The CPUrepresents a finite state machine (FSM) for a CPU module, implemented in VHDL. The FSM controls a game between two players, where the players interact through various inputs and the system processes these interactions to determine the game state. Here’s a detailed explanation of the module:

### Overview

The FSM model consists of two concurrent processes:

1. State Registers (state\_reg): Handles the current state of the FSM based on the clock and reset signals.
2. Combinational Logic (comb\_logic): Describes the state transitions and outputs based on the current state and inputs.

### Entity Declaration

The cpu entity defines the module’s interface, including its inputs and outputs:

* Inputs:
  + clock, reset: Basic control signals.
  + clk\_1ms, clk\_1hz: Clock signals at different frequencies.
  + player\_A, player\_B: Inputs from the players.
  + start, target\_confirm: Control signals for starting the game and confirming the target score.
* Outputs:
  + target\_score, test\_cycles, score\_playerA, score\_playerB: Various score and cycle counters.
  + config\_enable\_o: Configuration enable signal.
  + p1\_Hex5\_4, p2\_Hex5\_4: Outputs to display player status.
  + stimulus\_playerA, stimulus\_playerB: Stimulus signals for the players.

### Architecture

The architecture of the cpu entity is named FSM and consists of several components and internal signals.

#### State Types

Two state types are defined to manage the main FSM states and a 5-second delay FSM:

* state\_type: Represents the main states of the FSM, including IDLE, CONFIG, WAIT\_for\_START, DT\_P1, STIMULUS1, WP1, SAVE\_P1, DT\_P2, STIMULUS2, WP2, SAVE\_P2, COMPARE, CHECK\_TARGET, WINNER, Penalty, SP1, and SP2.
* state\_type\_5s: Represents the states for the 5-second delay FSM, including IDLE\_5s, RUN\_5s, and STOP\_5s.

#### Internal Signals

Several internal signals are declared to hold intermediate values and states, such as:

* Registers for storing scores (target\_score\_reg, test\_cycles\_reg, score\_playerA\_reg, score\_playerB\_reg).
* Control signals (config\_enable, config\_done, timer\_enable, timer\_done, stopwatch\_enable1, stopwatch\_enable2).
* Timer and delay signals for managing time-related operations.

#### Component Declarations

Three components are instantiated within the architecture:

1. fsm\_configuration: Handles configuration settings.
2. random\_delay: Generates random delays.
3. timer: Manages the stopwatch functionality for both players.

#### Processes

1. state\_reg Process:
   * Sensitive to clock and reset.
   * Updates the current\_state based on the next\_state.
2. comb\_logic Process:
   * Sensitive to various inputs and the current\_state.
   * Uses a case statement to determine the next\_state based on the current state and inputs.
   * Handles state transitions and updates signals accordingly, such as enabling timers, updating scores, and generating stimuli for the players.
3. second\_delay Process:
   * Manages a delay of 31 milliseconds.
   * Uses a counter to generate the delay and sets sec\_done when the delay is complete.
4. five\_sec\_delay Process:
   * Manages a 5-second delay.
   * Transitions through IDLE\_5s, RUN\_5s, and STOP\_5s states to count 5 seconds and sets sec5\_done when the delay is complete.
5. Penalty Counters (penalty\_countP1, penalty\_countP2):
   * Increment penalty counters for each player if they incur penalties (SP1 and SP2 states).

### Outputs

The final block maps internal signals to the output ports of the entity, ensuring that the external interface correctly reflects the internal state of the FSM.

### Summary

This CPU module implements a complex FSM to manage a game involving two players, with various states representing different stages of the game. It handles configurations, delays, and interactions from the players, updating scores and generating stimuli based on the game's rules. The FSM is designed to be responsive to inputs and control signals, ensuring smooth transitions and accurate state management throughout the game.