DESIGN SPEC DOCUMENT

ECE-593: Fundamentals of Pre-Silicon Validation

Maseeh College of Engineering and Computer Science

Winter, 2025

Project Name:

Design and Verification of an Asynchronous FIFO

Members:

Pavan Gaddam, Gene Hu, Surya Teja Purma, Kai Roy

Date:

4/20/2025

Project Name	Design and Verification of an Asynchronous FIFO
Location	Portland State University
Start Date	4/10/2025
Estimated Finish Date	5/25/2025
Completed Date	N/A

Prepared by: Team 5
Prepared for: Prof. Venkatesh Patil

Team Member Names	Emails
Pavan Gaddam	pavang@pdx.edu
Gene Hu	ghu@pdx.edu
Surya Teja Purma	purma@pdx.edu
Kai Roy	roykai@pdx.edu

Design Features
Supports data widths up to 256 bits
Supports memory depths of up to 65,535 locations
Memory may be implemented in either SelectRAM+ or Distributed RAM
Fully synchronous and independent clock domains for the read and write ports
Supports FULL and EMPTY status flags
Optional ALMOST_FULL and ALMOST_EMPTY status flags
Invalid read or write requests are rejected without affecting the FIFO state
Four optional handshake signals (WR_ACK, WR_ERR, RD_ACK, RD_ERR) provide feedback (acknowledgment or rejection) in response to write and read requests in the prior clock cycle

Optional count vector(s) provide visibility into number of data words currently in the FIFO, synchronized to either clock domain

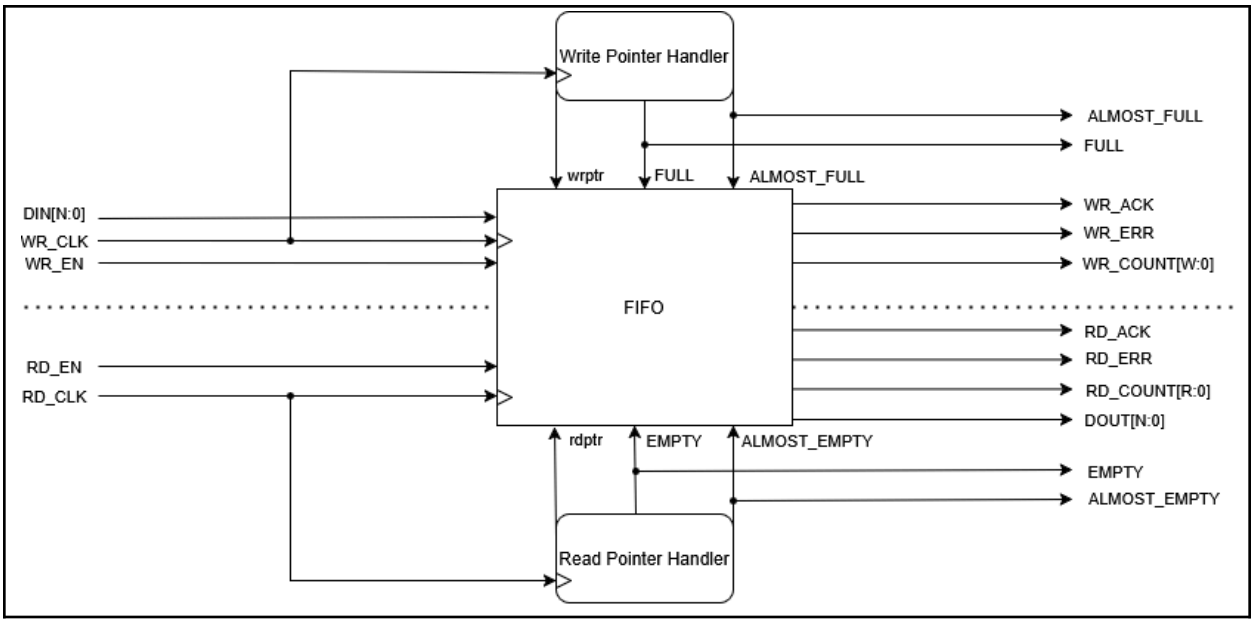
Project Description
The team will implement an asynchronous FIFO using SystemVerilog, and perform thorough testing to confirm correct functionality in accordance with design features. There will be a functional test plan drafted which will assign responsibilities to team members for their part in the verification process. These responsibilities include the DUT and its components, design documentation, testbench development, verification plan, writing tests, running simulations, generating reports, and debugging.

Important Signals/Flags		
Signal Name	Signal Direction	Description
DIN[N:0]	input	Data input
WR_EN	input	Write enable (request)
WR_CLK	input	Clock for write domain operations (rising edge)
RD_EN	input	Read enable (request)
RD_CLK	input	Clock for read domain operations (rising edge)
AINIT	input	Asynchronous reset of all FIFO functions, flags, and pointers. The four FIFO flags will reset to active high, and deactivated on their next respective clock pulse
FULL	output	Full: no additional writes can be performed, synchronous to WR_CLK
ALMOST_FULL	output	Almost Full: only one additional write can be performed before FIFO is FULL, synchronous to WR_CLK
WR_COUNT[W:0]	output	Write Count: count vector (unsigned binary) representing the number of data words currently in FIFO, synchronized to WR_CLK. If 2^(W+1) < [FIFO depth +1], the least significant bits of count are truncated. (W=0 produces a half-full flag)
WR_ACK	output	Write Acknowledge: hand-shake signal indicates that data was written to the FIFO on the previous CLK edge while WR_EN was active

WR_ERR	output	Write Error: handshake signal indicates that no data word was written to the FIFO on the previous CLK edge while WR_EN was active
DOUT[N:0]	output	Data Output: synchronous to RD_CLK
EMPTY	output	Empty: no additional reads can be performed, synchronous to RD_CLK
ALMOST_EMPTY	output	Almost Empty: only one additional read can be performed before FIFO is EMPTY, synchronous to RD_CLK
RD_COUNT[R:0]	output	Read Count: count vector (unsigned binary) representing the number of data words currently in FIFO, synchronized to RD_CLK. If (2^R+1)<(FIFO depth+1), the least significant bits of count are truncated (R=0, produces a half-full flag)
RD_ACK	output	Read Acknowledge: hand-shake signal indicates that data was read from the FIFO and placed on the DOUT output pins on the previous CLK edge while RD_EN was active
RD_ERR	output	Read Error: handshake signal indicates that no data word was read from the FIFO on the previous CLK edge while RD_EN was active and subsequently data on DOUT output pins was not updated
DOUT[N:0]	output	Data output

Design Signals	
Signal Name	Description
wrptr	Pointer from the write pointer handler for keeping track of where in memory the FIFO should place its next write
rdptr	Pointer from the read pointer handler for keeping track of where in memory the FIFO should retrieve its next read

Block Diagram



References/Citations

[1] Xilinx Asynchronous FIFO V3.0 design specification