*Project: RMC75E FPGA TEST BENCH*

*Module: DIO8.vhd*

*Author: Satchel Hamilton*

*Company: Delta Motion*

*Date: 7/26/2023*

*Last updated: July 13, 2023*

The DIO8 module is an 8-bit digital input/output (DIO) interface designed to work with multiple expansion slots. It provides access to four DIO8 modules, each containing eight bidirectional digital I/O channels. The module operates on a clocked architecture and includes a state machine that controls the read and write sequences to the DIO8 modules.

Key features and functionality of the DIO8 module include:

1. Input and Output Interfaces: The module contains separate input and output interfaces. The input interface receives data from the external devices through the Exp0D8\_DataIn, Exp1D8\_DataIn, Exp2D8\_DataIn, and Exp3D8\_DataIn lines. The output interface sends data to external devices through the d8DataOut line.
2. Configuration Control: The module supports configuration control through the ExpDIO8ConfigRead and ExpDIO8ConfigWrite input signals. These signals allow the CPU to read and write configuration data to the dual-port memory in the DIO8 module.
3. Data Storage: The module features dual-port memory for storing data from external devices and the CPU. The registers, D8OutputReg0, D8OutputReg1, D8OutputReg2, and D8OutputReg3, store output data, while D8InputReg0, D8InputReg1, D8InputReg2, and D8InputReg3 hold input data.
4. State Machine: The core of the module is a state machine that sequences through the read and write operations. The state machine controls various aspects of the module, such as loading data into registers, shifting data in and out, and managing the read and write states. The state machine transitions through different states such as IdleState, LoadState1, LoadState2, LoadState3, s3\_LoadShiftState, s4\_ShiftIOState, s5\_SRAMWriteState, s6\_SRAMReadState, s7\_LoadShiftState, s8\_ShiftOutState, and s9\_EndState.
5. Clocking: The module uses multiple clocks, including the system clock (SysClk), a synchronized tick (SynchedTick), and an internal clock (intExpD8\_Clk).
6. Expansion Slot Selection: The module supports four expansion slots (Exp0, Exp1, Exp2, Exp3). The ExpSlot signal is used to select the desired expansion slot for read and write operations.
7. Slow Enable: The SlowEnable signal enables a 3.75MHz clock output for serial communication to control the read and write sequences to the DIO8 modules.
8. Shift Register: The module utilizes shift registers for data loading and shifting operations.

Overall, the DIO8 module provides a versatile and configurable interface for handling digital input and output operations. It is well-suited for applications that require multiple DIO8 modules to be accessed through different expansion slots. The state machine-based design ensures efficient handling of read and write operations, making it suitable for real-time applications with strict timing requirements.

**DIO8** is responsible for handling discrete input/output functionality. It defines various input and output ports, including control signals, data signals, and configuration signals.

The architecture **DIO8\_arch** implements the behavior of the **DIO8** entity. Let's go through the code to understand its functionality:

1. State Encoding: The code defines a custom type called **STATE\_TYPE** which represents different states of the state machine used in the code.
2. Signals and Registers: Various signals and registers are defined to hold and manipulate the data during the operation of the module. These include **State** (representing the current state of the state machine), **ExpSlot** (selects the expansion slot location), **InputShiftRegister** (holds the input data for shifting), **OutputShiftRegister** (holds the output data for shifting), and several others.
3. Multiplexer: There is a multiplexer section that maps the appropriate signals to the respective expansion slot based on the value of **ExpSlot**. It controls the data flow between the different DIO8 modules in the expansion slot stack-up.
4. Data Output: The **d8DataOut** signal represents the data output lines. Depending on the values of **ExpDIO8DinRead** and **ExpDIO8ConfigRead**, it combines the input data registers and output data registers to generate the output data.
5. Register Write: The process labeled **RESET, H1\_CLKWR** handles the write operation to the output registers (**D8OutputReg0**, **D8OutputReg1**, **D8OutputReg2**, **D8OutputReg3**) when the **H1\_CLKWR** signal rises. The values to be written are taken from **intDATA** based on the values of **ExpDIO8ConfigWrite**.
6. Register Mapping: The process labeled **ExpSlot, D8OutputReg0, D8OutputReg1, D8OutputReg2, D8OutputReg3** maps the appropriate output register based on the **ExpSlot** value to the **IntDout** signal. It ensures that the correct register is selected for reading.
7. Register Read: The process labeled **rising\_edge(SysClk)** handles the read operation from the input registers (**D8InputReg0**, **D8InputReg1**, **D8InputReg2**, **D8InputReg3**) when **IntWrite** is active and the corresponding **ExpSlot** value matches. It loads the data from the input shift register (**InputShiftRegister**) into the appropriate input register.
8. Data Output Assignment: The **d8DataOut** signal is assigned the value of the **OutputShiftRegister(15)** for the purpose of data output.
9. State Machine: The process labeled **RESET, SysClk** implements the state machine functionality. It defines the behavior of the state transitions based on the current state (**State**) and various control signals.
   * The state machine starts in the **IdleState** and waits for a rising edge of **SysClk**.
   * The state machine progresses through different states such as **LoadState1**, **LoadState2**, **LoadState3**, **s3\_LoadShiftState**, **s4\_ShiftIOState**, **s5\_SRAMWriteState**, **s6\_SRAMReadState**, **s7\_LoadShiftState**, **s8\_ShiftOutState**, and **s9\_EndState**.
   * The state transitions are controlled by conditions based on control signals such as **SynchedTick**, **SlowEnable**, **ExpDIO8ConfigWrite**, and **ExpSlot**.
   * The state machine manages the loading, shifting, writing, and reading of data registers and controls the flow of data within the module.