Gowin PicoRV32 Software Programming
Reference Manual

IPUG911-1.6E, 09/26/2025

Revision History

Date	Version	Description
01/16/2020	1.0E	Initial version published.
03/12/2020	1.1E	● MCU supports GPIO of Wishbone bus interface. ● MCU supports extension AHB bus interface. ● MCU supports off-chip SPI-Flash download and startup. ● MCU supports the read, write and erasure SPI-Flash. ● MCU supports Hardware Stack Protection and Trap Stack Overflow.
06/01/2020	1.2E	● MCU on-line debug function supported. ● MCU core interrupt handler function enhanced. ● MCU core instruction optimized.
02/08/2022	1.3E	● The register address mapping of the peripheral Simple UART updated. ● The register definitions of the peripheral I2C Master and GPIO updated. ● The driver function definition of the peripheral SPI-Flash updated. ● The interrupt handler enhanced.
08/18/2023	1.4E	Arora Ⅴ FPGA products supported.
03/29/2024	1.5E	Software programming library updated.
09/26/2025	1.6E	● GMD software version information updated. ● Software programming reference designs updated to be compatible with GMD 2025.01. ● Chapter 10 Application Program added.

Contents

List of Tables

1 Software Programming Library

Gowin_PicoRV32 offers software programming library: …\library, and the software programming library definitions are as shown in Table 1-1.

Table 1-1 Software Programming Library Definition

File	Description
System Definition	
start.S	MCU core bootloader
custom_ops.S	The macro definition of custom interrupt processing instruction
config.h	The start-up operation mode of user configurations
picorv32.h	The definition of register, address mapping, clock signal, and system configuration, etc
irq.c	Interrupt handler function
Linker Definition	
sections.lds	Flash linker script file: MCU boot from external Flash and run in ITCM and MCU boot and run in ITCM
sections_xip.lds	Flash linker script file: MCU boot and run in external Flash
sections_debug.lds	Flash linker script file: MCU software debugging and running mode
Firmware Definition	
firmware.c	The definition of performance statistics, latency, etc
loader.c	The definition of loader
printf.c	Simple UART/WB UART printing ouptput
Driver Function Definition	
simpleuart.c simpleuart.h	The definition of Simple UART driver function
wbi2c.c wbi2c.h	The definition of Wishbone I2C Master driver function
wbspi.c wbspi.h	The definition of Wishbone SPI Master/Slave driver function
wbuart.c	The definition of Wishbone UART driver function

File	Description
wbuart.h	
wbgpio.c wbuart.h	The definition of Wishbone GPIO driver function
wbspiflash.c wbspiflash.h	The definition of Wishbone SPI-Flash driver function

2 Memory System

The definition of memory mapping address of Gowin_PicoRV32 standard peripheral is as shown in Table 2-1.

The definition location of memory mapping address of Gowin_PicoRV32 standard peripheral: picorv32.h.

Table 2-1 Memory Mapping Definition of Standard Peripheral

Standard Peripheral	Type	Address Mapping	Description
DTCM	–	0x01000000	Size: 8KB, 16KB, 32KB, 64KB, 128KB, 256KB, 512KB
ITCM	–	0x02000000	Size: 8KB, 16KB, 32KB, 64KB, 128KB, 256KB, 512KB
SIMPLEUART	SIMPLEUART_RegDef	0x03000000	Simple UART
WBUART	WBUART_RegDef	0x11000000	Wishbone UART
WBGPIO	WBGPIO_RegDef	0x11001000	Wishbone GPIO
WBSPI_MASTER	WBSPI_RegDef	0x11002000	Wishbone SPI Master
WBSPI_SLAVE	WBSPI_RegDef	0x11003000	Wishbone SPI Slave
WBSPI_FLASH	SPI_FLASH_RegDef	0x11004000	Wishbone SPI-Flash
WBI2C_Master	WBI2CMASTER_RegDef	0x11005000	Wishbone I2C Master
OPEN_WB_INTERFACE	-	0x20000000	Open Wishbone Interface for Customized Peripherals
OPEN_AHB_INTERFACE	-	0x80000000	Open AHB Interface for Cunstomized Peripherals

3 Interrupt Handler

3.1 Interrupt Feature

The interrupt controller includes the following features:

- Lightweight interrupt controller and interrupt control instructions.
- The interrupt is triggered by the rising edge.
- Provides 32 interrupt handler signals available to users.
- Software-controlled interrupt priority.
- Interrupt nesting is not supported.

3.2 Interrupt Control

The definitions of the interrupt control function of Gowin_PicoRV32 are shown in Table 3-1.

Table 3-1 Definitions of Interrupt Control Function

Interrupt control function	Parameter	Description
mask_irq	32-bit mask code	Interrupt mask function The corresponding interrupt status of the bit of the “0” in mask code is open; The corresponding interrupt status of the bit of the “1” in mask code is masked.
irq_enable_one_bit	32-bit interrupt number	Single interrupt enable function The interrupt bit corresponding to this number is enabled
irq_disable_one_bit	32-bit interrupt number	Single interrupt disable function The interrupt bit corresponding to this number is disabled
enable_external_interrupt	-	Enable external interrupt
disable_external_interrupt	-	Disable external interrupt
enable_timer_interrupt	-	Enable timer interrupt
disable_timer_interrupt	-	Disable timer interrupt
enable_interrupt_global	-	Enable global interrupt
disable_interrupt_global	-	Disable global interrupt

Interrupt control function	Parameter	Description
set_timer	32-bit timing length “timer_val”	Timer interrupt setting function The timer starts automatically and is triggerred when “timer_val” clock cycles are reached.
irq	32-bit interrupt register; 32-bit interrupt number	Interrupt handler function When the interrupt is triggerred, the function is automatically entered and the corresponding interrupt handler is called according to the interrupt bit number of the triggered interrupt.

3.3 Interrupt Numbering

The definitions of the preset interrupt numbering of Gowin_PicoRV32 are as shown in Table 3-2. The definition location of interrupt handler function in Gowin_PicoRV32: irq.c.

Table 3-2 Interrupt Number Definitions

Number	Description
0	32-bit timer interrupt
1	Execute debug breakpoint instruction (ebreak)
2	Misalignment access error of memory (bus error)
3	Illegal instruction error
4	Debug module interrupt
5	Hardware stack protection interrupt
6~9	Reserved
10	Wishbone SPI Master interrupt
11	Wishbone SPI Slave interrupt
12	Wishbone I2C Master interrupt
13	Wishbone UART interrupt
14~19	Reserved
20~31	Reserved for Wishbone and AHB extension bus interfaces to add Wishbone and AHB peripherals for users

4 Simple UART

4.1 Features

Gowin_PicoRV32 contains a lightweight Simple UART peripheral:

- No parity bit
- 8-bit data bit
- 1-bit stop bit
- Interrupt is not supported

Set the baud rate frequency division register when you enable Simple UART. For example, if the system clock frequency is running at 12MHz, and the baud rate is required to be 9600, you can set the baud rate frequency division register to 1250 (2000000/9600=1250).

4.2 Register Definition

The definitions of Simple UART register are as shown in Table 4-1. Simple UART register definitions are located in library\lib\picorv32.h.

Table 4-1 Simple UART Register Definitions

Name	Address Offset	Type	Width	Initial Value	Description
RESERVED	0x000	-	-	-	Reserved
CLKDIV	0x004	RW	32	0x1	Frequency division factor register, used to configure the baud rate
DATA	0x008	RW	32	0x-	Input/Output data register

4.3 Usage of Driver

The usage of Simple UART drivers is shown in Table 4-2. Simple UART driver definitions are located in library\lib\bsp\simpleuart.c.

Table 4-2 Usage of Simple UART Drivers

Name	Description
uart_init	Initialize the Simple UART and configure the baud rate
outbyte	Output a character, and return to the line head automatically

Name	Description
	when output a newline character.
uart_putchar	Output a character
getchar_prompt	Return the character received by UART RX
uart_getchar	Return the character received by UART RX

5 Wishbone I2C Master

5.1 Features

Gowin_PicoRV32 contains one I2C Master peripheral accessed by the Wishbone bus:

- Wishbone bus interface
- Compliant with I2C protocol
- Bus arbitration and arbitration lost detection
- Bus busy detection
- Interrupt flag generation
- Start/Stop/Repeated Start/Acknowledge generation
- Supports Start/Stop/Repeated Start detection
- Supports 7-bit addressing mode

5.2 Register Definition

The definitions of Wishbone I2C Master register are as shown in Table 5-1. Wishbone I2C Master register definitions are located in library\lib\picorv32.h.

Table 5-1 Definitions of Wishbone I2C Master Register

Name	Address Offset	Type	Width	Initial Value	Description
PRER	0x00	RW	32	0x0000FFFF	Clock prescale register [31:15] Reserved [15:0] Prescale value = sys_clk/(5*SCL)-1
CTR	0x04	RW	32	0x00000000	Control register [31:8] Reserved [7] Enable I2C function [6] Enable I2C interrupt [5:0] Reserved
TXR	0x08	WO	32	0x00000000	Transmit data register [31:8] Reserved

Name	Address Offset	Type	Width	Initial Value	Description
					[7:1] Next transmission data [0] Data direction
RXR	0x08	RO	32	0x00000000	Receive data register [31:8] Reserved [7:0] Last received data
CR	0x0c	WO	32	0x00000000	Command register [31:8] Reserved [7] STA, Start transmission status [6] STO, Over transmission status [5] RD, Read enable, read data from slave [4] WR, Write enable, write data to slave [3] Acknowledge [2:1] Reserved [0] Interrupt acknowledge
SR	0x0c	RO	32	0x00000000	Status register [31:8] Reserved [7] Receive acknowledge signal from slave [6] I2C busy status [5] Arbitration loss [4:2] Reserved [1] Data transmission status flag [0] Interrupt flag
CHANNEL	0x10	RW	32	0x00000000	Channel select register [31:1] Reserved [0] Channel select 0 = Channel 0 1 = Channel 1

5.3 Usage of Driver

The usage of Wishbone I2C Master driver is shown in Table 5-2. Wishbone I2C Master driver definitions are located in library\lib\bsp\wbi2c.c.

Table 5-2 Usage of Wishbone I2C Master Drivers

Name	Description
I2C_Init	I2C Initialization
I2C_SendByte	Send a byte to I2C bus
I2C_SendBytes	Send multiple bytes to I2C bus
I2C_SendWord	Send a word to I2C bus

Name	Description
I2C_SendArray	Send an array bytes to I2C bus
I2C_ReceiveByte	Read a byte from I2C bus
I2C_ReadBytes	Read multiple bytes from I2C bus
I2C_ReceiveWord	Read a word from I2C bus
I2C_RecevieArray	Read an array bytes from I2C bus
I2C_Rate_Set	Set I2C traffic rate
I2C_Enable	Enable I2C bus
I2C_Disable	Disable I2C bus
I2C_InterruptOpen	Open I2C interrupt
I2C_InterruptClose	Close I2C interrupt

6 Wishbone SPI Master/Slave

6.1 Features

Gowin_PicoRV32 contains an SPI peripheral accessed by the Wishbone bus, including SPI Master and SPI Slave:

- Wishbone bus interface
- Full duplex synchronous serial data transmission
- Configurable clock polarity and phase
- Configurable serial clock frequency generated by SPI
- Bit width for data receive register and data transmission register

6.2 Register Definition

The definitions of Wishbone SPI register are shown in Table 6-1. Wishbone SPI register definitions are located in library\lib\picorv32.h.

Table 6-1 Definitions of Wishbone SPI Register

Name	Address Offset	Type	Width	Initial Value	Description
RXDATA	0x00	RO	32	0x00000000	Receive data register [31:8] Reserved [7:0] Receive data
TXDATA	0x04	WO	32	0x00000000	Transmit data register [31:8] Reserved [7:0] Transmit data
STATUS	0x08	RW	32	0x00000000	Status register [31:8] Reserved [7] Overflow error status [6] Receive ready status [5] Transmit ready status [4] Be transmitting [3] Transmit overrun error status [2] Receive overrun error status

Name	Address Offset	Type	Width	Initial Value	Description
					[1:0] Reserved
CONTROL	0x0C	RW	32	0x00000000	Control register [31:5] Reserved [4:3] Clock selected, CLK_I / 2/4/6/8 [2] Clock polarity [1] Clock polarity [0] Direction, 1 is MSB first
SSMASK	0x10	RW	32	0x00000000	[31:1] Reserved [0] Select and enable slave

6.3 Usage of Driver

The usage of Wishbone SPI drivers is shown in Table 6-2. Wishbone SPI driver definitions are located in library\lib\bsp\wbspi.c.

Table 6-2 Usage of Wishbone SPI Drivers

Name	Description
wbspi_master_select_slave	WBSPI Master selects a slave device to communicate with
wbspi_enable_interrupt	Enable WBSPI interrupt
wbspi_disable_interrupt	Disable WBSPI interrupt
wbspi_master_txdata	WBSPI Master sends data to Slave device
wbspi_master_rxdata	WBSPI Master reads data from Slave device
wbspi_slave_prepare_txdata	WBSPI Slave preparation data, waiting for Master to read
wbspi_slave_read_data	WBSPI Slave reads data sent from Master

7 Wishbone UART

7.1 Features

Gowin_PicoRV32 contains one UART peripheral accessed by Wishbone bus:

- Wishbone bus interface
- No parity bit
- 8-bit data bit
- 1-bit stop bit

7.2 Register Definition

The definitions of Wishbone UART register are as shown in Table 7-1. Wishbone UART register definitions are located in library\lib\picorv32.h.

Table 7-1 Definitions of Wishbone UART Register

Name	Address Offset	Type	Width	Initial Value	Description
SETUP	0x00	RW	32	0x00000000	UART parameter/setup register
FIFO	0x04	RO	32	0x00000000	Status register of input FIFO and output FIFO
RXREG	0x08	RO	32	0x00000000	UART receiving data register
TXREG	0x0C	RW	32	0x00000000	UART transmitting data register

7.3 Usage of Driver

The usage of Wishbone UART drivers is shown in Table 7-2. Wishbone UART driver definitions are located in library\lib\bsp\wbuart.c.

Table 7-2 Usage of Wishbone UART Drivers

Name	Description
wbuart_init	Wishbone UART initialization, configure Baud Rate
wbuart_putc	Wishbone UART transmits a byte
wbuart_getc	Wishbone UART receives a byte

Name	Description
wbuart_outbyte	Wishbone UART transmits a byte, and returns to the line head automatically when output a newline character

8 Wishbone GPIO

8.1 Features

Gowin_PicoRV32 contains one GPIO peripheral accessed by Wishbone bus:

- Wishbone bus interface
- 32-bit, and each bit can be independently configured as input and output state

8.2 Register Definition

The definitions of Wishbone GPIO register are as shown in Table 8-1. Wishbone GPIO register definitions are located in library\lib\picorv32.h.

Table 8-1 Definitions of Wishbone GPIO Register

Name	Address Offset	Type	Width	Initial Value	Description
CFG	0x00	RW	32	0x00000000	GPIO configuration register [31:0] Each pin configuration
IE	0x04	RW	32	0x00000000	GPIO interrupt enable register [31:0] Each pin interrupt enable
RSV[2]	0x08-0x0C	-	-	-	Reserved
DIR	0x10	RW	32	0xFFFFFFFF	GPIO input/output direction register [31:0] Control each pin input/output direction 1 = Output 0 = Input
IN	0x14	RO	32	0x00000000	GPIO input register [31:0] Each pin input
OUT	0x18	WO	32	0x00000000	GPIO output register [31:0] Each pin output

8.3 Usage of Driver

The usage of Wishbone GPIO drivers is shown in Table 8-2. Wishbone GPIO driver definitions are located in library\lib\bsp\wbgpio.c.

Table 8-2 Usage of Wishbone GPIO Drivers

Name	Description
GPIO_Init	Wishbone GPIO initialization
GPIO_SetDir	Wishbone GPIO setting input/output
GPIO_GetDir	Wishbone GPIO getting input/output
GPIO_EnableWriteBit	Wishbone GPIO enable each bit output
GPIO_EnableReadBit	Wishbone GPIO enable each bit input
GPIO_WriteData	Wishbone GPIO output
GPIO_ReadData	Wishbone GPIO input

9 Wishbone SPI-Flash

9.1 Features

Gowin_PicoRV32 contains one SPI-Flash Memory peripheral accessed by Wishbone:

- SPI-Flash memory supports software programming design BIN File download startup and Instruction run
- SPI-Flash Memory supports read, write and erasure functions

9.2 Register Definition

The definitions of SPI-Flash memory registers are shown in Table 9-1. SPI-Flash Memory register definitions are located in library\lib\picorv32.h.

Table 9-1 Definitions of SPI-Flash Memory Register

Name	Address Offset	Type	Width	Initial Value	Description
IDREV	0x00	RO	32	0x020 02000	ID and revision register [31:8] ID number [7:4] Major revision number [3:0] Minor revision number
RESERVED0 [3]	0x04-0x 0C	-	-	-	Reserved
TRANSFMT	0x10	RW	32	0x000 20780	SPI transfer format register [31:18] Reserved [17:16] Address length in bytes 00 = 1 byte 01 = 2 bytes 10 = 3 bytes 11 = 4 bytes [15:13] Reserved [12:8] Data length [7] Enable data merge mode [6:5] Reserved [4] Bi-directional MOSI in single mode 0 = MOSI is uni-directional

Name	Address Offset	Type	Width	Initial Value	Description
					signal 1 = MOSI is bi-directional signal [3] Transfer data with the lease significant bit first 0 = Most significant bit first 1 = Least significant bit first [2] SPI master/slave mode selection 0 = Master mode 1 = Slave mode [1] SPI clock polarity 0 = SCLK is LOW in the idle states 1 = SCLK is HIGH in the idle states [0] SPI clock phase 0 = Sampling data at odd SCLK edges 1 = Sampling data at even SCLK edges
DIRECTIO	0x14	RW	32	0x0	SPI direct IO control register [31:25] Reserved [24] Enable direct IO 0 = Disable 1 = Enable [11:22 PM] Reserved [21] Output enable for SPI-Flash hold signal [20] Output enable for SPI-Flash write protect signal [19] Output enable for the SPI MISO signal [18] Output enable for the SPI MOSI signal [17] Output enable for SPI SCLK signal [16] Output enable for SPI CS signal [3:14 PM] Reserved [13] Output value for SPI-Flash hold signal [12] Output value for SPI-Flash write protect signal [11] Output value for SPI MISO signal [10] Output value for SPI MOSI signal [9] Output value for SPI SCLK

Name	Address Offset	Type	Width	Initial Value	Description
					signal [8] Output value for SPI CS signal [7:6] Reserved [5] Status of SPI-Flash hold signal [4] Status of SPI-Flash write protect signal [3] Status of SPI MISO signal [2] Status of SPI MOSI signal [1] Status of SPI SCLK signal [0] Status of SPI CS signal
RESERVED1 [2]	0x18-0x 1C	-	-	-	Reserved
TRANSCTRL	0x20	RW	32	0x0	SPI transfer control register [31] Reserved [30] SPI command phase enable 0 = Disable the command phase 1 = Enable the command phase (Master mode only) [29] SPI address phase enable 0 = Disable the address phase 1 = Enable the address phase (Master mode only) [28] SPI address phase format 0 = Address phase is single mode 1 = The format of the address phase is the same as the DualQuad data phase (Master mode only) [27:24] Transfer mode 0000 = Write and read at the same time 0001 = Write only 0010 = Read only 0011 = Write, Read 0100 = Read, Write 0101 = Write, Dummy, Read 0110 = Read, Dummy, Write 0111 = None data 1000 = Dummy, Write 1001 = Dummy, Read 1010~1111 = Reserved [23:22] SPI data phase format

Name	Address Offset	Type	Width	Initial Value	Description
					00 = Single mode 01 = Dual I/O mode 10 = Quad I/O mode 11 = Reserved [21] Append and one-byte special token following the address phase for SPI read transfers [20:12] Transfer count for write data [11] The value of the one-byte special token following the address phase for SPI read transfers 0 = token value is 0x00 1 = token value is 0x69 [10:9] Dummy data count [8:0] Transfer count for read data
CMD	0x24	RW	32	0x0	SPI command register [31:8] Reserved [7:0] SPI command
ADDR	0x28	RW	32	0x0	SPI address register [31:0] SPI address (Master mode only)
DATA	0x2C	RW	32	0x0	SPI data register [31:0] Data to transmit or the received data
CTRL	0x30	RW	32	0x0	SPI controller register [31:21] Reserved [20:16] Transmit FIFO threshold [15:13] Reserved [12:8] Receive FIFO threshold [7:5] Reserved [4] TX DMA enable [3] RX DMA enable [2] Transmit FIFO reset [1] Receive FIFO reset [0] SPI reset
STATUS	0x34	R0	32	0x0	SPI status register [31:24] Reserved [23] Transmit FIFO full flag [22] Transmit FIFO empty flag [21] Reserved [20:16] Number of valid entries int the transmit FIFO [15] Receive FIFO full flag

Name	Address Offset	Type	Width	Initial Value	Description
					[14] Receive FIFO empty flag [13] Reserved [12:8] Number of valid entries in the receive FIFO [7:1] Reserved [0] SPI register programming is in progress
INTREN	0x38	RW	32	0x0	SPI interrupt enable register [31:6] Reserved [5] Enable the slave command interrupt [4] Enable the end of SPI transfer interrupt [3] Enable the SPI transmit FIFO threshold interrupt [2] Enable the SPI receive FIFO threshold interrupt [1] Enable SPI transmit FIFO underrun interrupt (Slave mode only) [0] Enable SPI receive FIFO overrun interrupt (Slave mode only)
INTRST	0x3C	WO	32	0x0	SPI interrupt status register [31:6] Reserved [5] Slave command interrupt (Slave mode only) [4] End of SPI transfer interrupt [3] TX FIFO threshold interrupt [2] RX FIFO threshold interrupt [1] TX FIFO underrun interrupt (Slave mode only) [0] RX FIFO overrun interrupt (Slave mode only)
TIMING	0x40	RW	32	0x0	SPI interface timing register [31:14] Reserved [13:12] The minimum time between the edges of SPI CS and the edges of SCLK [11:8] The minimum time the SPI CS should stay HIGH [7:0] The clock frequency ratio between the clock source and SPI interface SCLK
RESERVED2 [3]	0x44-0x 4c	-	-	-	Reserved
MEMCTRL	0x50	RW	32	0x0	SPI memory access control register [31:9] Reserved

Name	Address Offset	Type	Width	Initial Value	Description
					[8] This bit is set when “MEMCTRL”/“TIMING” is written [7:4] Reserved [3:0] Selects the SPI command
RESERVED3 [3]	0x54-0x 5C	-	-	-	Reserved
SLVST	0x60	RW	32	0x0	SPI slave status register [31:19] Reserved [18] Data underrun occurs in the last transaction [17] Data overrun occurs in the last transaction [16] SPI is ready for data transaction [15:0] User defined status flags
SLVDATACN T	0x64	R0	32	0x0	SPI slave data count register [31:25] Reserved [24:16] Slave transmitted data count [15:9] Reserved [8:0] Slave received data count
RESERVED4 [5]	0x68-0x 78	-	-	-	Reserved
CONFIG	0x7C	R0	32	0x0	Configuration register [31:15] Reserved [14] Support for SPI slave mode [13] Reserved [12] Support for memory-mapped access through AHB bus [11] Support for direct SPI IO [10] Reserved [9] Support for Quad I/O SPI [9] Support for Dual I/O SPI [7:6] Reserved [5:4] Depth of TX FIFO 00 = 2 words 01 = 4 words 10 = 8 words 11 = 16 words [3:2] Reserved [1:0] Depth of RX FIFO 00 = 2 words 01 = 4 words 10 = 8 words 11 = 16 words

9.3 Usage of Driver

The usage of SPI-Flash memory drivers is shown in Table 9-2. SPI-Flash Memory driver definitions are located in library\lib\bsp\wbspiflash.c.

Table 9-2 Usage of SPI-Flash Memory Drivers

Name	Description
spi_flash_init	Initialize SPI-Flash Memory
spi_get_fifo_depth	Get SPI fifo depth
change_mode_spi_flash	Switch SPI-Flash Memory mode between download and read, write, erase memory
spi_flash_read	Read data from SPI-Flash Memory
spi_flash_write	Write data into SPI-Flash Memory
spi_flash_page_program	Write data into SPI-Flash Memory with pages
spi_flash_sector_erase	Erase SPI-Flash Memory with sector
spi_flash_write_cmd	Write command to SPI-Flash Memory
spi_flash_read_status	Read SPI-Flash Memory status

10 Application Program

Gowin_PicoRV32 provides applications for the GMD (GOWIN MCU Designer) software (V2025.01, tested).

…\ref_design\MCU_RefDesign\picorv32_demo: Runs the Gowin_PicoRV32 application program on physical hardware, demonstrating how to use peripherals and interfaces such as UART, GPIO, SPI Flash, Wishbone interface, AHB interface, and user interrupts.

…\ref_design\MCU_RefDesign\picorv32_qemu_demo: Emulates the Gowin_PicoRV32 application program in QEMU, demonstrating how to emulate the use of peripherals such as UART and GPIO.

PROGRAMMING FOR THE FUTURE