High Level Design Specification (HLDS)

For

Asynchronous FIFO Design Specification

Version 1.1

Prepared By: Amith Reddy Mukku
Bhadrinath Alludurgam
Ramakrishna Gopavaru
Risheek Bharadwaj

ECE-593: Fundamentals Of Pre-Silicon Validation
Prof.Venkatesh Patil

Revision History

Name	Date	Change Description	Version
Risheek Bharadwaj	04/23/2024	Base Documentation	1.0
Megha Sai Amith Reddy Mukku	04/23/2024	FIFO Depth Calculations For Our DUT	1.1

1.Introduction

1.1 Purpose

This document provides a detailed specification of the Asynchronous First-In-First-Out (FIFO) memory module, Version 1.0. Its purpose is to offer a clear and thorough overview of the module's architecture, encompassing functional elements, interface criteria, and verification aspects. It aims to assist system architects, design engineers, and verification engineers in understanding, integrating, and utilising the FIFO within digital systems. Furthermore, it elucidates how the FIFO facilitates asynchronous data transfer and synchronisation across different clock domains, thereby improving the reliability and effectiveness of digital data management.

1.2 Document Conventions

In this High-Level Design Specification (HLDS) document, Revision 1.0, for the Asynchronous First-In-First-Out (FIFO) memory module, specific typographical and formatting styles have been implemented to enhance readability and understanding. The document outlines the following conventions:

Bold text is employed to emphasize significant words, headings, and key points throughout the document. It also highlights important features, component names, and interfaces. Underlining is reserved for interactive elements and hyperlinks in electronic versions of the document, providing immediate access to referenced resources or papers. Monospaced font is utilized for code snippets, programming language keywords, and command-line examples to differentiate them from regular text. "Quotation marks" are used when directly citing specifications, standards, or essential statements from referenced documents.

1.3 Intended Audience & Reading Suggestions

The primary audience for this High-Level Design Specification (HLDS) for the Asynchronous FIFO memory module consists of system architects, design and verification engineers, and technical project managers. This document is tailored to provide comprehensive technical insights for development and integration purposes. It aims to serve the needs of individuals engaged in deep technical analysis as well as those involved in complete project planning requirements.

1.4 Product Scope

The Asynchronous FIFO memory module serves the purpose of enhancing data flow within complex digital systems by providing buffering and seamless data synchronization across multiple clock domains. Its integration is crucial in larger systems to improve data integrity and reduce latency. This module finds application in various components such as CPUs, network chips, SoCs, and communication interfaces. Its robust performance and adaptability to different data rates contribute to the goal of increasing efficiency and reliability in high-speed digital environments. Consequently, it plays a vital role in cutting-edge digital architectures, significantly enhancing their overall effectiveness and scalability

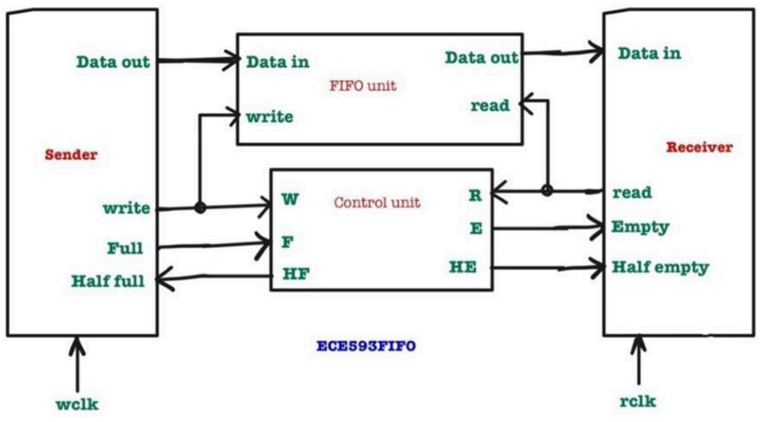
1.5 References

- Simulation and Synthesis Techniques for Asynchronous FIFO Design.
- Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons.
- Embedding Asynchronous FIFO Memory Blocks in Xilinx Virtex Series FPGAs Targeted for Critical Space System Applications.
- CALCULATION OF FIFO DEPTH -MADE EASY.
- FIFO Computing & Electronics
- Asynchronous FIFO Design with Gray code Pointer for High Speed AMBA AHB Compliant Memory controller.
- Crossing clock domains with an Asynchronous FIFO
- system on chip - Why do we use a gray encoded signal by 2 stage flip-flop in asynchronous FIFO to avoid race-condition issue? - Electrical Engineering Stack Exchange
- https://www.chipverify.com/verification/constraint-random-verification
- https://github.com/raysalemi/uvmprimer
- https://www.youtube.com/@theuvmprimer4123/playlists

2.Overall Description

2.1 Product Perspective

The asynchronous FIFO memory module stands out as a critical component tailor-made for integration into intricate digital systems such as System-on-Chips (SoCs), CPUs, and network chips. Its primary role lies in preserving data transmission synchronization among subsystems operating at similar frequencies but processing data at different rates, thus ensuring uninterrupted data flow and integrity. Developed to address the shortcomings of synchronous techniques, this module caters to the demands of systems necessitating high data throughput and minimal latency, thereby offering improved data transfer mechanisms in such contexts.



High
Level Block Diagram of the Design System

2.2 Product Functions

- Data Buffering: Temporarily stores data to balance fast producers and slower consumers.
- Clock Domain Synchronization: Ensures data integrity between components with different clock domains.
- Data Integrity Maintenance: Ensures data correctness throughout transmission.
- Flow Control Management: Regulates data flow to prevent overflow or underflow.
- Status Indication: Provides system operation signals such as 'FIFO full' or 'FIFO empty.'
- Flexible Data Handling: Adapts to varying data sizes and formats.
- Interface Compatibility: Easily integrates with SoCs, CPUs, and network chips through common interfaces.

2.3 User Classes and Characteristics

The Asynchronous FIFO memory module caters to various user classes, including digital system architects, hardware designers, firmware/software developers, system integrators, embedded system designers, research teams, academic institutions, and project managers/operators. With capabilities such as clock synchronization, flexible data handling, binary-to-gray code translation, and interface compatibility, it addresses specific requirements within digital system architecture. These features provide effective solutions for professionals and researchers encountering a range of design and integration challenges.

2.4 Tools And Software

The Asynchronous FIFO memory module interfaces with various hardware and software components within digital system environments. It operates within a context that includes
:

Within the hardware platform, the module serves as a critical data manager, functioning within digital systems such as CPUs, network chips, and System-on-Chips (SoCs).

Operating across a spectrum of scenarios, the FIFO module remains platform-independent, not reliant on any specific operating system.

For integration into hardware designs, users utilize Electronic Design Automation (EDA) software and digital design tools. Common tools like Synopsys Design Compiler, Cadence Virtuoso, and Xilinx Vivado are often employed.

Regarding configuration, firmware, and software components integrated with the FIFO module, the selection is based on the specific needs of the digital system. It seamlessly integrates with software and firmware intended for the respective system.

2.5 Design and Implementation Constraints

Here are the design and implementation restrictions for an asynchronous FIFO memory module based on the provided specifications:
Clock Frequency Compatibility: The module is designed to operate with a 50% duty cycle on both the producer clock (clk1) and consumer clock (clk2) frequencies, which may have different frequencies.
Data Width Compatibility: The data width of the module can be adjusted but must align with the specifications of the application to prevent data truncation or overflow. Consider the maximum write burst size of 200 stated in the design specifications.
Clock Domain Synchronization: While the module assists with clock domain synchronization, ensure that your designs incorporate proper clock domain crossings to avoid data synchronization issues in accordance with the design parameters.
Data Handling Limitations: Be mindful of data rates and flow within the specified parameters. Failure to manage high data rates adequately may lead to FIFO overflow or data loss. Consider factors such as the maximum write burst size and optimal cycles between consecutive writes and reads for effective control mechanisms.
Interface Compatibility: Verify that the interfaces in your system meet the module's requirements for data widths and clock frequencies specified in the design specifications.
Binary Code to Gray Code Conversion: If your application requires binary to gray code conversion, configure the module accordingly. Ensure that this feature aligns with the specific needs of your application before implementation.

2.6 Assumptions and Deadlines

2.6.1 Assumptions

Clock Signals: Users are required to provide stable clock signals (clk1 and clk2) at specified frequencies and duty cycles as per the design criteria.
Configuration: Users must specify the data width, clock frequencies, and operation modes of the FIFO module to meet the design requirements.

Flow Control: To prevent data overflow and underflow, users need to implement suitable flow control mechanisms.
Binary-to-Gray Code Conversion: If enabled, users assume alignment with their application needs and undertake any required configurations.

2.6.2 Dependencies
Clock Sources: Users must ensure the use of suitable clock sources that meet the module's requirements for proper functionality.
Tools: Users may find the need for digital design tools and verification tools such as Questa sim and EDA software to aid in integration and testing.
Hardware: Users may require FPGAs to facilitate the integration and verification of the design.
Clock Domain Crossing: Adequate clock domain crossing methods are necessary to synchronize data effectively.

3.Dataflow and Working

3.1 Dataflow and Clock Edge Triggering

Write Operation: Data entries into the FIFO occur on the positive edge of the producer clock (clk1). This ensures consistent latching of data at the peak of the clock cycle, offering stability and predictability, particularly in high-speed scenarios.
Read Operation: Data is read from the FIFO on the positive edge of the consumer clock (clk2). This synchronization with the rising edge of the clock minimizes read latency and ensures data integrity.

3.2 Dataflow and Clock Edge Triggering

Clock Domains: The FIFO operates at 500 MHz on two separate clock domains, with clk1 dedicated to writing and clk2 to reading.
Synchronizers: Due to the asynchronous nature of the FIFO, synchronizers are essential for secure transfer of read and write pointers between the CLK1 and CLK2 domains. This mitigates issues associated with moving signals between asynchronous clock domains, such as metastability.

3.3 Status Indicators

FIFO Empty: This signal is asserted when the read pointer matches the write pointer, signaling the absence of data in the FIFO.
FIFO Full: This signal is asserted when the write pointer is on the verge of overlapping the read pointer, indicating no available space remaining.
Almost-Empty Warning: An Almost Empty flag is asserted when the FIFO approaches a predetermined threshold, signaling that it is nearing empty status.

Almost-Full Warning: Similar to Almost Empty, an Almost Full flag is asserted when the FIFO approaches a predetermined threshold before reaching full capacity, indicating that it is nearing its maximum storage capacity.

3.4 Reset Functionality

Reset Mechanism: The FIFO incorporates a reset input for initializing its state. When activated, this mechanism clears the FIFO content, resets the read/write pointers, and clears all associated status flags (Full, Empty, Almost Full, Almost Empty).
Asynchronous Reset: The reset function is designed to be asynchronous, ensuring an immediate response independent of clock cycles.
Got it! Here's the modified version:

3.5 Counter Design

The design employs a dual Binary-Gray counter system. Gray code is used for pointer comparisons and generating status flags to ensure stability across clock cycles with one-bit changes. Internally, the binary counter manages the read and write pointers.

3.6 Timing Diagram

Read Operation:

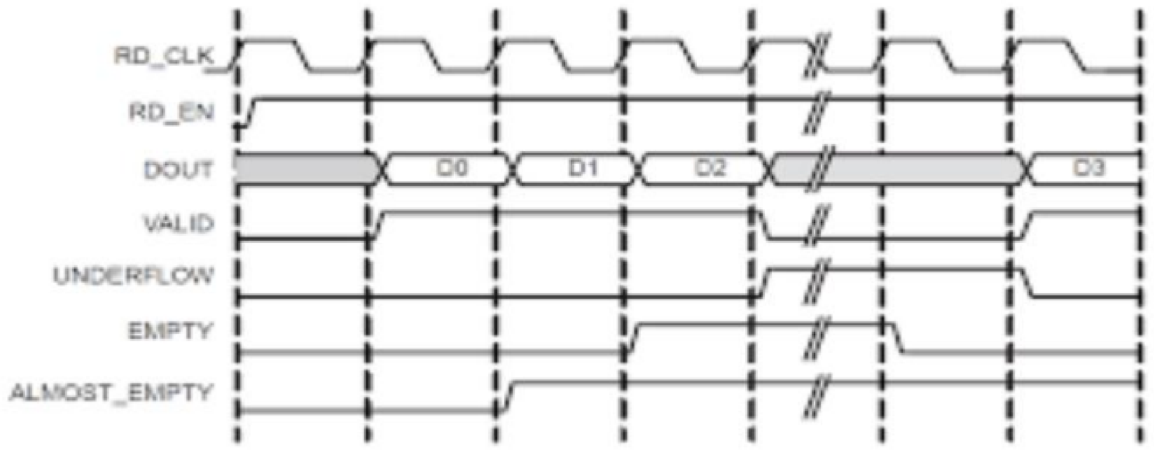


Fig 2 : Read Operation

Write Operation:

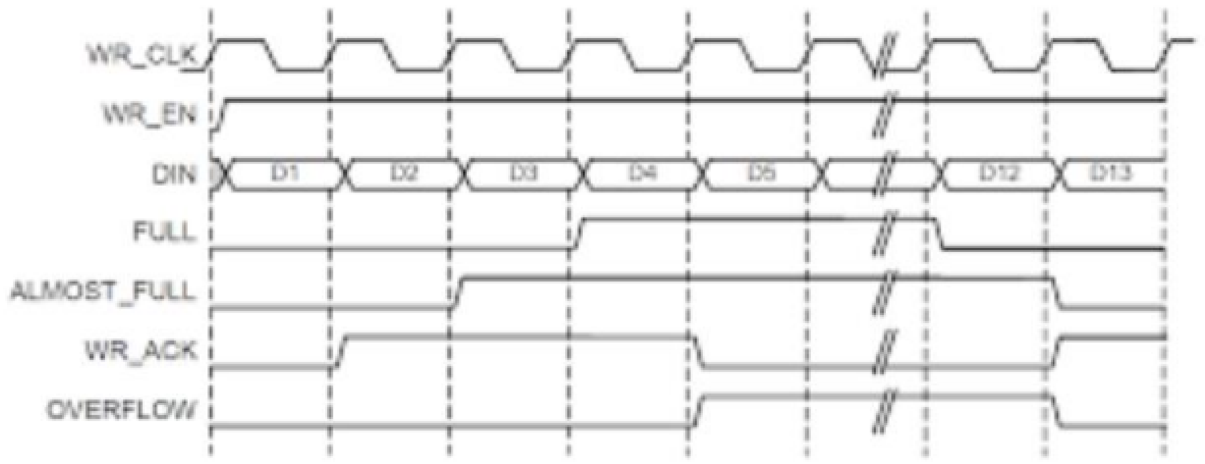


Fig 2 : Write Operation

4. External Interface Requirements

4.1 Hardware Interfaces

In the hardware interfaces section, we'll delve into the logical properties of the Asynchronous FIFO memory module and its interaction with interface modules. Here's a summary of the hardware interfaces and key features:

4.1.1 Data Input and Output

Data In (DI): External modules can input data into the FIFO through this interface. The data width arrangement must adhere to the specifications outlined in the design standards.
Data Out (DO): External modules can retrieve data from the FIFO through this interface. The data width must match the configuration initially established.

4.1.2 Clock Signals

clk1: This clock signal drives the FIFO's write operations, operating at a frequency of 1 GHz with a 50% duty cycle, as specified in the design.
clk2: This clock signal controls the readout process and operates at a frequency of 500 MHz with a 50% duty cycle.

4.1.3 Control Signals

Write Enable (WE): External modules utilize this signal to enable or disable write operations to the FIFO.
Read Enable (RE): This signal governs the reading of data from the FIFO.
Clear/Reset (CLR): Following design guidelines, the CLR signal is employed to reset the FIFO, clearing its contents and returning it to an empty state.

4.1.4 Status Signals

FIFO Full (FULL): This signal indicates that the FIFO is at maximum capacity, and no further write operations can be accommodated.
FIFO Empty (EMPTY): This signal indicates that the FIFO is devoid of data and cannot be read until new data is written.
FIFO Half Empty: This signal indicates that the FIFO is at half the capacity,but further read operations is allowed
FIFO Half Full: This signal indicates that the FIFO is at half the capacity, but further write operations is allowed

4.1.5 Binary-to-Gray Code Conversion

If the FIFO is set up for binary-to-gray code conversion, it can have extra logic for this function. This feature can be interfaced with by external modules as required

4.1.6 Synchronization Logic

In order to maintain data synchronization between the two clock domains (clk1 and clk2) in accordance with design criteria, the FIFO has internal logic for clock domain crossing.

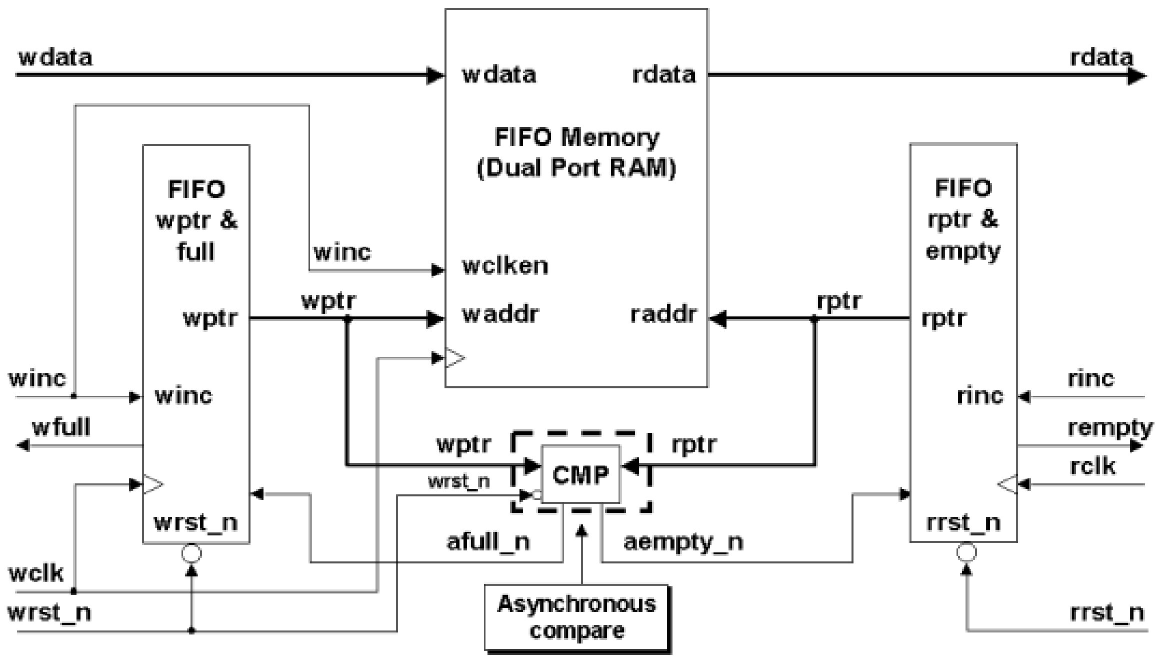


Fig 4 : Detailed Hardware Interface Blocks

4.2 Software Interfaces

4.2.1 Digital Design and Verification Tools

Clock domain crossover techniques, while not typically categorized as conventional software, are crucial for synchronizing data across multiple clock domains. Users can employ various techniques and tools to implement appropriate clock domain crossover approaches in their projects.

4.2.2 Clock Domain Crossing Techniques
Clock domain crossover techniques, although not software in the conventional sense, are essential to the synchronization of data between several clock domains. Users can use techniques and tools to apply suitable clock domain crossover approaches in their projects.

5. Product Features

5.1 FIFO Memory

At the core of the layout, this component serves as the primary data conduit. It facilitates the transfer of data written by the producer clock (clk1 at 1 GHz) to the consumer clock (clk2 at 500 MHz) by storing it. To support this functionality, the FIFO depth of the memory has been configured to handle a maximum burst size of 200 writes. Despite the asynchronous operation of the producer and consumer clocks, the FIFO maintains data integrity seamlessly.

5.2 Synchronizers

Critical for maintaining data integrity across multiple clock domains, these components synchronize the read and write pointers between the producer and consumer clocks. Typically, a multi-stage flip-flop chain is employed in the architecture to ensure precise pointer updates, even in the presence of varying duty cycles and clock skew.

5.3 Write and Read control Blocks

These blocks dictate the timing of write and read operations. According to the design requirements, there must be a minimum of two idle cycles between consecutive writes and four idle cycles between consecutive reads. This is crucial to prevent data collisions and ensure that the FIFO operates without encountering overrun or underrun errors.

5.4 Binary and Gray counter
Binary counters are employed in the design to manage read and write pointers. Gray coding is utilized to facilitate secure transfer of these pointers between separate clock domains, operating at 500 MHz with a 50% duty cycle for both the producer and the consumer clocks.

This coding ensures that only one bit changes at a time during pointer updates, significantly reducing the risk of metastability during clock domain switching.

In this design, the Gray code value (ptr, either wptr or rptr) represents the outputs of the register bits via the style #1 Gray code counter. These Gray code outputs are then fed into a Gray-to-binary converter (bin), which subsequently provides the next-binary-count-value (bnext) to a conditional binary-value incrementer. Finally, the binary-to-Gray converter generates the next-Gray-count-value (gnext), which is fed back to the register inputs.

The logic flow for this process is illustrated in the top half of the Figure block diagram. The logic related to the second Gray code counter, which will be further discussed in the subsequent section, is depicted in the bottom half.

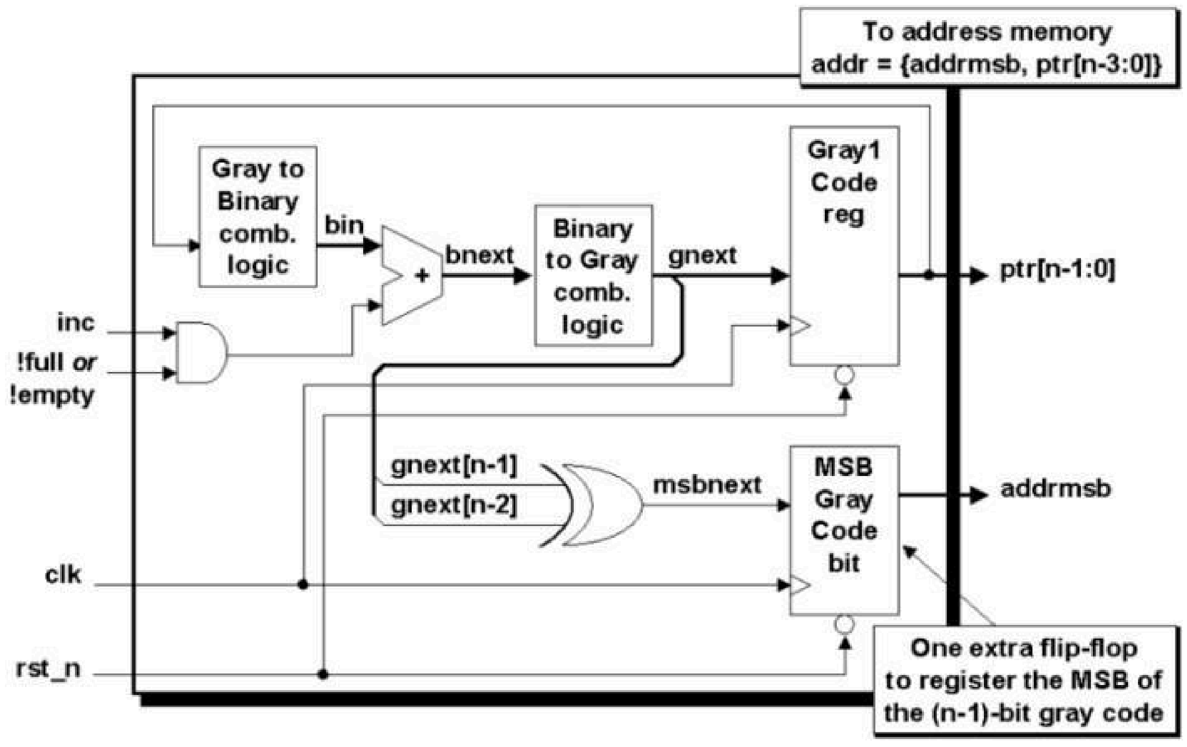

To address memory
addr = {addrmsb, ptr[n-3:0]}
Gray to Binary comb. logic
bin
bnext
Binary to Gray comb. logic
gnext
Gray1 Code reg
ptr[n-1:0]
inc
!full or !empty
gnext[n-1]
gnext[n-2]
msbnext
MSB Gray Code bit
addrmsb
clk
rst_n
One extra flip-flop to register the MSB of the (n-1)-bit gray code

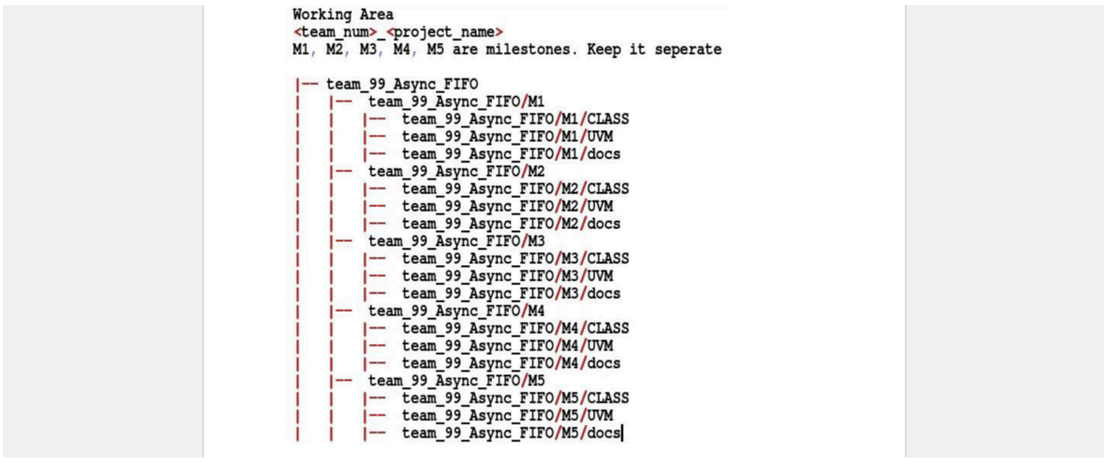
Fig 5 : Dual n-bit Gray Code Counter Block Diagram.

5.5 Interface Logic

This comprises the functionality required to interface the FIFO with other system elements. It guarantees compatibility and smooth data flow, taking into account the high frequency and particular timing needs.

6. Logic Design

6.1 Working Directory Structure



```
Working Area
<team_num>_<project_name>
M1, M2, M3, M4, M5 are milestones. Keep it seperate

|-- team_99_Async_FIFO
|   |--  team_99_Async_FIFO/M1
|   |    |--  team_99_Async_FIFO/M1/CLASS
|   |    |--  team_99_Async_FIFO/M1/UVM
|   |    |--  team_99_Async_FIFO/M1/docs
|   |--  team_99_Async_FIFO/M2
|   |    |--  team_99_Async_FIFO/M2/CLASS
|   |    |--  team_99_Async_FIFO/M2/UVM
|   |    |--  team_99_Async_FIFO/M2/docs
|   |--  team_99_Async_FIFO/M3
|   |    |--  team_99_Async_FIFO/M3/CLASS
|   |    |--  team_99_Async_FIFO/M3/UVM
|   |    |--  team_99_Async_FIFO/M3/docs
|   |--  team_99_Async_FIFO/M4
|   |    |--  team_99_Async_FIFO/M4/CLASS
|   |    |--  team_99_Async_FIFO/M4/UVM
|   |    |--  team_99_Async_FIFO/M4/docs
|   |--  team_99_Async_FIFO/M5
|   |    |--  team_99_Async_FIFO/M5/CLASS
|   |    |--  team_99_Async_FIFO/M5/UVM
|   |    |--  team_99_Async_FIFO/M5/docs
```

6.2 Design Modules

6.3 System Verilog Abstraction Features Used

6.4 Simulation,Tools

7. Verification

7.1 Testbench Style

7.2 Testing Strategies

7.3 Test case scenarios

7.4 Others

Summary

Appendix A: Glossary

FIFO Depth Calculation:

As per the group specifications specified our group is "J"
Sender Clock Frequency / Writing Frequency = 250Mhz.
Receiver Clock Frequency / Reading Frequency = 100Mhz.
Write Idle Cycles = 1.
Read Idle Cycles = 3.

Write Cycles:

The number of idle cycles between 2 successive writes is 1 which means after writing one data the next one is initiated after waiting for 1 clock cycle.
For every 2 clock cycles 1 data is written.
Time required to write one data item = 2 * 1/250Mhz = 2 * 4 nano sec.

Time required to write one data item = 8 nano seconds.

Read Cycles:

The number of idle cycles between 2 successive reads is 3 which means after reading one data the next one is initiated after waiting for 3 clock cycles.

For every 4 clock cycles 1 data is read.

Time required to read one data item = 4 * 1/100Mhz = 4 * 10 nano sec
Time required to read one data item = 40 nano seconds.
As per the given data the total burst of data is = 200.
So at a time the total number of writes that can be done is = 200.

Time taken to write all the 200 burst of data = 200 * 8 = 1600 nano seconds.

Total number of reads that can be done in 1600 nano seconds = 1600/40 = 40 reads.

Remaining data items to be read = 200 – 40 = 160.
The minimum depth of FIFO is = 160.
The depth of FIFO specified for the minimum requirements (i.e., 160) in this project is 256.