each test of the 3-node behavioral testbench is listed in the file <can-ctrl_tests.xlsx>, which is included in the release package. The full set of tests is listed in a compressed format in order to give an overview of all tests. With this file it is possible to see on a single monitor page which tests are included and successfully done which makes it easy to identify the settings that are not tested.

The file <can-ctrl_tests.xlsx> contains structure of a tree, that lists from left to the right, what is tested under which condition, under which sub condition and so on. Each row defines a test. On the right side each test is given a name and it is noted if the test has been successfully passed.

To make the file <can-ctrl_tests.xlsx> as short as possible, all conditions are a listed only with a short name of registers or bits. (E.g. the transmit buffer is just named TBUF.) All names refer to the names explained in the User's Guide.

A second sheet inside the file <can-ctrl_tests.xlsx> lists all synthesis parameters and the combination of these parameters that have been verified.

There are several hundreds of listed tests. The complete set focuses on the full CAN FD release package. If only CAN 2.0B functionality is required and configured by the appropriate generic parameter, then some tests become meaningless and are automatically not executed.

The tests of the 3-node behavioral testbench focus on specific tests for CAN FD and on tests for the host interface as well as the features of the CAN-CTRL core. The CAN 2.0B communication protocol is tested using the Bosch reference model (chap. 2.2.1). Please note that CAN FD is an evolution of CAN 2.0B and therefore a lot of the mechanisms are used by all versions of the protocol and are not tested again with the 3-node environment.

2.2.4 The Multicore Testbench

Several instances of CAN-CTRL can be made using the CAN multicore (see the User's Guide chap. "CAN Multicore"). The appropriate testbench is described in the User's Guide in the chapter "Testbenches" and the testbench sources are included in the release package.

This testbench verifies the operation of the multicore instantiation and executes some basic communication tests. All details about this test can be found in the User's Guide.

2.2.5 The AMBA APB Testbench

The release package includes an interface wrapper for the AMBA APB bus (see chapter "Host Interfaces" in the User's Guide). This testbench verifies the operation of the APB wrapper and includes all tests of supported read and write accesses. This testbench is included in the release package.

2.2.6 The AMBA AHB Testbench

The release package includes an interface wrapper for the AMBA AHB bus (see chapter "Host Interfaces" in the User's Guide). This testbench verifies the operation of the AHB wrapper and includes all tests of supported read and write accesses. This testbench is included in the release package.

2.2.7 The Wishbone Testbench

The release package includes an interface wrapper for the 32 bits wide Wishbone bus (see chapter "Host Interfaces" in the User's Guide). This testbench verifies the operation of the Wishbone wrapper and includes all tests of supported read and write accesses. This testbench is only delivered by request.

2.2.8 The Host Controller Testbench

The release package includes a software header file for ANSI C for the MSP430 microcontroller CPU. This is verified using a HDL simulation of two MSP430-based microcontrollers running some basic communication tests.

2.3 Verification IP

This IP core has been tested using Avery Design Systems' verification IP (VIP). The VIP provides a very large set of behavioral tests for classic CAN and CAN FD including tests from ISO 16845-1 "Road vehicles – Controller area network (CAN) conformance test plan – Part 1: Data link layer and physical signaling". Refer to Avery's VIP documentation for details about these tests.

CAN-CTRL includes synchronizers for input data which can be (de)selected by a generic parameter. Synchronizers add a signal delay to the input path. For verification with the VIP the synchronizers need to be deselected. In contrast to this the real world application system will include the synchronizers and a conformance test house will compensate the resulting input delay.

2.4 LINT

Several LINT tools exist and every tool can be used with different checks enabled or disabled. Furthermore checks may be classified as error, waring or information depending on the configuration of the LINT tool. The following gives a tool-independent guideline how to use LINT with the IP core CAN-CTRL and how to handle the LINT reports.

In general it is highly recommended to inspect the source code of the IP core where a LINT message points to. Source code comments may give useful information and additionally the source code is written in a style where the interpretation of a LINT message related to a part of source code is not difficult.

2.4.1 General

The top component of the IP core CAN-CTRL is the component <can_ctrl>, but this component only connects the inner core with the memories. The memory model includes an option for a true dual port implementation of the memory, which synthesizes well for common FPGAs. The memory model is based on the templates given by FPGA vendors and requires the usage of a shared variable (VHDL core) and multiple drivers to the memory array (VHDL and Verilog code). Such code will result in a LINT tool claiming a problem of synthesizability. Such a hard error stops the LINT tool. Therefore it is recommended to exclude the memory model from the LINT check and as a result it is recommended to not run LINT with the top component <can_ctrl> but only with the inner core <can_core_nobuf>.

The Bosch testbench (chap. 2.2.1) requires access to internal signals of the IP core. For VHDL this is done by the usage of global signals. Global signals are excluded from synthesis using pragmas, but LINT tools throw an error anyway. Therefore the global signals have to be disabled and this is done by the parameter <LINT_ENABLE> in <src/can_package.vhd> which has to be set to 1 for running LINT.

2.4.2 CDC and Clocks

Clock Domain Crossing (CDC) is done for most parts of the design using standard 2-stage CDC and handshaking if multiple bits have to cross the domain. This is recognized by LINT tools. For a few signals handshaking is done with the help of state machines where it is guaranteed, that the source signals is stable when the domain is crossed. This may not be recognized by all LINT tools and if not then it will result in a few LINT errors. For all of them the source code includes comments that CDC is done. Therefore if such CDC LINT errors occur, then they shall be waived.

CAN-CTRL is an IP core where several clock domains exist and there are subcomponents where more than one clock is used. This may result in a LINT warning about multiple master clocks inside a component. CDC is done with care and using multiple master clocks is not a problem. These LINT warnings shall be waived.

Clocks and asynchronous sets / resets are renamed inside subcomponents which results in LINT warnings. These shall be waived.

2.4.3 Configurations

The IP core can be configured using several parameters before synthesis. This enables or disables features and therefore forces several logic expressions to evaluate to constants or to be unreachable at all. This includes unreachable states in state machines. As a result several internal signals may be fixed to be constant or may be unused depending on the chosen configuration. LINT tools will throw several errors related to this but for every LINT error the source code is always clear to read so that it is obvious that the chosen configuration of the IP core was the reason. Configuration-dependent LINT errors and warnings shall be waived.

2.4.4 Combinational Paths

CAN-CTRL is an IP core where combinational paths cross several levels of hierarchy. LINT tools may throw an error because of this, but it is done by intention and these LINT errors shall be waived.

2.4.5 Top Level Outputs

Top level outputs which are intended to be connected to other IP cores at the same ASIC and are not intended to be connected to ASIC I/O pins are not all registered. Registering all outputs would lead to additional delays which is inefficient and synthesis will take care that the timing is met. These LINT warnings shall be waived.

2.4.6 Index of Vectors

LINT tools throw warnings that for a few vectors the index is potentially out of the defined range. This happens e.g. if a counting signal has a bigger range that the vector size. Counting signals are used for multiple tests and their size depends on the more than requirement. Therefore such signals cannot

always match to the size of every vector. Exhaustive behavioral testing has taken care, that it will never cause trouble and such LINT warnings shall be waived.

2.4.7 Arithmetic

LINT tools check the operands of arithmetic operators for equal bit width. If for a simple incrementer a "+1" is written, then the operator is an integer (32 bit) with in most cases does not have the same width as the other operator. This causes no trouble, synthesizes well to optimal size and is most importantly clean to read and easy to understand by an engineer, but LINT tools throw warnings which shall be waived.

2.4.8 Host interface

The address signal of the host interface <host_adr> addresses bytes but the data width is 32 bits (4 bytes). To select individual bytes for read or write access the selectors <host_rd_b> and <host_wr_b> are 4 bits wide. Therefore the lower bits (1:0) of <host_adr> are redundant and not used. This is done by intention and shall prevent from mistakes when creating an instance of the IP core and connecting external signals. LINT tools warnings according to that shall be ignored.

2.5 Code Coverage

Table 2-2 Code coverage

Coverage Type	CAN/FD core	Interfaces (overall)
Block	96.77% (2486/2569)	100% (77/77)
Branch	96.37% (2174/2256)	100% (70/70)
Statement	95.26% (2592/2721)	100% (44/44)
Expression	91.22% (4386/4808)	88.81% (246/277)
Toggle	97.02% (4460/4597)	94.00% (580/617)
FSM	97.53% (79/81)	(* no FSM)

In Table 2-2 the coverage is separately given for the interfaces (AMBA AHB/APB, Wishbone and Multicore) and the CAN core itself. The coverage is reported by the coverage tool IMC and the data was collected by using the VHDL source and the interface testbenches, the 3-node behavioral testbench and the Bosch testbench. Some signals that are not suitable have been excluded from toggle coverage.

2.6 Real World Test Environment Description

2.6.1 The FPGA System

The IP core has been tested on an FPGA platform as described at http://www.cast-inc.com/ip-cores/interfaces/canfd-rd/index.html. This system includes a microcontroller as host and CAN-CTRL as peripheral. Connection to a real CAN bus is done using various transceivers. Fast CAN FD tests are done using the transceivers listed in Table 2-3.

Table 2-3 Tested Transceivers

Company	Transceiver	Comment
Denso	RSCANFD (ringing suppression)	Preview sample
Infineon	TLE9250	Preview sample
Microchip	MCP2562FD	Commercial available
NXP	TJA1057 (Mantis series)	Commercial available
ON Semiconductor	NCV7340-4	Preview sample

The achievable speed strongly depends on the used CAN bus topology and the used transceivers. In laboratory conditions tests up to 10 Mbit/s were successfully done. Note, that this is not a benchmark and performance will strongly depend on the used hardware and the bus topology.

2.6.2 Communication with Vector CANoe

The company Vector is one of the most important players in the market of CAN bus development and monitoring. Therefore their hardware (VN1610) and software (CANoe) have been used for real world tests of CAN-CTRL. All kinds of CAN and FD frame types with all possible number of payload bytes have been successfully transmitted and received.

2.6.3 CiA Plug Fest

CAN in Automation (CiA, http://www.can-cia.org/) is an organization that provides a platform for development and promotes the image of the CAN bus. CiA organizes plug fests, where developers of CAN / CAN FD protocol machines, of transceivers, of CAN bus observation tools as well as car manufacturing companies meet and plug all of their devices together using various and challenging CAN bus topologies. Such a plug fest is a hard test in the real world, where all the different devices and implementations interact with each other.

A plug fest is not a fixed benchmark or some kind of certification. It is a platform where various different tests can be executed and the interoperability of the devices is verified. The CAN-CTRL has been successfully tested a CiA plug fest.

The achievable bus communication speed strongly depends on the used devices and the bus topology. CAN-CTRL was able to successful participate in the communication at all speeds, which verifies the mechanisms of delay compensation and robustness. Depending on the used bus topology 4 to 6 Mbit/s was reached at the plug fest.

The CAN-CTRL was connected to CAN FD products of the following companies while accomplishing several tests at CiA plug fest: Bosch, Cypress, Etas, Denso, Intrepid, K2L, Kvaser, Microchip, Renesas, TI, and Vector.

2.6.4 Linux Test System

The Linux device driver has been tested using a Zybo Z7 board by Digilent based on a Zynq-7000 SoC running PetaLinux 2016.2. Two CAN FD transceiver TCAN337G (Texas Instruments) are used for connecting the CAN bus. As described in the INM the Linux SocketCAN networking layer is used by the device driver. Available tools (can-utils) are used for testing transmission and reception of CAN and CAN FD frames.

2.7 ISO 16845

ISO 16845 exists as a conformance test specification. Conformance test houses offer the appropriate tests.

Conformance certification can only be done for a complete CAN node, which includes the bus protocol machine as well as the host system. This is necessary because an erroneous host may be able to override features of the protocol machine and may be able to interact with the bus in a way that is not allowed. The certification is lost, if the host system is changed. Therefore the developer of the CAN node is suggested to do the conformance test.