This code defines a hierarchical design for a pedestrian crossing traffic light controller. The design includes modulo counters for various counting tasks and a finite state machine (FSM) to control the traffic light behavior. Overview:

1. **mod13counter**: This module implements a 4-bit modulo-13 up counter. It counts from 0 to 12, resetting when the counter reaches 12 or when a reset signal is asserted.
2. **mod8counter**: This module implements a 3-bit modulo-8 up counter. It counts from 0 to 7, resetting when the counter reaches 7 or when a reset signal is asserted.
3. **mod2counter**: This module implements a 1-bit modulo-2 up counter. It toggles between 0 and 1, resetting to 0 when a reset signal is asserted.
4. **FSM**: This module represents the finite state machine that controls the traffic light behavior based on the states and transitions defined. It uses the outputs from the modulo counters to make decisions about state transitions and signal outputs for the traffic lights.
5. **controller**: This module acts as the highest-level controller hierarchy. It connects the FSM module to inputs such as clock, reset, and pedestrian request signals, and receives the outputs to control the traffic lights.

The FSM follows a state-based approach to control the traffic light behavior. Each state corresponds to a specific traffic light configuration, and transitions between states occur based on the conditions specified in the code. The **always** blocks are used to define the behavior of the FSM based on the current state and various inputs, including signals from the modulo counters.

The project code was based on a Moore FSM. To implement it in a module, we use the typical format we have learned that is divided into three always procedural blocks. Kindly refer to the submitted state diagram for more details.
This FSM works around 11 states that are one-hot encoded for simpler circuit implementation. It also uses a single input instead of two for simpler code.
The first always block checks for the reset signal. If asserted, the FSM is in stateA. If not, the FSM is ready to go to the nextState.
The second always block is responsible for the nextState. It discusses the different cases for that, while relying on the currentState, the input, or the number of clock cycles via counter instances.
The third final always block is responsible for the five outputs of the traffic and pedestrian lights controller. It assigns the values for each type of light depending on the state the FSM is in.

In the end, we created a separate module for the controller in which we declare a wire corresponding to SB or NB since we have two inputs. We also use an instance of the FSM module we have previously made.

That’s how the project Verilog code was completed. It has been tested thoroughly and works perfectly.