Rename Record Table Module Documentation

Overview

This document describes the architecture of the Rename Record Table module, a parameterizable component implemented in Verilog. The module is designed for applications requiring the sequential processing and temporary storage of data, including inter-process communication and buffering across asynchronous interfaces. The design prioritizes reliability, easy integration, and straightforward verification.

I/O Signals

- **clk (input)**: Clock signal to which all operations are synchronized.
- **reset (input)**: Asynchronous reset signal, resets the table state when asserted.
- **write_enable (input)**: When high, allows data on the `data_in` line to be stored into the table if it is not full.
- **read_enable (input)**: When high, the data is read from the table if it is not empty.
- **data_in (input)**: Input bus carrying the data to be stored, with a width defined by `DATA_WIDTH`.
- **data_out (output)**: Output bus delivering data read from the table, also defined by `DATA_WIDTH`.
- **table_full (output)**: Indicates whether the table has reached its capacity.
- **table_empty (output)**: Indicates whether the table contains no elements.

Architecture-Level Registers and Memories

- **table_memory**: An array sized by `TABLE_DEPTH`, where each entry is `DATA_WIDTH` bits wide. This array stores data elements within the table.
- **write_pointer**: A register that indicates the next write position within `table_memory`, sized to address all entries.
- **read_pointer**: A register that tracks the next read position within the table, similarly sized to `write_pointer`.
- **element_count**: A register counting the number of elements currently stored in the table, aiding in determining `table_full` and `table_empty` statuses.

Functionality

- **Write Operation**: On a high `write_enable` and when the table is not full, data from `data_in` is stored at the location pointed by `write_pointer`, which is subsequently incremented, and the `element_count` is updated.
- **Read Operation**: On a high `read_enable` and when the table is not empty, data is retrieved from the position indicated by `read_pointer` and outputted to `data_out`.

Following the read, `read_pointer` is incremented, and the `element_count` is decreased.

- **Reset Behavior**: Asserting the `reset` signal resets `write_pointer`, `read_pointer`, and `element_count` to their initial states, clearing the table.

Reset Logic

The module employs an asynchronous reset strategy for `write_pointer`, `read_pointer`, and `element_count`, ensuring that the table can be quickly initialized or cleared irrespective of the clock state.

Integration and Verification

The Rename Record Table module can be seamlessly integrated into larger systems and verified through various test cases, such as underflow, overflow, and data integrity during sequential operations. Specific assertions are recommended to validate operational integrity and address potential data mismatches during the write and read processes.