**Overview**

This project implements an asynchronous FIFO (First-In-First-Out) memory with dual-clock domains. The FIFO allows data transfer between different clock domains using Gray code-based pointers for synchronization.

**Features**

Parameterizable Data Size (DSIZE) and Address Size (ASIZE)

Dual-clock asynchronous operation

Gray code pointer synchronization for metastability prevention

Full and Empty flag generation

Support for vendor-specific RAM instantiation (via VENDORRAM macro)

**Module Descriptions**

fifo1

This is the top-level FIFO module that instantiates:

sync\_r2w: Synchronizes the read pointer (rptr) into the write clock domain.

sync\_w2r: Synchronizes the write pointer (wptr) into the read clock domain.

fifomem: Memory storage for FIFO.

rptr\_empty: Generates the empty flag and manages the read pointer.

wptr\_full: Generates the full flag and manages the write pointer.

Fifomem

Implements a simple dual-port memory using an RTL memory model (or a vendor RAM instance if VENDORRAM is defined).

Stores DSIZE-bit wide data.

Indexed using ASIZE-bit wide addresses.

Read and write are controlled by wclken and wfull.

sync\_r2w

Uses two-stage synchronizers to transfer rptr safely into the wclk domain.

Helps maintain FIFO stability across clock domains.

sync\_w2r

Uses two-stage synchronizers to transfer wptr safely into the rclk domain.

rptr\_empty

Tracks the read pointer (rptr) and checks for FIFO empty conditions.

Generates the empty flag when rptr matches the synchronized wptr.

wptr\_full

Tracks the write pointer (wptr) and checks for FIFO full conditions.

Generates the full flag using a comparison between wptr and rptr in Gray code.

Parameters

Parameter

Description

DSIZE

Data width of FIFO (default: 80 bits)

ASIZE

Address width of FIFO (default: 6 bits)

Ports

FIFO Top-Level (fifo1)

Signal

Direction

Description

rdata

Output

Data read from FIFO

wfull

Output

FIFO full flag

rempty

Output

FIFO empty flag

wdata

Input

Data to write into FIFO

winc

Input

Write enable signal

wclk

Input

Write clock domain

wrst\_n

Input

Write clock reset (active low)

rinc

Input

Read enable signal

rclk

Input

Read clock domain

rrst\_n

Input

Read clock reset (active low)

Usage Example

fifo1 #(.DSIZE(80), .ASIZE(6)) fifo\_inst (

.wclk(write\_clk),

.rclk(read\_clk),

.wrst\_n(~rst),

.rrst\_n(~rst),

.winc(write\_en),

.rinc(read\_en),

.wdata(data\_in),

.rdata(data\_out),

.wfull(full),

.rempty(empty)

);

**Simulation**

To test the FIFO, a testbench (fifo\_testbench.sv) is provided, which:

Generates write and read clocks.

Uses DPI-C function request\_packet(pkt) to provide random data.

Writes data into FIFO and verifies if it is read correctly