Usage instructions for the compressed file

The development environment for the software code is Libero SoC v11.8 SP2. The design is implemented using a mix of VHDL and Verilog. When opening the project file within a compressed file from the author's GitHub repository in Libero, you can see the nine code modules shown in Fig. 1. The function of each module is briefly introduced as follows.

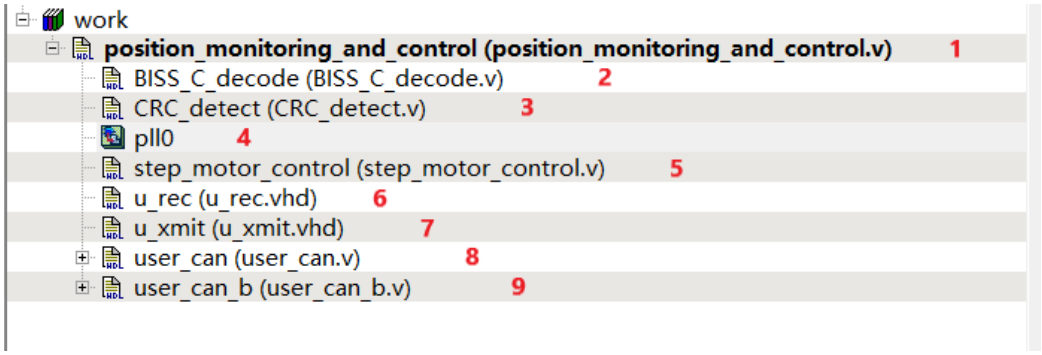

Fig. 1. Code module.

Code Module 1: The top-level module in an FPGA software design acts as the entry point and central coordinator, responsible for connecting and organizing the entire design. It connects all other modules within the FPGA and coordinates their functions. The main functions are summarized as follows:

Submodule Instantiation: It instantiates all other modules in the design, essentially "wiring" them together.

Interconnection: It defines the signals and connections between submodules, enabling communication and data exchange.

I/O Interface: It manages the FPGA's external interface, connecting it to the external world via input/output (I/O) pins. This includes receiving input signals and sending output signals.

Top-Level Logic: While primarily a coordinator, the top-level module may also contain some high-level logic that doesn't naturally fit into any submodule. However, best practice generally emphasizes keeping the top-level module clean and simple, focusing on interconnection and minimizing complex logic.

Clock and Reset Distribution: Typically, the top-level module is responsible for distributing clock and reset signals to all other modules, ensuring proper synchronization and initialization.

In short, the top-level module is the “glue” that binds the entire code design together, defining the system's overall structure and behavior. It is the first module that the FPGA tools “see” when compiling the design, and it is responsible for mapping the intended functionality onto the FPGA hardware.

Code Module 2: It is used to drive a linear encoder and receive data from the linear encoder.

Code Module 3: It is used to detect errors that occur during data transmission or storage.

Code Module 4: Taking a stable clock as its input, it generates clocks of various frequencies and phases, meeting the needs of individual modules within the FPGA, optimizing performance, and ensuring the stable operation of the entire system.

Code Module 5: It is a stepper motor driver module that implements open-loop and closed-loop focusing functions.

Code Module 6: It is a receiving module for serial communication.

Code Module 7: It is a transmitting module for serial communication.

Code Module 8: It is a transceiver module for CAN communication on channel *A*.

Code Module 9: It is a transceiver module for CAN communication on channel *B*.

FPGA software code debugging and testing is highly hardware-dependent. You may have difficulty replicating our work using only the project file within a compressed file from the author’s GitHub repository. Co-debugging the software and hardware is strongly recommended to avoid problems.