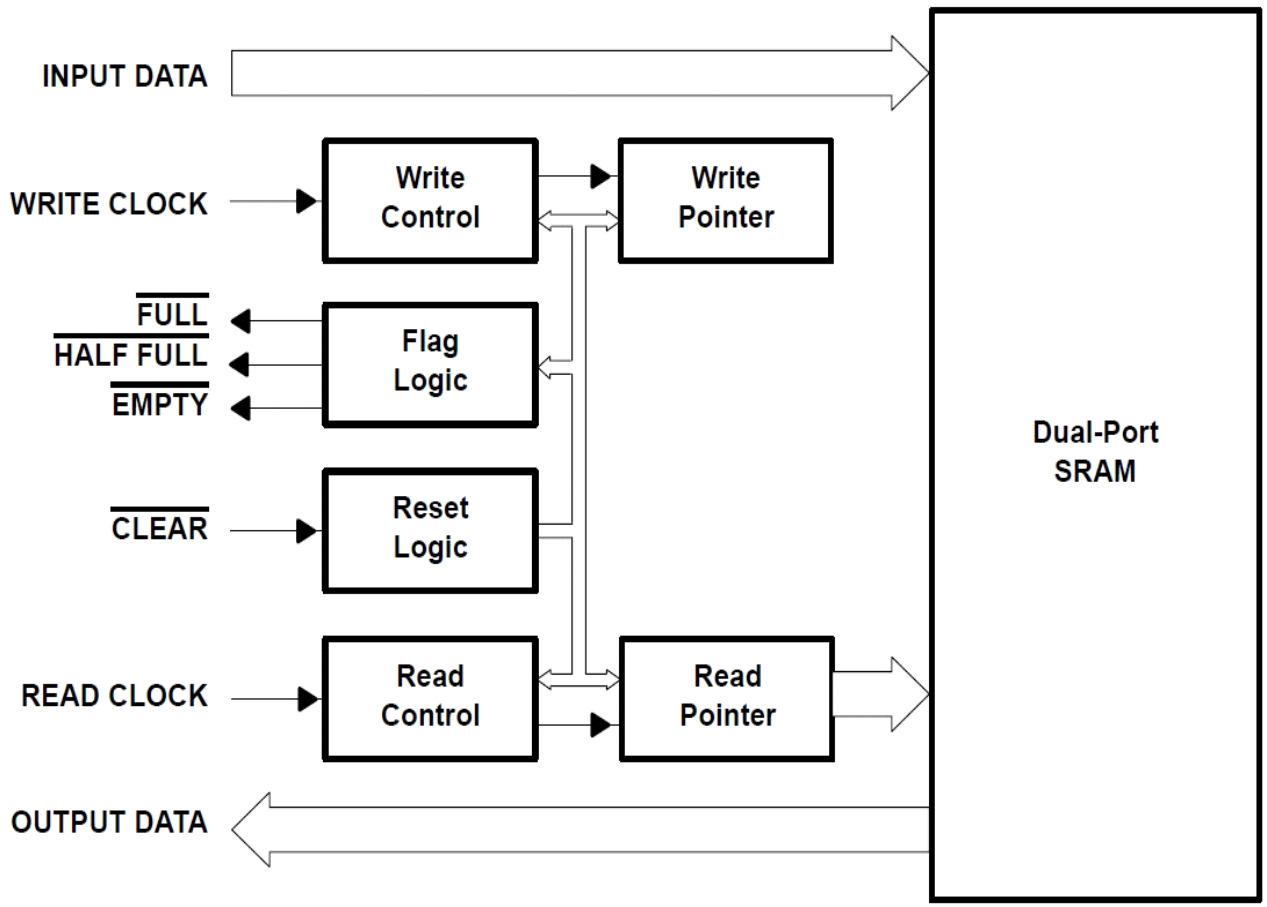
Asynchronous FIFO

Introduction :

The Asynchronous FIFO is a First-In-First-Out memory queue with control logic that performs management of the read and write pointers, generation of status flags, and optional handshake signals for interfacing with the user logic.



Block Diagram of FIFO With Static Memory

PIN DESCRIPTION :

Name	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)
clear	Input	Asynchronous reset of all FIFO functions, flags, and pointers
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: Handshake 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. This is an indication that a write operation was attempted, but the FIFO was Full.
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.

Name	Direction	Description
RD_ COUNT [R:0]	Output	Read Count: Count vector (unsigned binary) representing the number of data word 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: Handshake 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. This is an indication that a read operation was attempted, but the FIFO was Empty.

Functional Description

The Asynchronous FIFO is a First-In-First-Out memory queue with control logic that performs management of the read and write pointers, generation of status flags, and optional handshake signals for interfacing with the user logic. The individual read and write ports are fully synchronous (all operations qualified by a rising clock edge), but this FIFO does not require the read and write clocks to be synchronized to each other.

FIFO status cannot be corrupted by invalid requests. Requesting a read operation while the EMPTY flag is active will not cause any change in the current sate of the FIFO. Similarly, requesting a write operation while the FULL flag is active will not cause any change in the current state of the FIFO. If enabled, the RD_ERR and WR_ERR handshake signals will indicate the rejection of these invalid requests.

In addition to the EMPTY, ALMOST_EMPTY, FULL, and ALMOST_FULL flags, a count vector can be enabled to provide a more granular measure of the FIFO state. For the write domain the vector is WR_COUNT[W:0], and for the read domain it is RD_COUNT[R:0]. The width of these vectors are user programmable to provide easy generation of additional flags. For instance, a vector width of one creates a half-full flag; a width of two creates binary-encoded quadrant flags, and so on. In keeping with the fully synchronous interface, the count vector can be synchronized to either the read or the write clock domain, or two independent counts can be enabled, one for each clock domain.

Synchronization and Timing Issues

As previously stated, the read and write ports can be operated on independent asynchronous clock domains. However, the user interface logic still must address synchronization issues. The core schematic symbol, divides the signals according to their appropriate clock domains—write on the top half, read on the bottom. All signals, either input or output, are synchronous to one of the two clocks, with the exception of AINIT, which performs an asynchronous reset of the entire FIFO. On the write side, the control (WR_EN) and data input (DIN) are sampled by the rising edge of WR_CLK and should be synchronous to the WR_CLK. For the read side the read control (RD_EN) should be synchronous to the RD_CLK and the output data (DOUT) is valid after the subsequent rising edge of RD_CLK. All status outputs are synchronous to their respective clock domains and should be sampled only by logic operating on a synchronous clock. FIFO performance can be effectively constrained and analyzed by placing the desired clock PERIOD constraints on both the WR_CLK and RD_CLK source signals. WR_CLK and RD_CLK are always rising edge active for the FIFO core. They can be made falling edge active (relative to the clock source) by inserting an inverter between the clock source and the FIFO’s clock inputs.

Behavior of Status Signals

The activation of the AINIT, asynchronous initialization (reset), will force all four FIFO flags to the active (high) state. On the first WR_CLK after the release of AINIT the FULL and ALMOST_FULL flags will become inactive, indicating that the FIFO is now ready to accept write operations. EMPTY and ALMOST_EMPTY are deactivated on a rising edge of the RD_CLK following the first and second writes respectively. The ALMOST_EMPTY flag is active when the FIFO has one data word or is EMPTY. The ALMOST_FULL flag is active when the FIFO has only one available memory location or is FULL.
Optional handshake signals are provided to simplify user control logic interacting with the FIFO. The WR_ACK and WR_ERR signals indicate acknowledgment or rejection of requested write operations. Similarly, RD_ACK and RD_ERR signals indicate the acknowledgment or rejection of read operations. Each of these control signals can be made active high or low from the GUI. Note that all of these handshake signals are synchronous to their respective clock domains and indicate the acknowledgment or rejection of requests during the prior rising clock edge. Because an acknowledgment or error response depends on an active request (WR_EN or RD_EN), the ACK and ERR signals are not always the inverse of each other. If no operation is requested, both the acknowledgment and the error signal will be inactive during the subsequent clock period. For an example of expected signal sequencing.

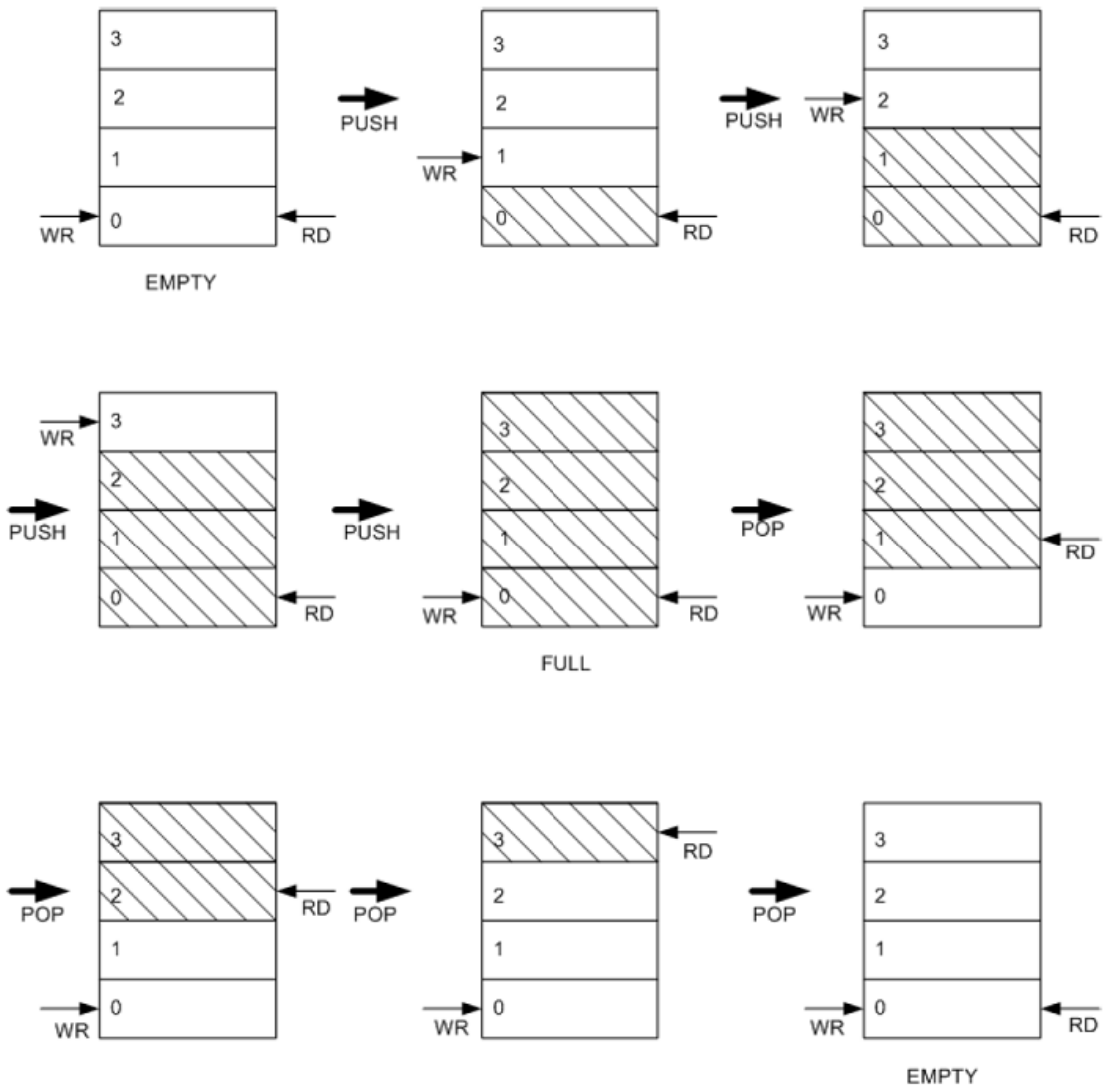
The optional data count outputs (WR_COUNT and RD_COUNT) support the generation of user programmable flags. In the simplest case, selecting a width of one for a data count produces a half-full flag. Like all other FIFO outputs, the counts are synchronized to their respective clock domains and should be sampled only by logic operating on the same (or a synchronous) clock. The data count vectors have clock latency and should not be used as substitutes for the FULL, ALMOST_FULL, EMPTY, or ALMOST _EMPTY flags. The clock latency of the counts in their respective clock domains is one cycle. For example, the WR_COUNT does not reflect the impact of a write operation performed as a result of a request (WR_EN active) during the prior clock cycle. WR_COUNT and RD_COUNT values are not guaranteed to produce a precise representation of the FIFO contents at a particular point in time. These values should be used as a gauge to determine the FIFO status.
The latency for operations in the opposing clock domain can be up to three clock cycles. For example, in the case of the WR_COUNT, read operations that may have been performed during the immediate three prior RD_CLK periods will not be reflected in the data count vector. This latency results from a design trade-off between clock frequency and count accuracy and is not as limiting as it may at first appear.

Consider the following scenario of a FIFO configured depth of 63 and a write count of two bits (WR_COUNT[1:0]).
Note that for this example:
Write_COUNT[1:0]=00: Indicates that the FIFO is less than 1/4 full and corresponds to the occupancy range of (0:16). The upper bound is 16 and not 15 due to the write latency of 1 clock cycle.

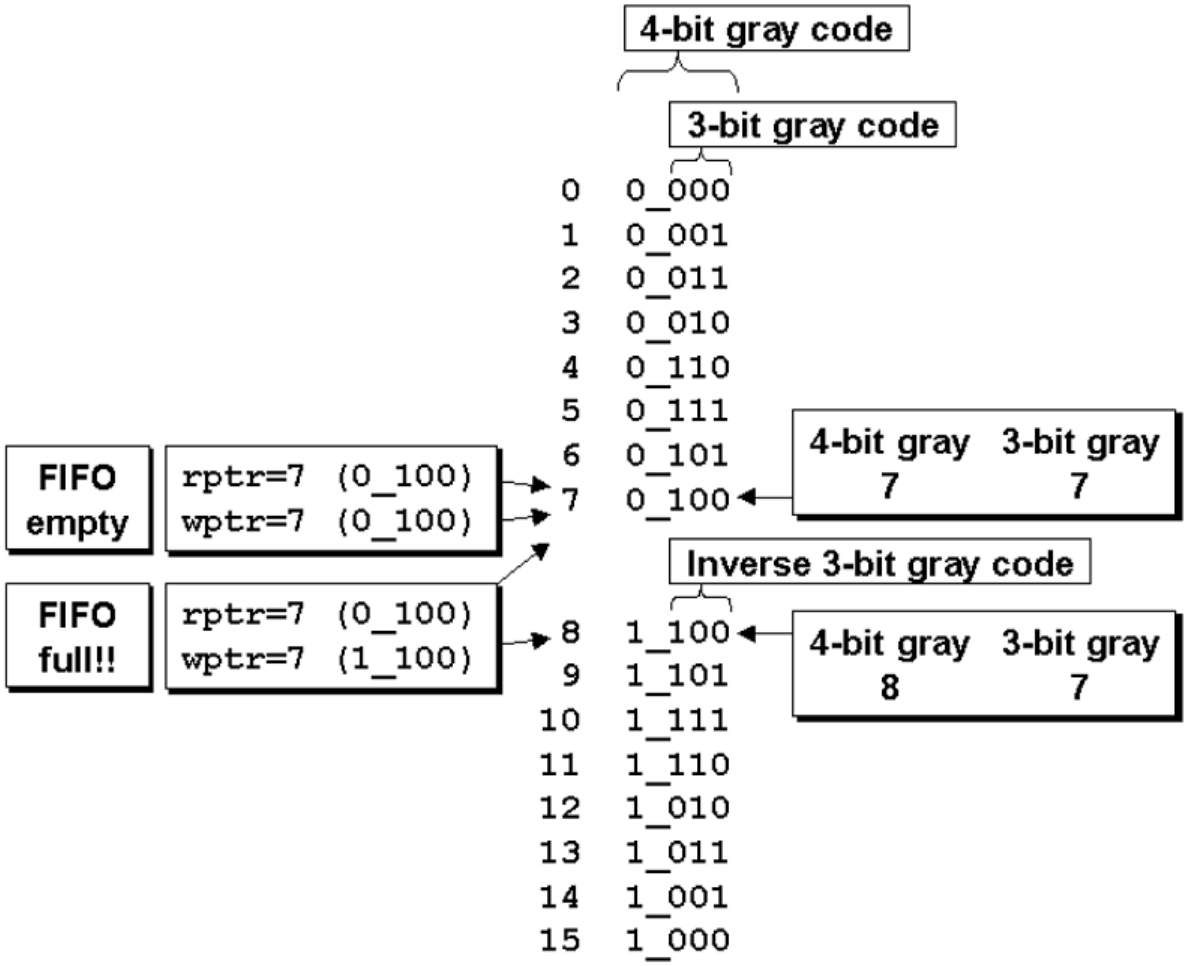
Write_COUNT[1:0]=01: Indicates that the FIFO is between 1/4 full and 1/2 full and corresponds to the occupancy range of (13:32). The lower bound is 13 and not 16 due to the read latency of 3 clock cycles.
Write_COUNT[1:0]=10: Indicates that the FIFO is between 1/2 full and 3/4 full and corresponds to the occupancy range of (29-48).
Write_COUNT[1:0]=11: Indicates that the FIFO is between 3/4 full and full and corresponds to the occupancy range of (45-63).

If the control logic needs to throttle back write operations based on the FIFO occupancy, it can use the write count vector in the following way. As shown above, WR_COUNT[1:0] equal to 11 corresponds to an occupancy greater than 45. As long as the user's WR_COUNT is not 11, no more than 48 data words (47 plus one for the write operation clock latency) are present in the FIFO. The user's control logic is assured that at least 15 (63-48) additional memory locations are available in the queue. There could be up to three more locations because of recent read operations, but this only increases the available memory locations. In this scenario, at least 14 additional writes can be performed without causing the FULL flag to transition to true.

FIFO Full and Emty description:



Gray code counter:


4-bit gray code
3-bit gray code
0 0_000
1 0_001
2 0_011
3 0_010
4 0_110
5 0_111
6 0_101
7 0_100
8 1_100
9 1_101
10 1_111
11 1_110
12 1_010
13 1_011
14 1_001
15 1_000
FIFO empty
rptr=7 (0_100)
wptr=7 (0_100)
FIFO full!!
rptr=7 (0_100)
wptr=7 (1_100)
4-bit gray 3-bit gray
7 7
Inverse 3-bit gray code
4-bit gray 3-bit gray
8 7