Lab 7: FIFO Design

Synchronous FIFO

Objective

The objective of this lab is to design and verify a **parameterizable synchronous FIFO (First-In-First-Out) buffer** using SystemVerilog. The FIFO should support configurable data width and depth, generate full/empty status flags, provide almost-full/almost-empty thresholds, and efficiently utilize FPGA resources.

Design Specifications

1. **Parameterizable FIFO**
    - Data width: Configurable (default 8 bits).
    - Depth: Configurable (default 16 entries).
2. **FIFO Operations**
    - Synchronous read and write operations.
    - Support simultaneous read and write in the same clock cycle.
3. **Status Flags**
    - **Full flag**: Indicates when FIFO is full.
    - **Empty flag**: Indicates when FIFO is empty.
    - **Almost Full flag**: Asserts when FIFO occupancy reaches a threshold (default 14 entries).
    - **Almost Empty flag**: Asserts when FIFO occupancy falls below a threshold (default 2 entries).
4. **Counter and Pointers**
    - Read and write pointers track positions in memory.

- A counter keeps track of current occupancy.

Design Methodology

1. **Pointer Selection**
    - Binary counters are used for read and write pointers.
    - Wrap-around is handled by resetting pointers to zero when reaching the maximum depth.
2. **Memory Implementation**
    - FIFO storage is implemented using an array, which infers block RAM.
3. **Flag Logic**
    - Full: Activated when count = depth – 1.
    - Empty: Activated when count = 0.
    - Almost Full: Activated when count ≥ threshold (e.g., 14).
    - Almost Empty: Activated when count ≤ threshold (e.g., 2).
4. **Simultaneous Read/Write**
    - Special handling ensures that when both operations occur, the FIFO occupancy count remains unchanged.

Key Design Decisions

- **Binary Counters vs Gray Code**: Binary counters were chosen for simplicity since the design is fully synchronous.
- **Registered vs Combinational Flags**: Status flags are generated combinationally to provide immediate updates.
- **Power-of-2 Depth**: A depth of 16 entries was chosen for easier pointer wrap-around and efficient FPGA synthesis.

Simulation and Verification

The FIFO was tested using a structured testbench with the following sequence:

1. **Reset Phase**: FIFO pointers and count initialized.
2. **Write Operations**: Random data written into FIFO until partially filled.
3. **Read Operations**: Several entries read back to check order (FIFO property).
4. **Full Condition Test**: FIFO written until full flag asserted.
5. **Empty Condition Test**: FIFO read until empty flag asserted.
6. **Threshold Tests**: Verified almost-full and almost-empty behavior.

The simulation confirmed correct operation of:

- Data storage and retrieval order (First-In-First-Out).
- Accurate full/empty detection.
- Proper flag assertion for almost-full and almost-empty conditions.

Results

- The FIFO correctly handled parameterized data width and depth.
- Status flags were asserted at appropriate times.
- The design supported simultaneous read and write without errors.
- Efficient FPGA synthesis was achieved using inferred memory.

Conclusion

The synchronous FIFO design met all specified requirements. It demonstrated reliable operation under varying read/write conditions, correct flag generation, and proper handling of boundary cases such as full and empty states.

This lab provided practical experience with:

- FIFO memory design.

- Pointer-based addressing.
- Status flag generation.
- Verification through simulation.