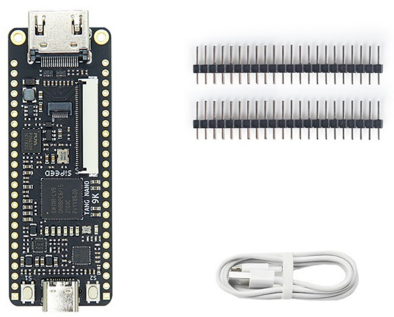
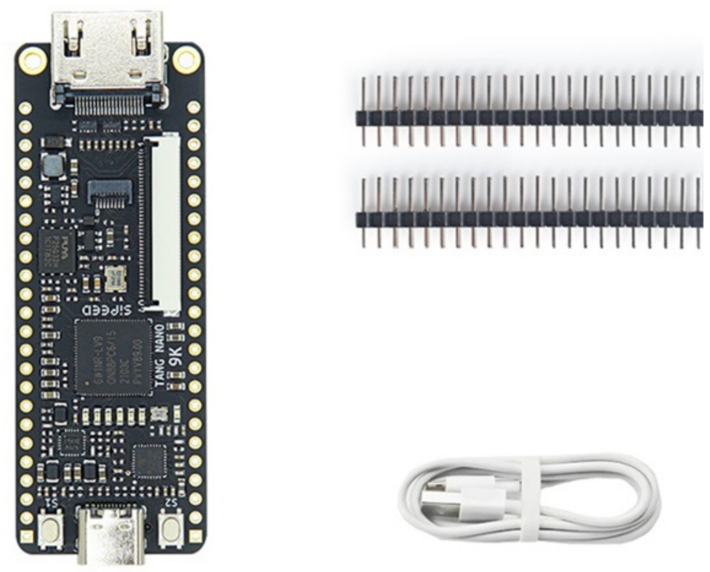


GOWIN FPGA Development Board RISCV Programming User Guide

Contents

GOWIN FPGA Development Board RISCV Programming



Revision History

Date	Version	Description
04/29/2019	1.0E	Initial version published.
11/11/2022	1.1E	• AndeSight RDS v311 software updated. • Reference design updated. • The description of downloading embedded project compilation results via SPI Flash updated.

Introduction

AE250 Introduction

AE250 is a 32-bit RISC-V MCU system; its structure is shown in Figure 1-1.

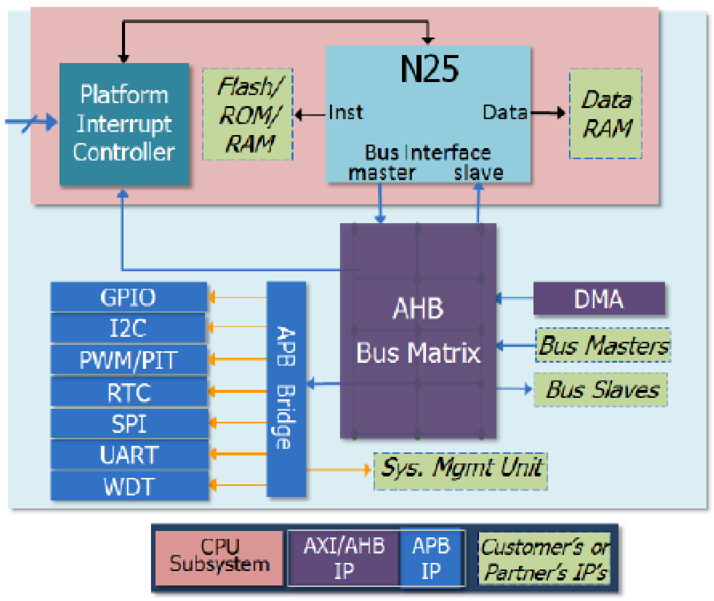


Figure 1-1 AE250 Structure Diagram

Based on Gowin FPGA development board, the RISC-V AE250 MCU development and debugging system is shown in Figure 1-2.

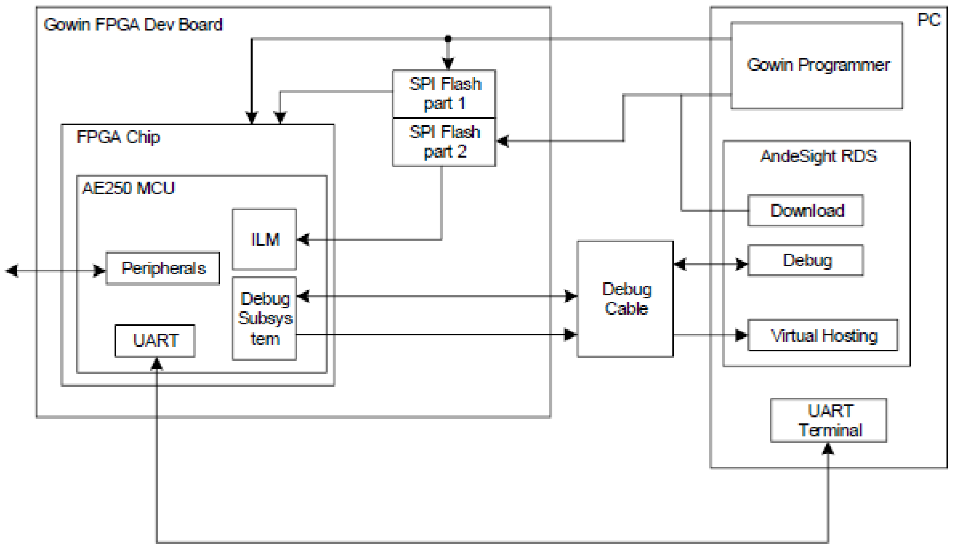


Figure 1-2 Development and Debugging System Structure Diagram

The FPGA chip on the development board is configured as an AE250 MCU using Gowin Programmer in PC, after the Debug Cable is connected, you can perform the embedded program development and debugging with AndeSight RDS v311 software.

Preparations

Before using Gowin FPGA and AE250 for development and debugging, the following tools need to be prepared:

1. Gowin GW2A series of FPGA development board.
2. Gowin Software installation package for configuring and downloading the FPGA chip.
3. AndeSight RDS v311 installation package for developing and debugging the embedded program.

4. Debug Cable is used for downloading and debugging the embedded program, and the default is AICE-MINI+; users need to purchase it by themselves.

Note

1. If it needs to output information through UART, a UART to USB cable is needed.
2. Other peripherals to be used are required.

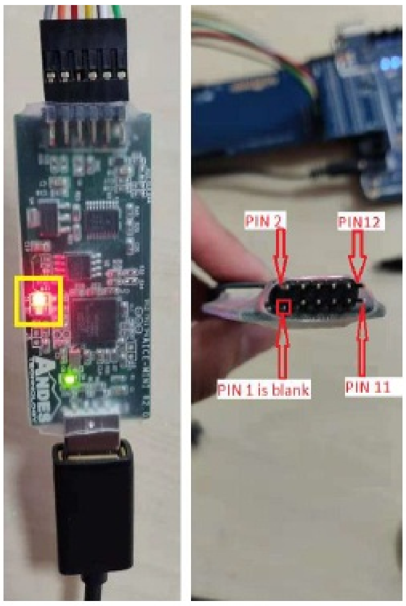
Developing and Debugging Steps

The basic steps for developing and debugging RISC-V AE250 MCU based on the GW2A-55C development board are as follows:

1. Install softwares: Gowin Software is used to configure and generate AE250 RTL design and generate the Bitstream file of the design; AndeSight RDS v311 software is used to develop and debug embedded programs; other softwares and drivers for debugging are also required.
2. Configure the power supply and download cable of the development board. The Bitstream file of AE250_chip is downloaded to the FPGA chip on the development board using Gowin Programmer, and AE250 is running on the development board.
3. Open RDS software to create a new embedded project or open an existing project for encoding, compiling and other operations. Connect the Debug Cable used for AE250 debugging, download the project compilation result to the instruction memory (ILM) in AE250, and start debugging on the chip.
4. During debugging, you can use UART to USB cable to connect the UART interface of AE250 to PC, use the built-in serial terminal in RDS to operate the input and output operations. You can use GPIO to connect to LED indicators, keys, or external pins for input/output operations; I2C, SPI, Ethernet, and other peripherals can also be selected to use.
5. AE250 can connect to a Flash via SPI, download the compilation result of embedded program to Flash using Gowin Programmer; when the chip is powered on, AE250 will automatically read the embedded program in SPI Flash and start. You can reuse the Flash that saves the FPGA Bitstream; some can save the FPGA bitstream, and others can save the compilation results of embedded programs. This is a practical and economical method.
   You can see chapter 2 Debug Cable Connection Instructions, chapter
   3 Use Instructions for RDS, and chapter 4 Reference Design for detailed steps.

Debug Cable Connection Instructions

RDS + AE250 use AICE-MINI+ debug cable by default; the exterior is shown at left in Figure 2-1, and the pins are shown at right in Figure 2-1. It is a 12-pin interface. It is should be noted that pin 1 is blank in the figure. When the cable is correctly connected and RDS is opened, the red LED light marked with yellow box in the figure will go out.
Figure 2-1 AICE-MINI+ Debug Cable and its Pins



The pin definition of AICE-MINI+ debug cable is as shown in Table 2-1. It should be noted that Pin 1 is defined as No Connection (NC), corresponding to the blank one. VREF needs to connect a 3.3V power pin, and GND only needs to connect the pin 3 or the pin 5.

Table 2-1 AICE-MINI+ Debug Cable Pin Definition

Pin Number	AICE-MINI+ Debug Cable Pin
1	NC
2	TSRST_N
3	GND
4	TTMS
5	GND
6	TCK
7	VREF
8	NC
9	NC
10	TTRST_N
11	TTDO
12	TTDI

Use Instructions for RDS

RDS Installation

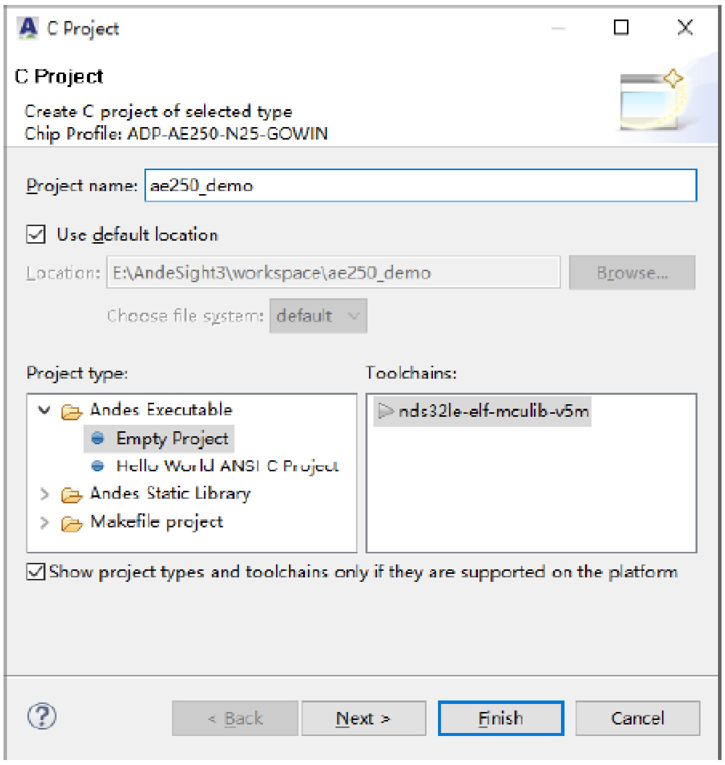
Unzip the installation package and enter Windows/Disk1; double-click setup.exe to install it. No special settings are required during installation. During installation, a dialog box will pop up asking whether to install the driver, please select yes. For installation steps, see AndeSight_RDS_v3.2_Installation_Guide_UM207_V1.0.pdf, which can be found in the installation package.

1. When setting the installation path and workspace path, do not include Chinese characters or space, or it will get a runtime error.
2. The current version of RDS supports AICE-MINI+ Cable by default.
3. GOWIN Programmer may be unable to connect to the development board after installing RDS, which can be fixed by reinstall Gowin Programmer driver.
4. For serial number and certificate files, please contact Gowin Semiconductor Corp.

Create a New Project

Click File > New > Project > Andes C project > Next on RDS interface to enter the configuration interface of New C Project, as shown in Figure 3-1.

Figure 3-1 Create a New Project


C Project

Create C project of selected type
Chip Profile: ADP-AE250-N25-GOWIN

Project name: ae250_demo

☑ Use default location

Location: E:\AndeSight3\workspace\ae250_demo

Choose file system: default

Project type:
- Andes Executable
  - Empty Project
  - Hello World ANSI C Project
- Andes Static Library
- Makefile project

Toolchains:
- nds32le-elf-mculib-v5m

☑ Show project types and toolchains only if they are supported on the platform

For the new C project, the following parameters need to be configured:

1. Project name
2. Location: The default location is the current workspace.
3. Connection Configuration is set to ICE, indicating that the development board is connected using ICE debug cable. If the emulator is used as a test platform, please select SID.
4. For Chip Profile, select ADP-AE250-N25-GOWIN, which is optimized according to Gowin FPGA.
5. Project Type includes an Empty Project and a Hello World ANSI C Project.
6. For Toolchains, nds32le-elf-mculib-v5m is the default.

   After creating a new project, right-click on the project name in the Project Explorer, select Build Project from the drop-down menu or click " " on the toolbar to compile and link the project; select Clean Project from the drop-down menu to make the project clean.

Import and Export a Project

Right click on the space of Project Explorer to select "Import" or "Export", as shown in Figure 3-2.

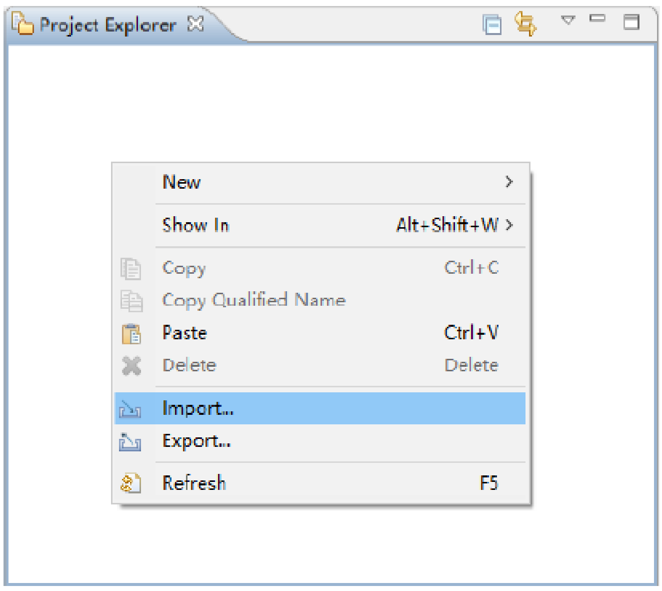


Figure 3-2 Import/Export a Project

Click “Import > General > Existing Project into workspace” to import a project, and the interface is as shown in Figure 3-3. When selecting “Select root directory”, import the project in folder; when selecting “Select archive fil”, import the project in zip.

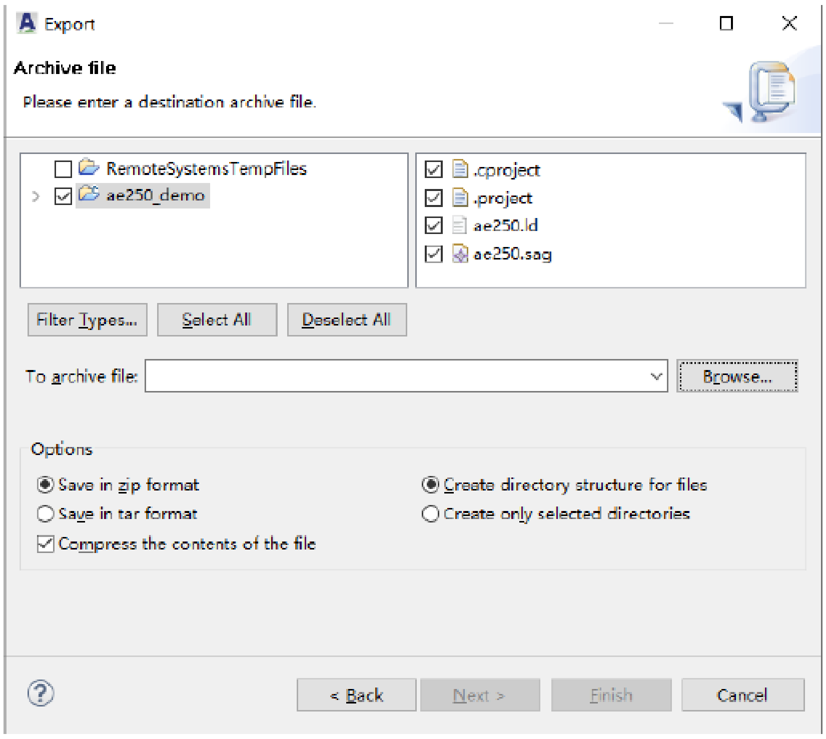
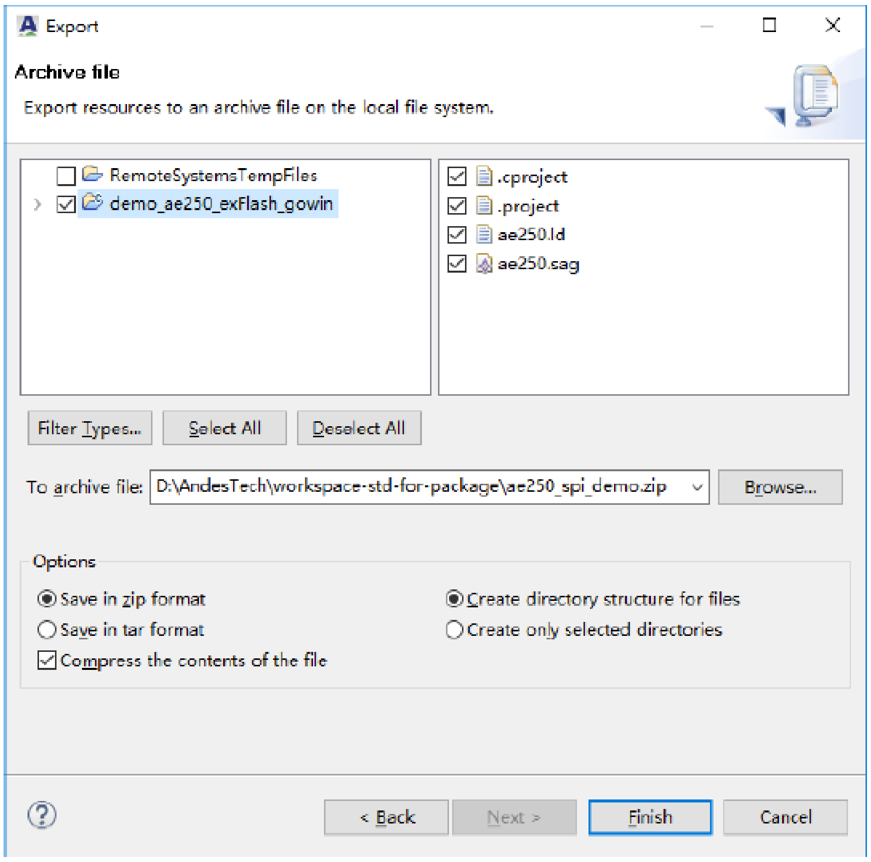


Figure 3-3 Import a Project

Select “Export… > Archive File” to open the export project interface, as shown in Figure 3-4. After selecting the project to be exported, compression format, save path, etc. you can complete the export.



Export

Archive file

Export resources to an archive file on the local file system.

- [ ] RemoteSystemsTempFiles
- [✓] demo_ae250_exFlash_gowin

- [✓] .cproject
- [✓] .project
- [✓] ae250.ld
- [✓] ae250.sag

To archive file: D:\AndesTech\workspace-std-for-package\ae250_spi_demo.zip

Options

[✓ Save in zip format] Save in tar format

[✓ Compress the contents of the file]

[✓ Create directory structure for files] Create only selected directories

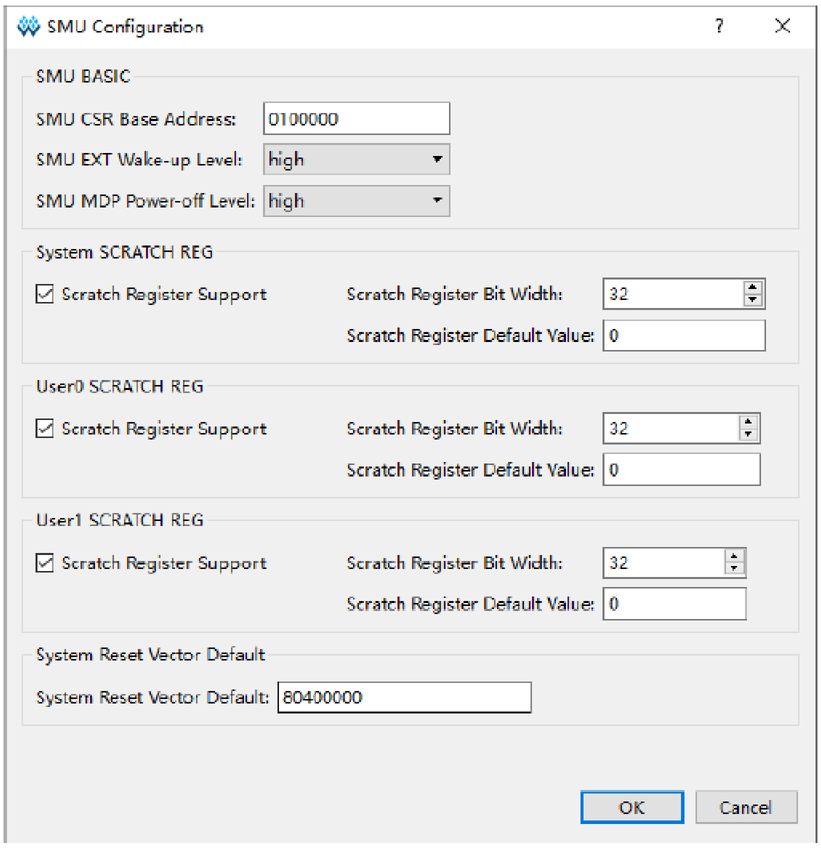
Figure 3-4 Export a Project

Download Programs to Flash

AE250 supports starting from Flash, then reads the embedded program from Flash via SPI interface and stores it in ILM, and then the embedded program is executed. The recommended method is to reuse SPI Flash that saves FPGA Bitstream; use the first half of Flash to save the FPGA Bitstream, and the remaining to save the binary files of embedded programs.

1. Open the IP core generator in Gowin Software and call AE250 RTL parameters. Double-click the SMU to open the SMU interface and set “System Reset Vector Default” to 0x80400000, as shown in Figure 3-5. Set the space of SPI Flash 0~0x400000 with a total of 4M bytes as the save address of Bitstream; starting from 0x400000 is used as the save address of binary files of embedded programs.

   Figure 3-5 System Reset Vector Default


SMU Configuration

SMU BASIC

SMU CSR Base Address: 0100000

SMU EXT Wake-up Level: high

SMU MDP Power-off Level: high

System SCRATCH REG

☑ Scratch Register Support

Scratch Register Bit Width: 32

Scratch Register Default Value: 0

User0 SCRATCH REG

☑ Scratch Register Support

Scratch Register Bit Width: 32

Scratch Register Default Value: 0

User1 SCRATCH REG

☑ Scratch Register Support

Scratch Register Bit Width: 32

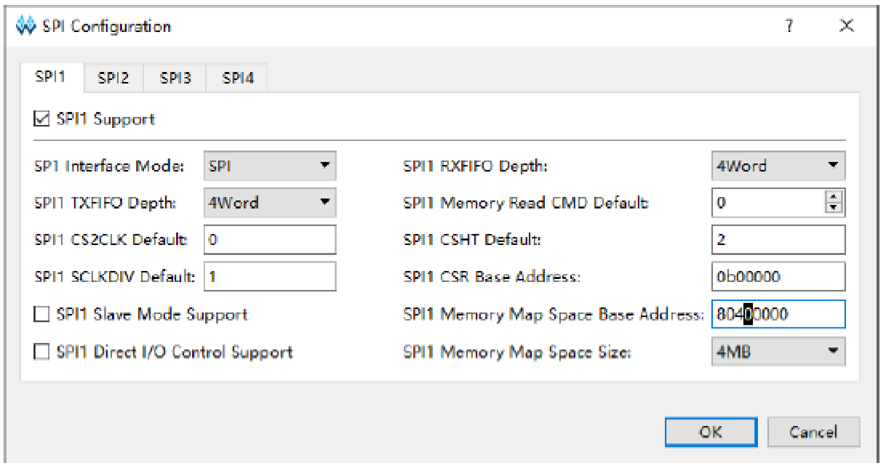
Scratch Register Default Value: 0

System Reset Vector Default

System Reset Vector Default: 80400000

2. Double-click SPI1 to open the SPI1 interface, check “SPI1 Support”, and set “SPI1 Memory Map Space Base Address” to 0x80400000, as shown in Figure 3 6.

Figure 3-6 SPI1 Configuration


SPI Configuration

SPI1 | SPI2 | SPI3 | SPI4

☑ SPI1 Support

SPI1 Interface Mode: SPI

SPI1 TXFIFO Depth: 4Word

SPI1 CS2CLK Default: 0

SPI1 SCLKDIV Default: 1

☐ SPI1 Slave Mode Support

☐ SPI1 Direct I/O Control Support

SPI1 RXFIFO Depth: 4Word

SPI1 Memory Read CMD Default: 0

SPI1 CSHT Default: 2

SPI1 CSR Base Address: 0b00000

SPI1 Memory Map Space Base Address: 80400000

SPI1 Memory Map Space Size: 4MB

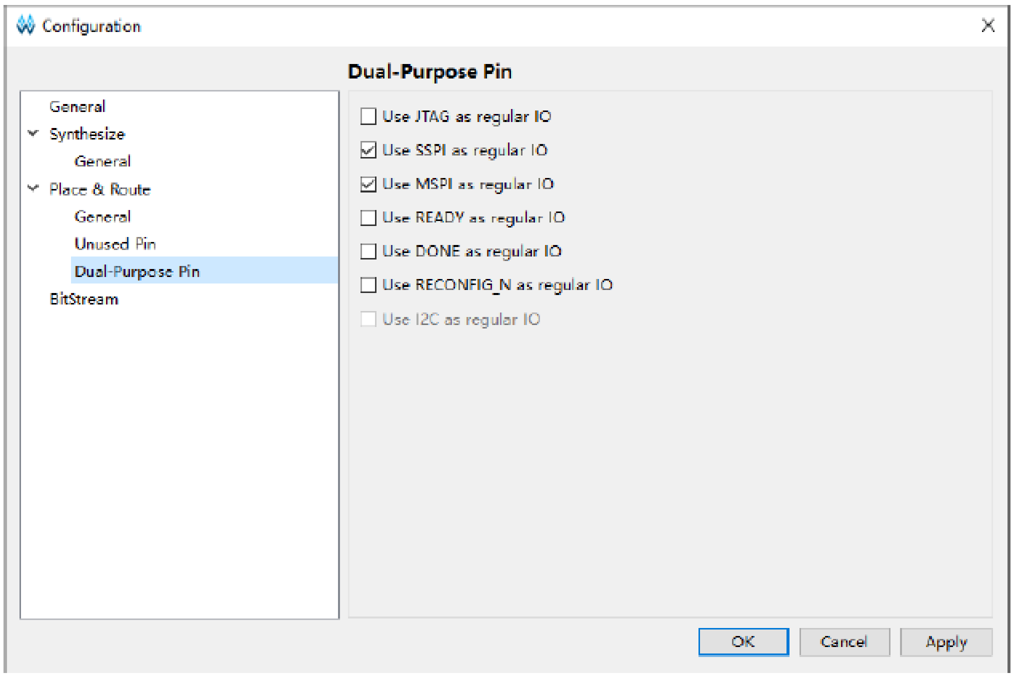
3. In the physical constraints of RTL design, the SPI1 interface should be connected to SPI Flash, and the SPI1 interface should be physically constrained according to the following table. For different FPGA chips, the location of MSPI interface is also different, and the constraint should be specific to the specific situation.

Table 3-1 SPI1 Interface Physical Constraints

AE250 SPI1 Interface	FPGA MSPI Interface
CSN	MCSN
CLK	MCLK
MISO	MSO
MOSI	MSI

4. Reuse MSPI interface as regular IO. In the “Process” window of Gowin Software, right-click “Place & Route”, select “Configuration” in the pop-up menu; select “Dual Purpose Pin” tab, and check “Use MSPI as regular IO” and click “OK” to finish placement and routing.

   Figure 3-7 Set MSPI Interface to Regular IO



Configuration

- General
- Synthesize
  - General
- Place & Route
  - General
  - Unused Pin
  - Dual-Purpose Pin
- BitStream

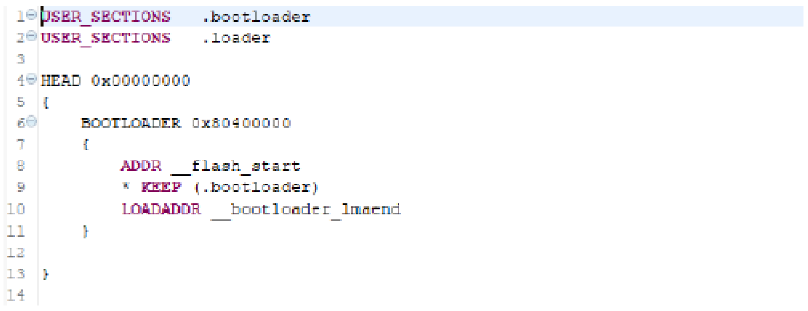
Dual-Purpose Pin

- [ ] Use JTAG as regular IO
- [✓] Use SSPI as regular IO
- [✓] Use MSPI as regular IO
- [ ] Use READY as regular IO
- [ ] Use DONE as regular IO
- [ ] Use RECONFIG_N as regular IO
- [ ] Use I2C as regular IO

5. Modify embedded program parameter settings. First, modify the parameters of bootloader in the linker script. Since the linker script in AE250 embedded program is automatically generated by SAG file, it should be modified in the SAG file. Open ae250.sag, find BOOTLOADER and modify it to the value of System Reset Vector Default in RTL design, as shown in Figure 3-8. Then modify config.h. Open src/bsp/config/config.h, and find the macro definition

   “BUILD_MODE” and modify it to “BUILD_BURN”.

   Figure 3-8 ae250.sag bootloader Parameters Setting



```
USER_SECTIONS   .bootloader
USER_SECTIONS   .loader

HEAD 0x00000000
{
    BOOTLOADER 0x80400000
    {
        ADDR __flash_start
        * KEEP (.bootloader)
        LOADADDR __bootloader_lmaend
    }

}
```

Note

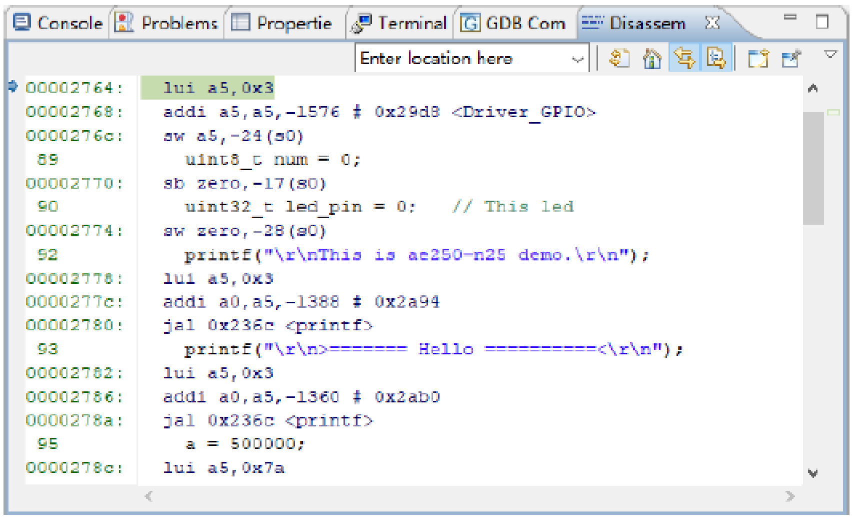
- The parameter should be consistent with the value of System Reset Vector Default of the RTL parameter.
- Modify the compilation settings; right-click the name of the embedded project, select Build Settings; select “Objcopy > General” tab, and uncheck “Disable”. (Do not auto-generate output file.)

Recompile the embedded program to generate binary files of the embedded project, and download the files to SPI Flash 0x400000 address using Gowin Programmer external Flash C Bin mode.
Synthesize and place & route the modified RTL design again, and download it to SPI Flash 0x000000 address

using Gowin Programmer external Flash mode.

On-chip Debug

After compilation, the compilation results of the embedded project can be downloaded to the development board for on-chip debug.
Modify config.h; open src/bsp/config/config.h, and find the macro definition BUILD_MODE; modify it to BUILD_LOAD, and recompile the embedded program.
Right-click on the project name in the Project Explorer, and select “Debug as > MCU Program “from the drop-down menu. For the first time, , a dialog box will pop up for setting “Debug Configuration”, as shown in Figure 3-9.

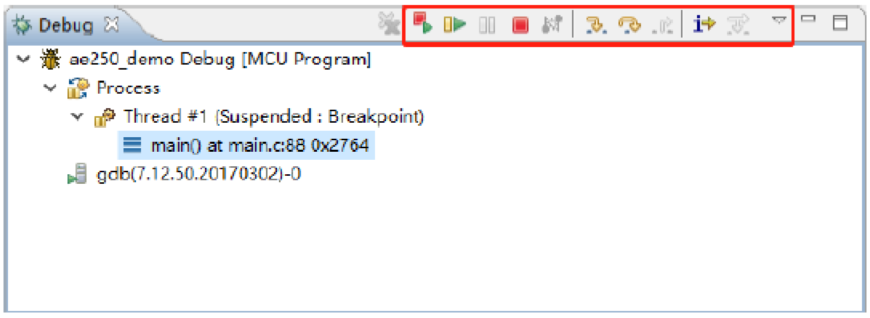


```
00002764:   lui a5,0x3
00002768:   addi a5,a5,-1576 # 0x29d8 <Driver_GPIO>
0000276c:   sw a5,-24(s0)
 89           uint8_t num = 0;
00002770:   sb zero,-17(s0)
 90           uint32_t led_pin = 0;   // This led
00002774:   sw zero,-28(s0)
 92           printf("\r\nThis is ae250-n25 demo.\r\n");
00002778:   lui a5,0x3
0000277c:   addi a0,a5,-1388 # 0x2a94
00002780:   jal 0x236c <printf>
 93           printf("\r\n>======= Hello ==========<\r\n");
00002782:   lui a5,0x3
00002786:   addi a0,a5,-1360 # 0x2ab0
0000278a:   jal 0x236c <printf>
 95           a = 500000;
0000278c:   lui a5,0x7a
```

Figure 3-9 Debug Configurations

In the “Startup” tab, check “Reset and Hold” option to stop the program before executing the first instruction. Enter load in the parameter box below this option to download the compilation results of the embedded project into the ILM before on-chip debug.
In “Runtime Options”, check “Set breakpoint at”. Enter a label, such as main in the input box. It can set a breakpoint at the beginning of the main function. Check “Resume”, and it will start the continuous operation directly after entering on-chip debug.
When entering on-chip debug, it automatically goes to the debug view and an area will be displayed, as shown in Figure 3-10. This area is the operation area for on-chip debug. Some shortcut buttons for debug are shown in the red box. From left to right, they mean restart DEBUG, continue to run, suspend, end, disconnect, link to one process, step into, step over, step return, and instruction stepping mode; in this mode, each time it runs a risc – v assembly instruction, otherwise each time it runs a C statement.

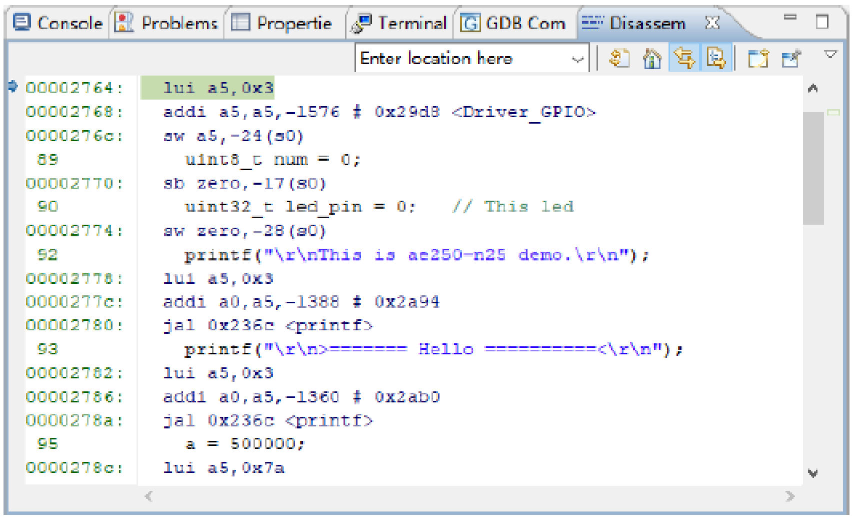
Note!
The grayed icons mean that they are unavailable at this time.
Double click the left on the line number in the code text to quickly set breakpoints or cancel breakpoints, and right click in the code text to select “run to line” from the pop-up menu.



```
ae250_demo Debug [MCU Program]
  Process
    Thread #1 (Suspended : Breakpoint)
      main() at main.c:88 0x2764
  gdb(7.12.50.20170302)-0
```

Figure 3-10 Debug Buttons Introduction

Figure 3-11 is an assembly statements window that displays the contents of assembly instructions running in real time in ILM.

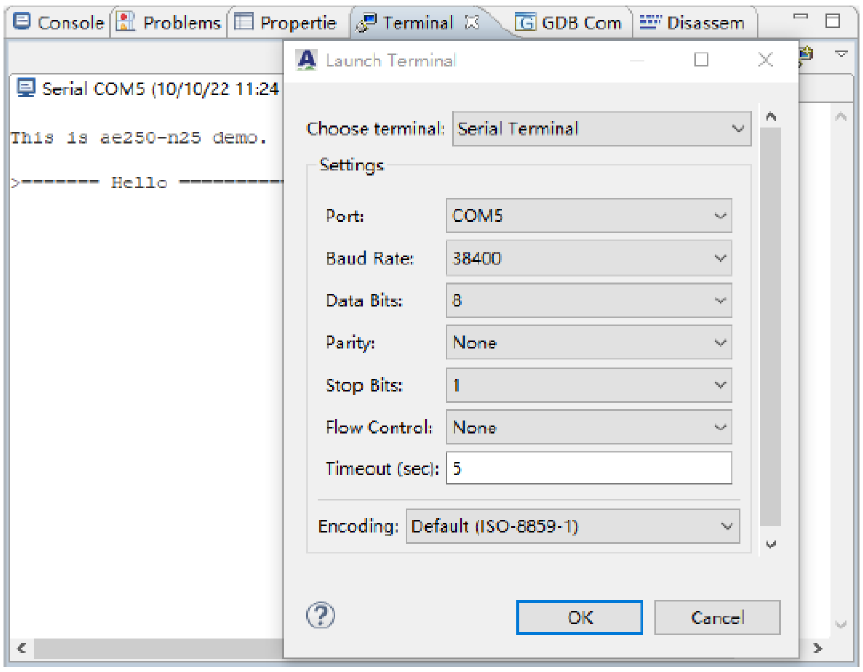


```
00002764:   lui a5,0x3
00002768:   addi a5,a5,-1576 # 0x29d8 <Driver_GPIO>
0000276c:   sw a5,-24(s0)
 89           uint8_t num = 0;
00002770:   sb zero,-17(s0)
 90           uint32_t led_pin = 0;   // This led
00002774:   sw zero,-28(s0)
 92           printf("\r\nThis is ae250-n25 demo.\r\n");
00002778:   lui a5,0x3
0000277c:   addi a0,a5,-1388 # 0x2a94
00002780:   jal 0x236c <printf>
 93           printf("\r\n>======= Hello ==========<\r\n");
00002782:   lui a5,0x3
00002786:   addi a0,a5,-1360 # 0x2ab0
0000278a:   jal 0x236c <printf>
 95           a = 500000;
0000278c:   lui a5,0x7a
```

Figure 3-11 Assembly Instruction Code Window

RDS Built-in Serial Terminal Usage

Figure 3-12 shows the UART Terminal built in RDS interface. If you need to use, click “Window > Show View > Terminal” in the top menu to open “Terminal” window, and then click “open a terminal” to create a new serial terminal. After setting the port number (which can be viewed in the hardware manager), baud rate and other parameters, click “OK” to start using.



Serial COM5 (10/10/22 11:24

```
This is ae250-n25 demo.

>======= Hello ==========
```

Choose terminal: Serial Terminal

Settings

Port: COM5

Baud Rate: 38400

Data Bits: 8

Parity: None

Stop Bits: 1

Flow Control: None

Timeout (sec): 5

Encoding: Default (ISO-8859-1)

Figure 3-12 RDS Built-in Serial Terminal

For the details, see the document AndeSight_RDS_v3.1_User_Manual_UM170_V1.0.pdf, which can be found in the doc path of the installation directory.

Reference Design

Project Code

The key files in the AE250 embedded project template are as follows:

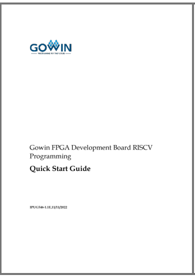
1. src/bsp/ae250/ae250.h: This file contains the system clock definition, peripheral register definition, peripheral register address mapping definition, and interrupts source number definition. The clock definition must be consistent with the AE250 parameters configuration.
2. src/bsp/ae250/ae250.c: The reset_handler function is the entry to start the embedded program. In the entry, UART initialization is performed before the main function is executed. The required UART port is selected and the required baud rate is configured according to the parameter configuration of AE250.
3. src/bsp/ae250/interrupt.c: This file is the definition of interrupt handler functions of AE250
4. src/bsp/config/config.h: This file contains the macro definition that control compilation method. #define BUILD_MODE can be defined as BUILD_LOAD or BUILD_BURN. BUILD_LOAD means that the program is loaded directly into ILM, and it is generally used when debugging. BUILD_BURN means the program is downloaded to SPI Flash, and the program is read from SPI Flash to ILM first after power on, and then run, which is applicable to release version program.
5. Start.S: The starter file written in assembly language.
6. src/bsp/loader.c: bootloader file, which is used to start from SPI Flash.
7. ae250.sag: Sag is the scattering-and-Gathering format script. It's used to generate linker script. It should be noted that the memory map parameters in ae250.sag need to be consistent with those in AE250.
8. src/bsp/driver: This directory contains two folders, ae250 is AE250 driver code, include is the call interface of driver functions.
9. src/bsp/lib: It contains two files. In printf.c, the form of subfunction in C standard library is redefined to output printf information through UART. In read.c, there is a simple function to read input information through UART.

Reference Design

After the installation, several basic reference designs can be found in the demo folder of the installation directory or in the reference design zip at the website; the reference design can be loaded into RDS for trial, debugging and redeveloping by the way of importing. The reference designs are shown as follows:

1. ae250_demo: Demonstrates UART input/output and GPIO output of the AE250.
2. ae250_plic: Demonstrates the response of the interrupt controller to interrupts, and provides demonstrations of the machine timer and pit timer.
3. ae250_freertos: Demonstrates that the AE250 ports embedded
   real-time operating system FreeRTOS multi-threading running program.
4. ae250_ucosiii: Demonstrates that the AE250 ports embedded real-time operating system uC/OS-III multi-threading running program.

Documents / Resources

 Gowin FPGA Development Board RISCV Programming **Quick Start Guide** IPUG546-1.1E,11/11/2022	**GOWIN FPGA Development Board RISCV Programming** [pdf] User Guide FPGA Development Board RISCV Programming, Board RISCV Programming, FPGA Development RISCV Programming, RISCV Programming, Board RISCV

References

- AICE-MINI+ - Andes Technology