



Gowin PCI to CAN IP

User Guide

IPUG949-1.01E,04/23/2021

Copyright © 2021 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.

GOWIN, Gowin and GOWINSEMI are trademarks of Guangdong Gowin Semiconductor Corporation and are registered in China, the U.S. Patent and Trademark Office, and other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders. No part of this document may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of GOWINSEMI.

Disclaimer

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. All information in this document should be treated as preliminary. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

Revision History

Date	Version	Description
09/22/2020	1.0E	Initial version published.
04/23/2021	1.01E	Minor linguistic errors corrected.

Contents

Contents	i
List of Figures	ii
List of Tables	iii
1 About This Guide	1
1.1 Purpose	1
1.2 Related Documents	1
1.3 Terminology and Abbreviations	1
1.4 Support and Feedback	1
2 Overview	2
2.1 Functions	2
2.2 Working Frequency	2
2.3 Resource Utilization	2
3 Functional Description	3
3.1 System Block Diagram	3
3.2 Block Diagram Description	3
3.3 Operating Flow	4
4 Interface Description	5
4.1 Pin Diagram	5
4.2 Pin Description	5
5 Timing	7
5.1 Write Operation Timing	7
5.2 Read Operation Timing	8
6 IP Generation	9

List of Figures

Figure 3-1 PCI to CAN Block Diagram	3
Figure 3-2 PCI to CAN Block Diagram	4
Figure 4-1 Pin Diagram of PCI to CAN	5
Figure 5-1 Write Operation Timing	7
Figure 5-2 Read Operation Timing	8
Figure 6-1 IP Generation	9
Figure 6-2 GUI	10
Figure 6-3 Instantiation	10

List of Tables

Table 1-1 Terminology and Abbreviations	1
Table 2-1 Gowin PCI to CAN IP Overview	2
Table 2-2 Resource Utilization	2
Table 4-1 Pin Description	5

1 About This Guide

1.1 Purpose

Gowin PCI to CAN IP user guide aims to help you quickly learn the features, functions and usage of Gowin PCI to CAN IP.

1.2 Related Documents

The latest user guides are available on the GOWINSEMI Website. You can find the related documents at www.gowinsemi.com:

- [DS100](#), GW1N series of FPGA Products Data Sheet
- [DS102](#), GW2A series of FPGA Products Data Sheet
- [SUG100](#), Gowin Software User Guide

1.3 Terminology and Abbreviations

The terminology and abbreviations used in this manual are as shown in Table 1-1.

Table 1-1 Terminology and Abbreviations

Terminology and Abbreviations	Meaning
IP	Intellectual Property
LUT	Look-up Tables
PCI	Peripheral Component Interconnect
CAN	Controller Area Network
REG	Register
BAR	Base Address Register

1.4 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

2 Overview

Gowin PCI to CAN IP allows you to configure CAN through PCI bus, realizing communication between Gowin PCI Target IP and Gowin CAN Controller IP.

Table 2-1 Gowin PCI to CAN IP Overview

Gowin PCI to CAN IP	
Logic Resource	See Table 2-2
Delivered Doc.	
Design Files	Verilog (encryption)
Reference Design	Verilog
TestBench	Verilog
Test and Design Flow	
Synthesis Software	Synplify Pro
Application Software	Gowin Software

2.1 Functions

The main function of Gowin PCI to CAN IP is to support a PCI device to communicate with a CAN device with REG interface. CAN device is designed as a backend device mounted on a BAR of PCI.

2.2 Working Frequency

Gowin PCI to CAN IP works together with Gowin PCI Target IP and Gowin CAN Controller IP at the same clock pci_clk, with a frequency of 33MHz.

2.3 Resource Utilization

Taking Gowin GW2A-55 device as an example, the resource utilization of Gowin PCI to CAN IP is shown in Table 2-2. For resource utilization on other Gowin FPGAs, please pay attention to later release information.

Table 2-2 Resource Utilization

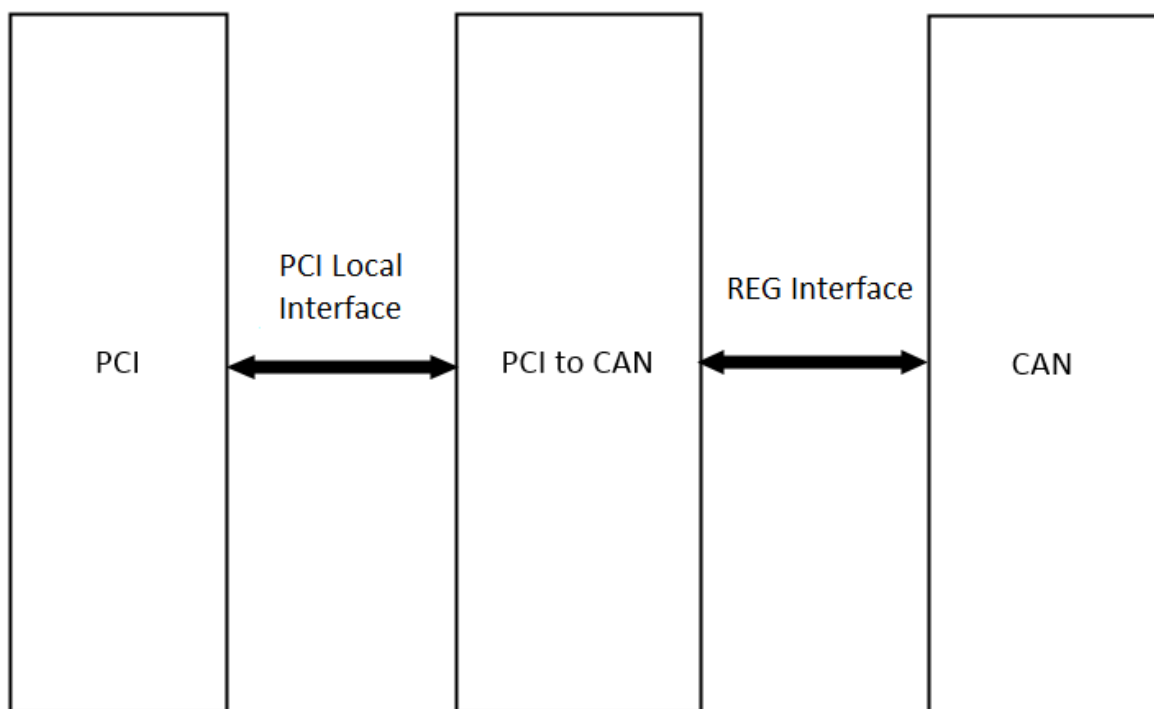
LUTs	REGs	Device Series	Speed Grade
86	85	GW2A55	-8

3 Functional Description

3.1 System Block Diagram

As shown in Figure 3-1, PCI to CAN is between the Local interface of PCI Target and REG interface of CAN, realizing interface conversion from PCI to CAN and communication.

Figure 3-1 PCI to CAN Block Diagram

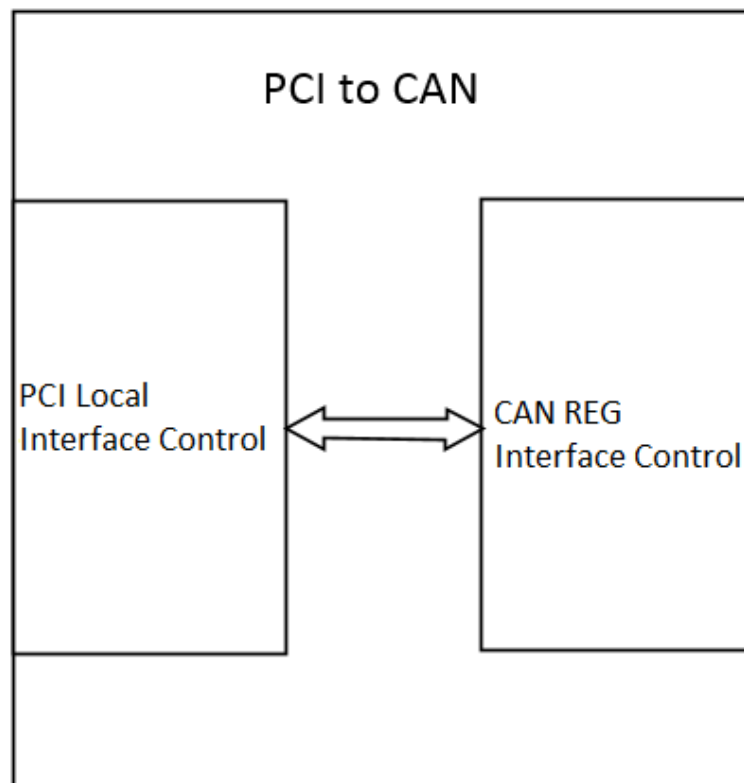


3.2 Block Diagram Description

This IP consists of two modules: PCI local interface control module and CAN REG interface control module. The functions are as follows:

- PCI Local interface control module realizes the communication with PCI Target local interface and completes the conversion of CAN REG interface signal.
- CAN REG interface control module realizes the communication with the CAN REG interface and completes the conversion of the PCI Local interface signal.

Figure 3-2 PCI to CAN Block Diagram



3.3 Operating Flow

You need to determine which BAR of PCI CAN is mounted on. Note that the `tg_bar_hit` signal of PCI Target can be 1 to 6 bits width, and the `tg_bar_hit` signal of PCI to CAN is 1 bit width. Therefore, one of the bits of `tg_bar_hit` needs to be connected to the `tg_bar_hit`, which can satisfy the operation of PCI Target on different bases.

The `cpu_addr` signal of CAN is mapped from the `tg_addr` signal of PCI Target. The lower 12 bits represent the register address, so the base size of PCI Target must be set to at least 8KB. Then you can use PCI to initialize CAN, the steps are as follows:

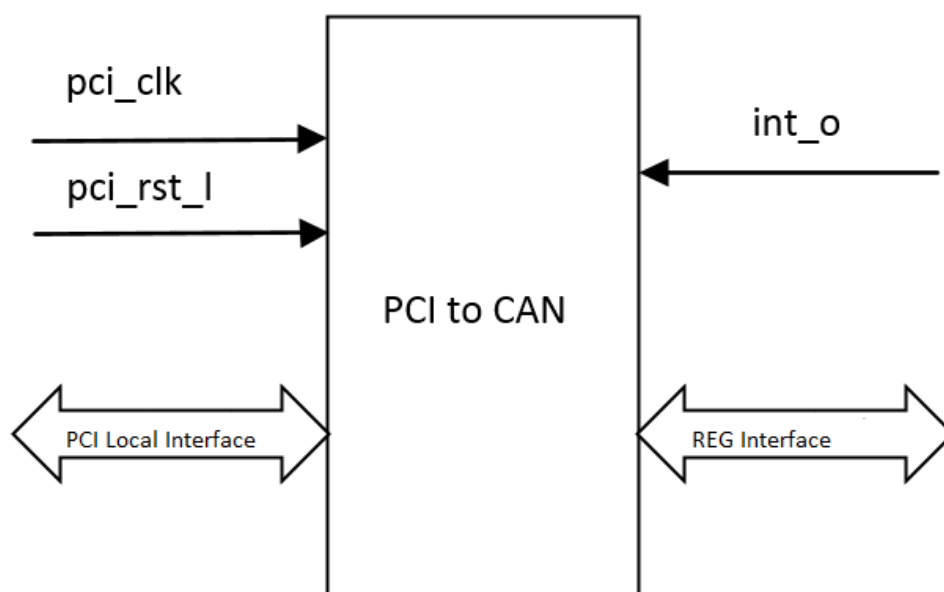
- Configure BRP register on the offset address 0x8 to set the prescaler value.
- Configure BTN and BTD registers related with the frame on offset address 0xC and 0x10.
- Configure RXBCFG, TXBCFG, TXHBCFG on offset address 0x40, 0x44, 0x48 to initialize receive and transmit buffers of CAN;
- Configure the interrupt enable register IE on the offset address 0x24;
- Configure AF registers on the offset addresses 0x100 to 0x13C, then configure AFM registers on the 0x140 to 0x17C to complete the setting of receive filter.
- Write 1 to offset address 0x4 so that CAN enters operating mode.

For details on Gowin PCI Target IP and Gowin CAN Controller IP, please refer to [IPUG904](#), Gowin PCI Target IP User Guide and [IPUG527](#), Gowin CAN Controller IP User Guide.

4 Interface Description

4.1 Pin Diagram

Figure 4-1 Pin Diagram of PCI to CAN



4.2 Pin Description

Table 4-1 Pin Description

Pin	I/O	Description
System Interface		
pci_clk	Input	PCI input clock. PCI Target and CAN Controller can work at the same clock frequency, so pci_clk is directly used as the system clock of PCI to CAN with the frequency of 33MHz.
pci_rst_l	Input	Asynchronous reset signal, active-low
PCI Local Interface		
tg_addr[31:0]	Input	PCI Target address signal
tg_data_out[31:0]	Input	PCI Target data output signal during write operation.

Pin	I/O	Description
tg_data_in[31:0]	Output	PCI Target data input signal during read operation.
tg_cbe_l[3:0]	Input	PCI Target byte enable signal, active-low.
tg_ready_l	Output	Ready to receive or transmit data, active-low.
tg_write_l	Input	PCI Target write operation indicator, active-low.
tg_read_l	Input	PCI Target read operation indicator, active-low.
tg_stop_l	Output	Transmission stop signal, active-low.
tg_abort_l	Output	Transmission abort signal, active-low.
tg_cmd_o[3:0]	Input	PCI Target transmission command signal
tg_bar_hit	Input	Single-bit base selection signal. This signal comes from one bit of the tg_bar_hit[5:0] signals on the PCI Target, which can be customized and can mount CAN on the required bar.
tg_access	Input	When this signal is valid, it indicates that PCI Target can access the local interface
tg_value	Input	When this signal is valid, it indicates that the data on tg_data_out[31:0] and tg_data_in[31:0] is valid during write/read operations.
tg_int_l	Output	Interrupt signal, active-low
RGE interface		
cpu_cs	Output	Chip select signal
cpu_read	Output	Read enable
cpu_write	Output	Write enable
cpu_addr[13:0]	Output	Address signal
cpu_wdat[31:0]	Output	Write data
cpu_rdat[31:0]	Input	Read data
cpu_ack	Input	Transmission done acknowledgement
cpu_err	Input	Transmission error
CPU interrupt signal		
int_o	Input	Interrupt signal

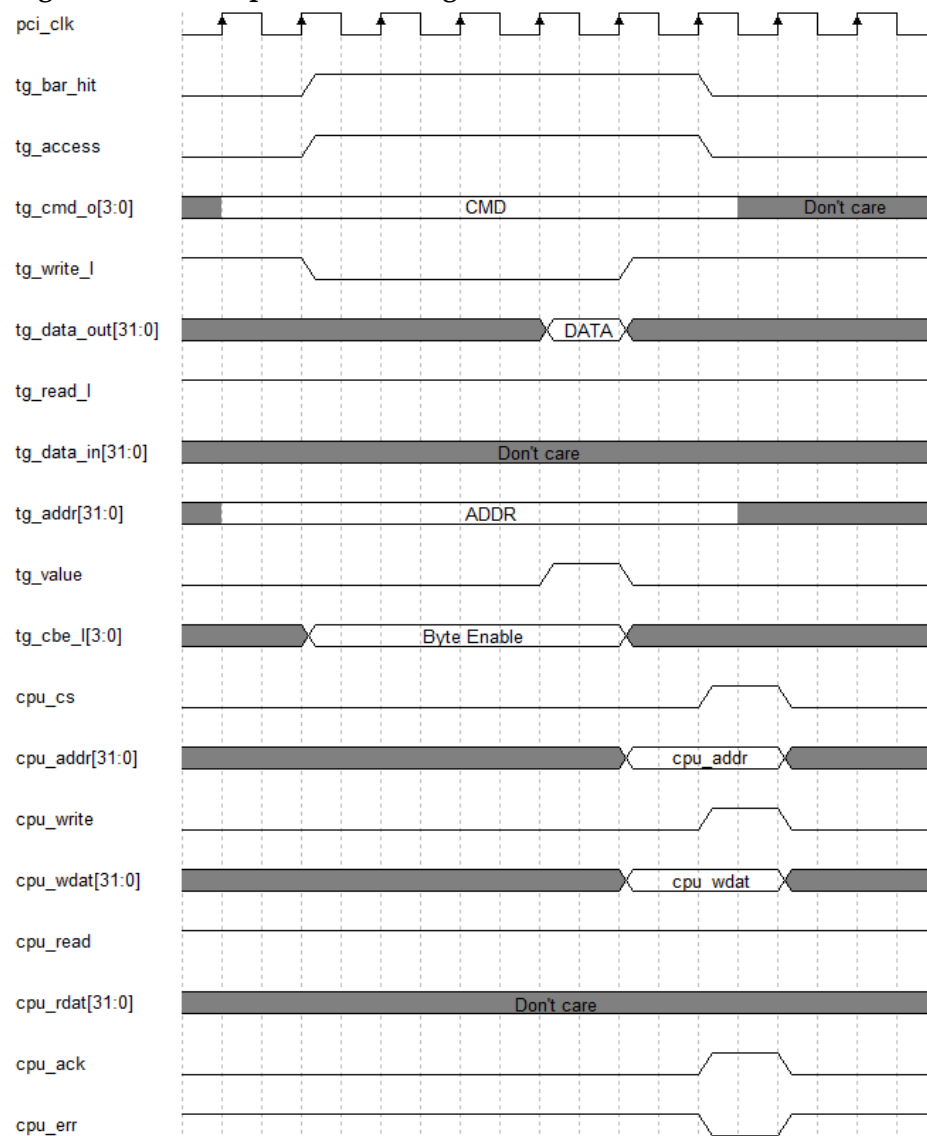
Note!

The I/O of all the signals takes PCI to CAN IP as reference.

5 Timing

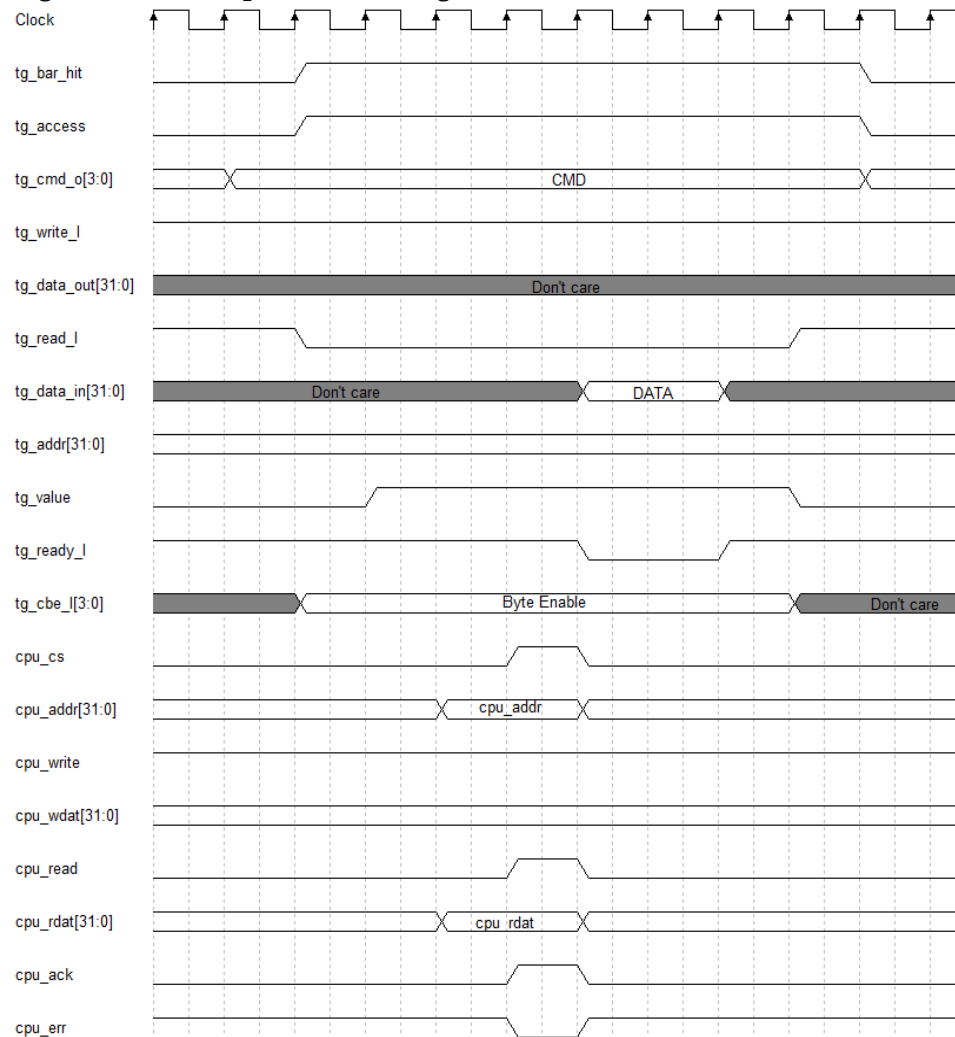
5.1 Write Operation Timing

Figure 5-1 Write Operation Timing



5.2 Read Operation Timing

Figure 5-2 Read Operation Timing



6 IP Generation

You can use IP core generator tool in the IDE to call and configure Gowin PCI to CAN IP.

Gowin PCI to CAN IP generation and GUI are shown in Figure 6-1 and Figure 6-2, and the instantiation is as shown in Figure 6-3.

Figure 6-1 IP Generation

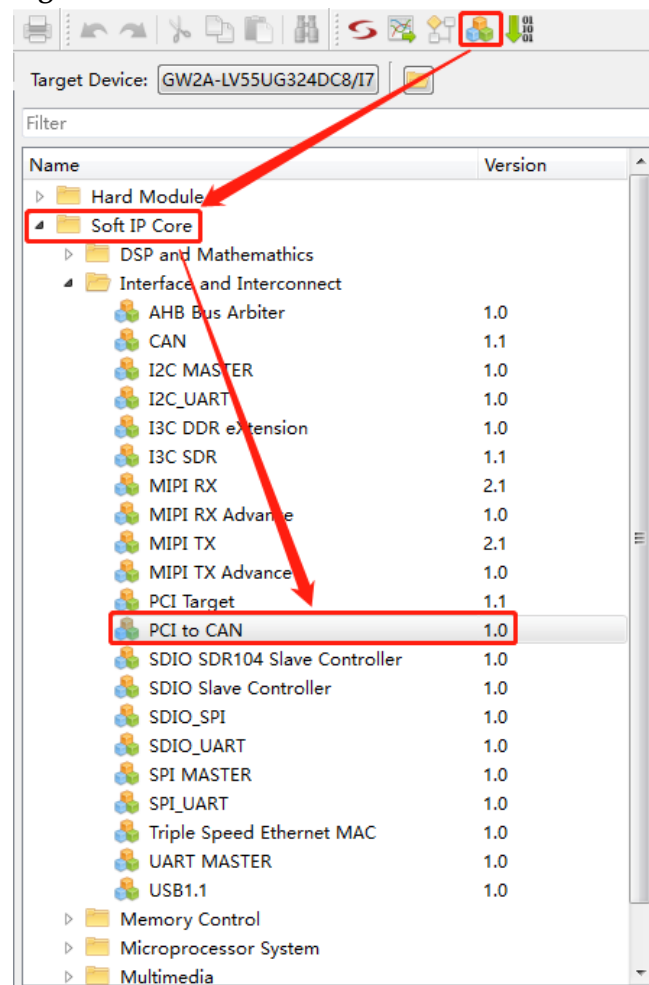


Figure 6-2 GUI

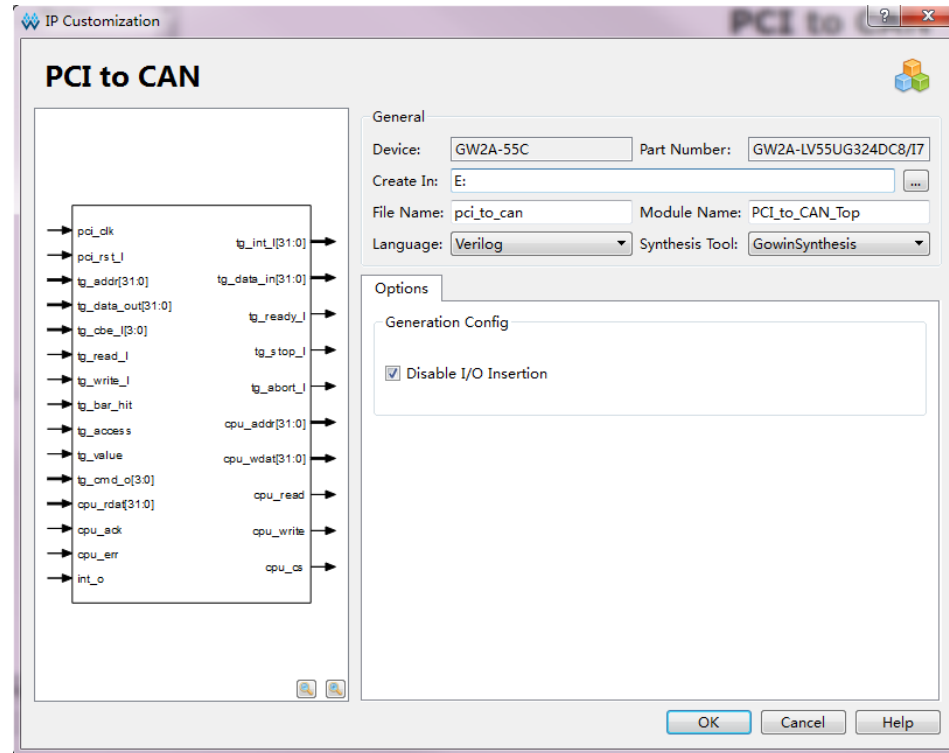


Figure 6-3 Instantiation

```
//Copyright (C) 2014-2020 Gowin Semiconductor Corporation.
//All rights reserved.
//File Title: Template file for instantiation
//GOWIN Version: GowinSynthesis V1.9.7Beta
//Part Number: GW2A-LV55UG324DC8/I7
//Device: GW2A-55C
//Created Time: Fri Sep 25 16:46:18 2020

//Change the instance name and port connections to the signal names
//-----Copy here to design-----

-->PCI_to_CAN_Top your_instance_name(
-->-->.pci_clk(pci_clk_i), //input: pci_clk
-->-->.pci_rst_l(pci_rst_l_i), //input: pci_rst_l
-->-->.cpu_read(cpu_read_o), //output: cpu_read
-->-->.cpu_write(cpu_write_o), //output: cpu_write
-->-->.cpu_cs(cpu_cs_o), //output: cpu_cs
-->-->.cpu_addr(cpu_addr_o), //output: [31:0] cpu_addr
-->-->.cpu_wdat(cpu_wdat_o), //output: [31:0] cpu_wdat
-->-->.cpu_rdat(cpu_rdat_i), //input: [31:0] cpu_rdat
-->-->.cpu_ack(cpu_ack_i), //input: cpu_ack
-->-->.cpu_err(cpu_err_i), //input: cpu_err
-->-->.int_o(int_o_i), //input: int_o
-->-->.tg_addr(tg_addr_i), //input: [31:0] tg_addr
-->-->.tg_data_out(tg_data_out_i), //input: [31:0] tg_data_out
-->-->.tg_cbe_l(tg_cbe_l_i), //input: [3:0] tg_cbe_l
-->-->.tg_read_l(tg_read_l_i), //input: tg_read_l
-->-->.tg_write_l(tg_write_l_i), //input: tg_write_l
-->-->.tg_bar_hit(tg_bar_hit_i), //input: tg_bar_hit
-->-->.tg_access(tg_access_i), //input: tg_access
-->-->.tg_value(tg_value_i), //input: tg_value
-->-->.tg_cmd_o(tg_cmd_o_i), //input: [3:0] tg_cmd_o
-->-->.tg_int_l(tg_int_l_o), //output: tg_int_l
-->-->.tg_data_in(tg_data_in_o), //output: [31:0] tg_data_in
-->-->.tg_ready_l(tg_ready_l_o), //output: tg_ready_l
-->-->.tg_stop_l(tg_stop_l_o), //output: tg_stop_l
-->-->.tg_abort_l(tg_abort_l_o) //output: tg_abort_l
-->);
```

