Intel® Stratix® 10 Embedded Memory User Guide

Updated for Intel® Quartus® Prime Design Suite: **20.1**

Subscribe
Send Feedback

UG-S10MEMORY | 2020.08.03
Latest document on the web: PDF | HTML

Contents

1. Intel® Stratix® 10 Embedded Memory Overview

Intel® Stratix® 10 embedded memory blocks are flexible and provide an optimal amount of various sized memory arrays to fit your design requirements.

Related Information

HyperFlex Core Architecture, Intel Stratix 10 Device Overview
Provides more information about Hyper-Registers and the HyperFlex core architecture. Hyper-Registers are additional registers available in every interconnect routing segment throughout the core fabric, including the routing segments connected to the memory logic array block (MLAB) and M20K block inputs and outputs.

1.1. Intel Stratix 10 Embedded Memory Features

The Intel Stratix 10 devices contain three types of memory blocks: Embedded SRAM (eSRAM) blocks, M20K blocks, and memory logic array blocks (MLABs).

- 47.25-Megabit (Mb) eSRAM blocks
  - Fast path, low latency, high bandwidth and very high random transaction rate (RTR) on-chip memory block
  - Each block consists of 8 channels and each channel has 42 banks.
  - Each bank is configurable to 2K depth and 72-bit data width
  - Supports only simple dual-port RAM with concurrent read and write access per channel
- 20-kilobit (Kb) M20K blocks
  - Blocks of dedicated memory resources.
  - Ideal for larger memory arrays, while providing a large number of independent ports.
- 640-bit MLABs
  - Enhanced memory blocks configured from dual-purpose logic array blocks (LABs).
  - Ideal for wide and shallow memory arrays.
  - Optimized for implementation of shift registers for digital signal processing (DSP) applications, wide and shallow FIFO buffers, and filter delay lines.
  - Each MLAB is made up of ten adaptive logic modules (ALMs).

In Intel Stratix 10 devices, you can configure each ALM in the MLAB as ten 32×2 blocks. The Intel Stratix 10 devices provide one 32×20 simple dual-port SRAM block per MLAB.

The Intel Stratix 10 embedded memory blocks support the following operation modes:

- Single-port
- Simple dual-port
- True dual-port
- Simple quad-port
- ROM

Table 1. Intel Stratix 10 Embedded Memory Features

This table summarizes the features supported by the Intel Stratix 10 embedded memory blocks.

Features	eSRAM	M20K	MLAB
Maximum operating frequency	750 MHz	• 1 GHz (simple dual-port RAM mode) • 600 MHz (true dual-port and simple quad-port RAM mode)	1 GHz
Total RAM bits (including parity bits)	47.25 Mb	20,480 bits	640 bits
Byte enable	N/A	Supported	Supported
Address clock enable	N/A	Supported (only in simple dual-port RAM mode)	Supported
Simple dual-port mixed width	N/A	Supported	N/A
FIFO buffer mixed width	N/A	Supported	N/A
Memory Initialization File (`.mif`)	N/A	Supported	Supported
Dual-clock mode	N/A	Supported (only in simple dual-port RAM mode)	Supported
Full synchronous memory	N/A	Supported	Supported
Asynchronous memory	N/A	N/A	Only for flow-through read memory operations
Power-up state	N/A	Output ports are cleared	• Registered output ports are cleared • Unregistered output ports read memory contents
Asynchronous/Synchronous Clears	N/A	Output registers and output latches	Output registers and output latches
Write/read operation triggering	Rising clock edges	Rising clock edges	Rising clock edges
Same-port read-during-write	N/A	Output ports set to *New Data* or *Don't Care*	Output ports set to *Don't Care*
Mixed-port read-during-write	Write-forwarding feature • ON = *New Data* • OFF = *Old Data*	• Simple Dual Port RAM: Output ports set to *Old Data* or *Don't Care* • True Dual Port RAM: Output ports set to *Don't Care* • Simple Quad Port: Output ports set to *new_a_old_b*	Output ports set to *New Data*, *Old Data*, or *Don't Care*

continued...

Features	eSRAM	M20K	MLAB
Error Correction Code (ECC) support	• Soft IP using the Intel Quartus® Prime software • Built-in support ×64-wide simple dual-port mode	• Soft IP using the Intel Quartus Prime software • Hard IP • Built-in support ×32-wide simple dual-port mode • Parity bits	Soft IP using the Intel Quartus Prime software
Force-to-Zero	N/A	Supported	N/A
Coherent read memory	N/A	Supported	N/A
Freeze logic	N/A	Supported	N/A
True dual port (TDP) dual clock emulator	N/A	Supported	N/A

1.2. Intel Stratix 10 Embedded Memory Capacity

Table 2. Embedded Memory Capacity and Distribution in Intel Stratix 10 Devices

This table lists the embedded memory capacity for Intel Stratix 10 GX, Intel Stratix 10 MX, Intel Stratix 10 SX, and Intel Stratix 10 TX variants.

Variant	Product Line	eSRAM		M20K		MLAB		Total RAM Bit (Mbits)
		Block	RAM Bit (Mbits)	Block	RAM Bit (Mbits)	Block	RAM Bit (Mbits)	
Intel Stratix 10 GX	GX 400	N/A	N/A	1,537	30	3,276	2	32
	GX 650	N/A	N/A	2,489	49	5,364	3	52
	GX 850	N/A	N/A	3,477	68	7,124	4	72
	GX 1100	N/A	N/A	5,461	107	11,556	7	114
	GX 1650	N/A	N/A	5,851	114	13,764	8	122
	GX 1660	N/A	N/A	6,162	120	14,230	9	129
	GX 2100	N/A	N/A	6,501	127	17,316	11	138
	GX 2110	N/A	N/A	6,847	134	17,856	11	145
	GX 2500	N/A	N/A	9,963	195	20,529	13	208
	GX 2800	N/A	N/A	11,721	229	23,796	15	244
	GX 10M	N/A	N/A	12,950	253	87,984	55	308
Intel Stratix 10 MX	MX 1650	2	94.5	6,162	120	14,230	9	223.5
	MX 2100	2	94.5	6,847	134	17,856	11	239.5
Intel Stratix 10 SX	SX 400	N/A	N/A	1,537	30	3,276	2	32
	SX 650	N/A	N/A	2,489	49	5,364	3	52
	SX 850	N/A	N/A	3,477	68	7,124	4	72
	SX 1100	N/A	N/A	5,461	107	11,556	7	114
	SX 1650	N/A	N/A	5,851	114	13,764	8	122
	SX 2100	N/A	N/A	6,501	127	17,316	11	138
	SX 2500	N/A	N/A	9,963	195	20,529	13	208

continued...

Variant	Product Line	eSRAM		M20K		MLAB		Total RAM Bit (Mbits)
		Block	RAM Bit (Mbits)	Block	RAM Bit (Mbits)	Block	RAM Bit (Mbits)	
	SX 2800	N/A	N/A	11,721	229	23,796	15	244
Intel Stratix 10 TX	TX 400	N/A	N/A	1,537	30	3,276	2	32
	TX 850	N/A	N/A	3,477	68	7,124	4	72
	TX 1100	N/A	N/A	5,461	107	11,556	7	114
	TX 1650	2	94.5	6,162	120	14,230	9	223.5
	TX 2100	2	94.5	6,847	134	17,856	11	239.5
	TX 2500	N/A	N/A	9,963	195	20,529	13	208
	TX 2800	N/A	N/A	11,721	229	23,796	15	244
Intel Stratix 10 DX	DX 1100	N/A	N/A	5,461	107	11,556	7	114
	DX 2100	2	94.5	6,847	134	17,856	11	145
	DX 2800	N/A	N/A	11,721	229	23,796	15	244

2. Intel Stratix 10 Embedded Memory Architecture and Features

The Intel Stratix 10 embedded memory features include operation modes, clocking modes, and configurations.

2.1. Byte Enable in Intel Stratix 10 Embedded Memory Blocks

The Intel Stratix 10 embedded memory blocks support byte enable controls.

- The byte enable controls mask the input data so that only specific bytes of data are written. The unwritten bytes retain the values written previously.
- The write enable (`wren`) signal, together with the byte enable (`byteena`) signal, control the write operations on the embedded memory blocks. By default, the `byteena` signal is high (enabled) and only the `wren` signal controls the writing.
- The byte enable registers do not have a `clear` port.
- The LSB of the `byteena` signal corresponds to the LSB of the data bus.
- The byte enable signals are active high.

2.1.1. Byte Enable Controls

Table 3. Byte Enable Controls in ×10 Data Width (MLAB)

Byte Enable [1:0]	Data Bits Written	
11 (default)	[9:5]	[4:0]
10	[9:5]	N/A
01	N/A	[4:0]
00	N/A	N/A

Table 4. Byte Enable Controls in ×20 Data Width (M20K)

Byte Enable [1:0]	Data Bits Written	
11 (default)	[19:10]	[9:0]
10	[19:10]	N/A
01	N/A	[9:0]
00	N/A	N/A

Table 5. Byte Enable Controls in ×40 Data Width (M20K)

Byte Enable [3:0]	Data Bits Written			
1111 (default)	[39:0]	[29:0]	[19:10]	[9:0]
1110	[39:0]	[29:0]	[19:10]	N/A
1101	[39:0]	[29:0]	N/A	[9:0]
1100	[39:0]	[29:0]	N/A	N/A
1011	[39:0]	N/A	[19:10]	[9:0]
1010	[39:0]	N/A	[19:10]	N/A
1001	[39:0]	N/A	N/A	[9:0]
1000	[39:0]	N/A	N/A	N/A
0111	N/A	[29:0]	[19:10]	[9:0]
0110	N/A	[29:0]	[19:10]	N/A
0101	N/A	[29:0]	N/A	[9:0]
0100	N/A	[29:0]	N/A	N/A
0011	N/A	N/A	[19:10]	[9:0]
0010	N/A	N/A	[19:10]	N/A
0001	N/A	N/A	N/A	[9:0]
0000	N/A	N/A	N/A	N/A

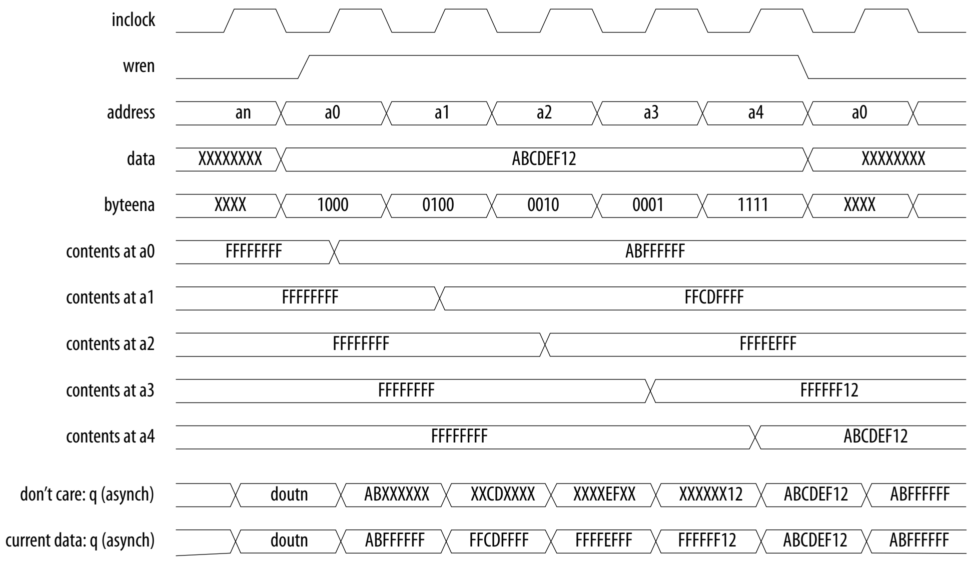
2.1.2. Data Byte Output

In M20K blocks or MLABs, when you de-assert a byte-enable bit to 0 during a write cycle, the corresponding data byte output appears as either a *Don't Care* value, or the current data at that location. You can control the output value for the masked byte in the M20K blocks or MLABs in the same-port read-during-write mode by using the Platform Designer in Intel Quartus Prime software.

2.1.3. Byte Enable Behavior

Figure 1. Byte Enable Functional Waveform

This figure shows how the `wren` and `byteena` signals control the operations of the embedded memory blocks.



2.2. Address Clock Enable Support

Intel Stratix 10 embedded memory blocks support address clock enable. When you enable address clock enable (`addressstall` = 1), it holds the previous address value.

Note: Only simple dual-port mode supports this feature.

When you configure the memory blocks in dual-port mode, each port has its own independent address clock enable.

Figure 2. Address Clock Enable

This figure shows an address clock enable block diagram.

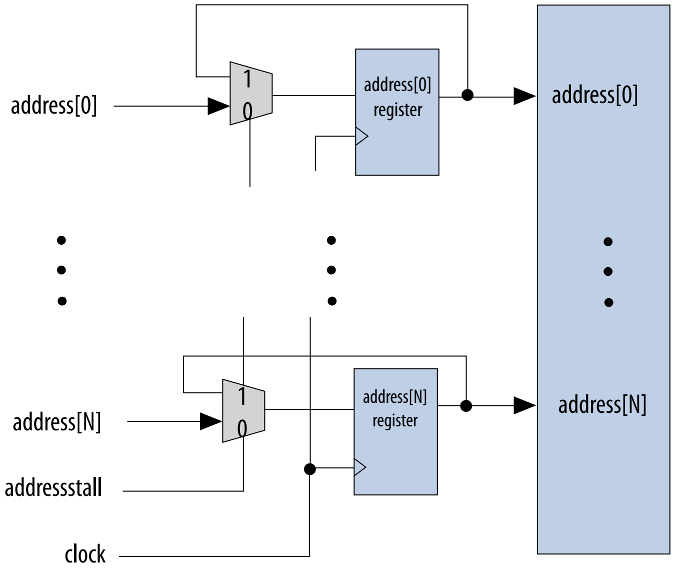


Figure 3. Address Clock Enable During Read Cycle

This figure shows the address clock enable behavior during read cycle.

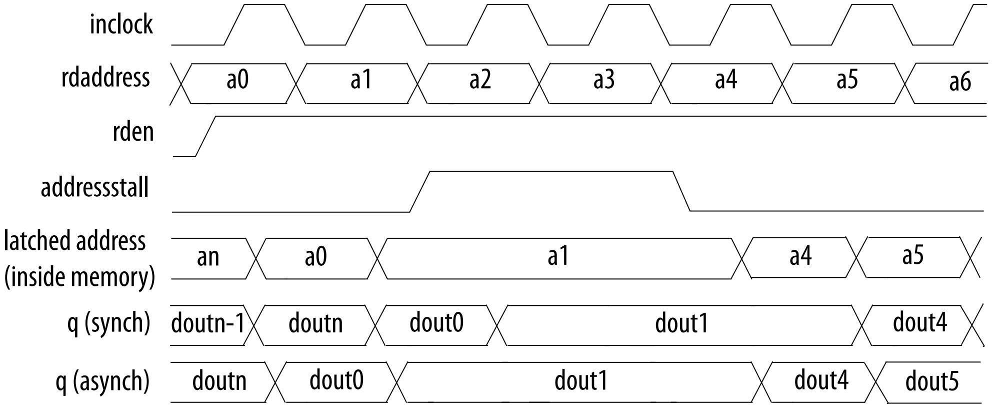
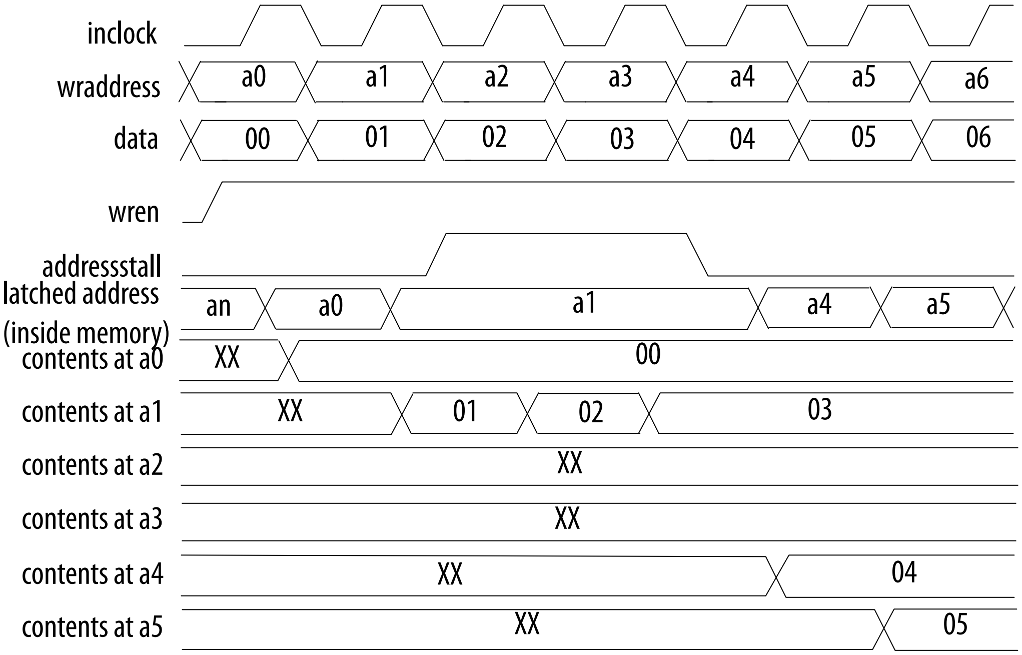


Figure 4. Address Clock Enable During Write Cycle

This figure shows the address clock enable behavior during write cycle.



2.3. Asynchronous Clear and Synchronous Clear

The Intel Stratix 10 M20K and MLAB embedded memory blocks support asynchronous clear and synchronous clear on output latches and output registers.

If your RAM does not use output registers, the RAM outputs are cleared using the latch asynchronous clear (aclr). The (aclr) signal is generated at any time. The internal logic extends the clear pulse until the next rising edge of the output clock. When the aclr signal asserts, the outputs are cleared and stay cleared until the next read cycle.

For the synchronous clear (sclr) signal, the RAM outputs are cleared at the next rising edge of the output clock when the (sclr) signal is asserted. The outputs will stay cleared until the next read cycle.

Note: Both aclr and sclr signals must be used separately for each RAM configuration.

Figure 5. Behavior of Asynchronous Clear and Synchronous Clear in Registered Mode

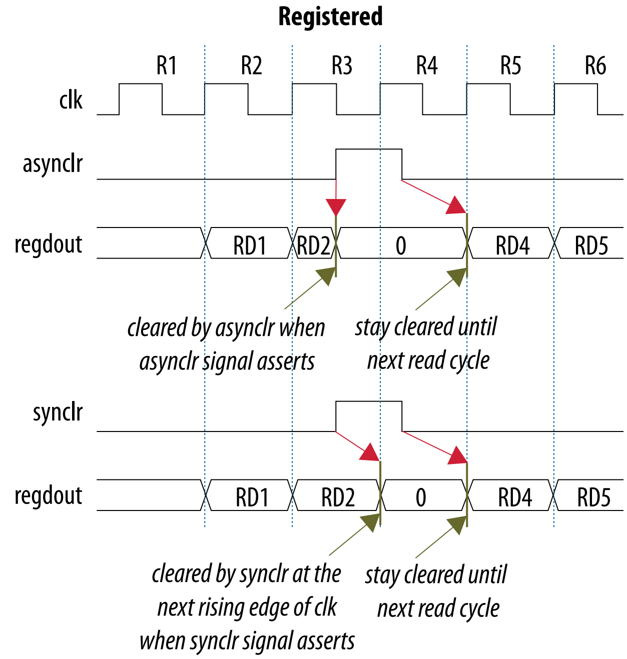
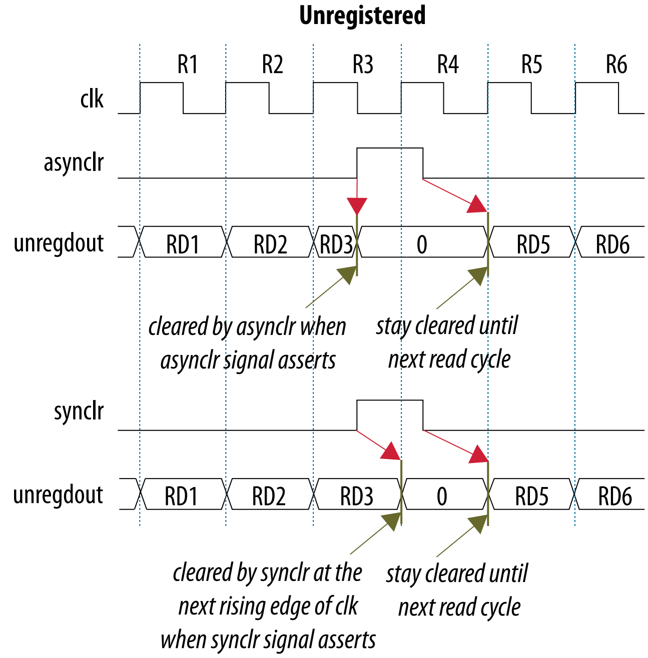

Registered
R1
R2
R3
R4
R5
R6
clk
asynclr
regdout
RD1
RD2
0
RD4
RD5
cleared by asynclr when asynclr signal asserts
stay cleared until next read cycle
synclr
regdout
RD1
RD2
0
RD4
RD5
cleared by synclr at the next rising edge of clk when synclr signal asserts
stay cleared until next read cycle

Figure 6. Behavior for Asynchronous Clear and Synchronous Clear in Unregistered Mode


Unregistered
R1
R2
R3
R4
R5
R6
clk
asynclr
unregdout
RD1
RD2
RD3
0
RD5
RD6
cleared by asynclr when asynclr signal asserts
stay cleared until next read cycle
synclr
unregdout
RD1
RD2
RD3
0
RD5
RD6
cleared by synclr at the next rising edge of clk when synclr signal asserts
stay cleared until next read cycle

2.4. Memory Blocks Error Correction Code Support

ECC detects and corrects data errors at the output of the memory.

Only M20K blocks and eSRAM blocks support the ECC feature.

If you engage the ECC feature, you cannot use the following features:

- Byte enable
- Coherent read

M20K Blocks

For M20K blocks, ECC performs single-error correction, double-adjacent-error correction, and triple-adjacent-error correction in a 32-bit word. However, ECC cannot guarantee detection or correction of non-adjacent two-bit or more errors.

The M20K blocks have built-in support for ECC when in ×32-wide simple dual-port mode.

- When you engage the ECC feature, the M20K runs slower than the non-ECC simple dual-port mode. However, you can enable optional ECC pipeline registers before the output decoder to achieve higher performance compared to non-pipeline ECC mode at the expense of one-cycle latency.
- Two ECC status flag signals—e (error) and ue (uncorrectable error) indicate the M20K ECC status. The status flags are part of the regular outputs from the memory block.

eSRAM Blocks

For eSRAM blocks, ECC performs single-error correction and double-error detection in a 64-bit word.

The eSRAM blocks have built-in support for ECC when in ×64-wide simple dual-port mode.

- Two ECC status flag signals—`c{7:0}_error_correct_0` (error corrected) and `c{7:0}_error_detect_0` (error detected) indicate the eSRAM ECC status.

2.4.1. Parity Bit

The following describes the parity bit support for M20K blocks:

- 8 parity bits are generated through the ECC encoder based on 32-bit input data width, resulting in up to a total of 40 bits of data width.
- You can inject and flip the parity bits by using the ECC parity flip feature.

2.4.2. ECC Parity Flip

The ECC parity flip feature dynamically flips the parity value generated in the encoder of M20K blocks to observe the ECC behavior through simulation.

When the ECC Encoder Bypass (`eccencbypass`) port is high, the built-in ECC encoder values are XOR-ed with the 8 parity bits through the parity ports to generate a new set of encoder value. When the ECC Encoder Bypass port is low, the encoder generates the parity bits according to the data input during a write process.

The following table shows an example to construct an 8-bit data width for the parity port.

Table 6. Example of Setting the 8-Bit Parity Ports

Parity Bit Sequence	ECC Feature	Is the ECC Decoder able to Recognize and Correct the Data Bit?
`00000001`	Single-error correction	Yes
`00000011`	Double-adjacent-error correction	Yes
`00000111`	Triple-adjacent-error correction	Yes
`00000101`	Triple-adjacent-error correction	Yes
`00010011`	Non-adjacent double/triple correction/detection	No guarantee

2.4.3. ECC Read-During-Write Behavior

For M20K blocks, you can select either *Old Data* or *Don't Care* output mode. By default, the mixed port read-during-write mode is set to *Don't Care*. When the mixed port read-during-write is set as *Don't Care*, both RAM data output and `eccstatus` will be 'X'. However, if the mixed port read-during-write mode is set as *Old Data*, the RAM data output will be the old data and the ECC status will be a deterministic value.

2.4.4. Error Correction Code Truth Table

Table 7. ECC Status Flags Truth Table for M20K

Eccstatus[1]e	Eccstatus[0]ue	Status
0	0	No error.
0	1	Illegal/Invalid.
1	0	A correctable error occurred and the error has been corrected at the outputs; however, the memory array has not been updated.
1	1	An uncorrectable error occurs and uncorrectable data appears at the outputs.

Figure 7. ECC Block Diagram for M20K Memory

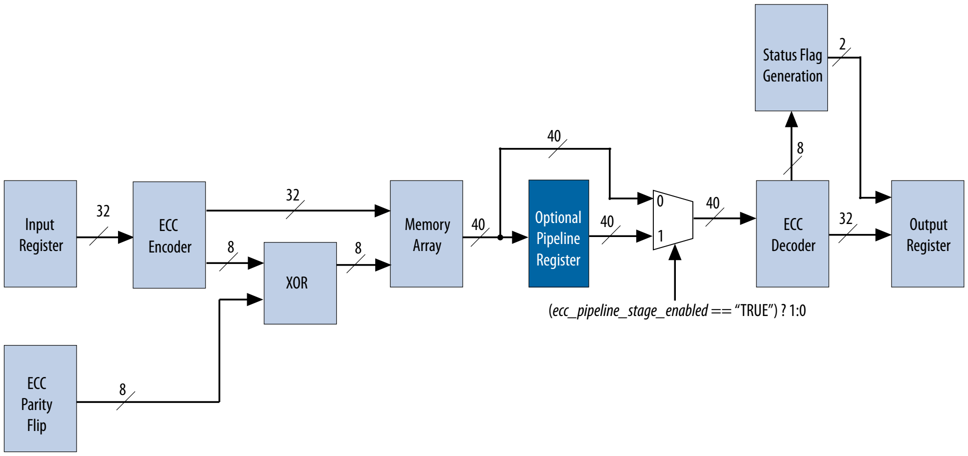


Table 8. ECC Status Flags Truth Table for eSRAM

C{7:0}_error_detect_0	C{7:0}_error_correct_0	Status
0	0	No error.
0	1	Illegal.
1	0	An error is detected but uncorrectable. The uncorrectable data appears at the outputs.
1	1	An error is detected and correctable. The error has been corrected at the outputs. The corrected data appears at the outputs but the memory array is not updated.

2.5. Force-to-Zero

The Force-to-Zero feature helps improve timing when a RAM memory block selected is larger than a single memory block. This feature is applicable only for M20K blocks.

For example, if the selected RAM memory block has a memory depth of 4096, the M20K block, which supports only a maximum memory depth of 2048, will require two RAMs to be multiplexed together. When you engage with this feature, you can replace OR gate with multiplexing circuitry at the output of the M20K block when performing address width stitching. As the MSB of address controls the read enable signal in the Force-to-Zero mode, the outputs of other memory blocks are forced to zero when the read enable signal is deasserted. This results the data output being read out from the output of the selected memory block only.

You have the option to turn on **Enable Force-to-Zero feature** in the parameter editors of the RAM/ROM IP cores.

Note: When you turn on **Enable Force-to-Zero feature**, the read enable signal does not retain previous values when you deassert the signal.

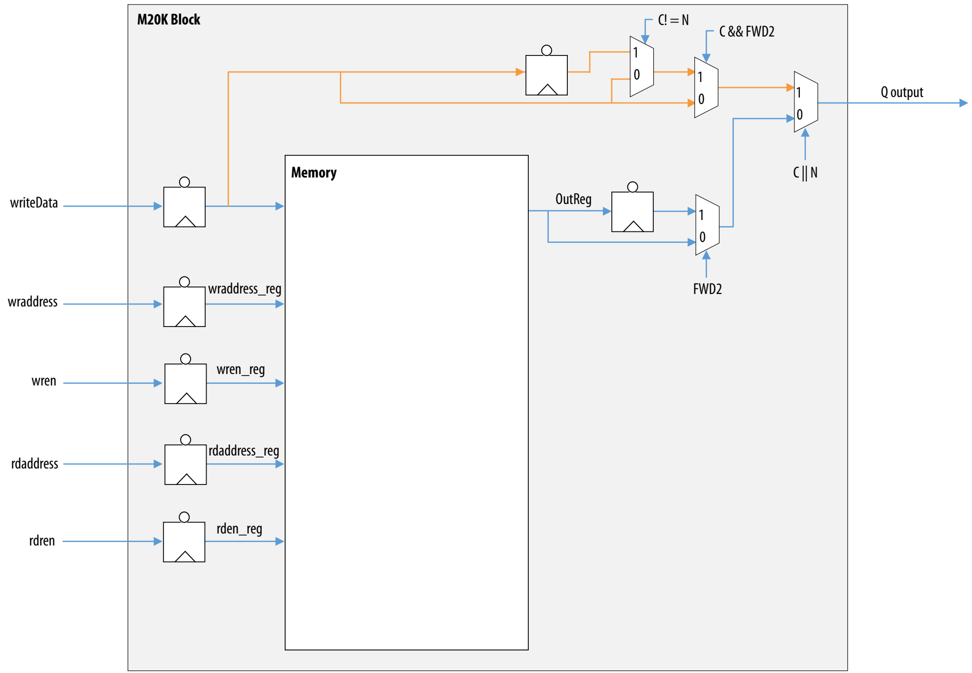
2.6. Coherent Read Memory

The coherent read memory feature allows you to read out the output data that will be written into the same memory content in a single clock cycle. In other words, you will experience the new data (flow through) behavior during the read-during-write operation. This feature is applicable only for M20K blocks and supported only in single clock configuration.

If you engage the coherent read memory feature, you cannot use the following configurations:

- Operating modes other than simple dual-port
- Simple dual-port with different port width
- Byte enable
- ECC
- Wide simple dual-port
- Dual clock configuration

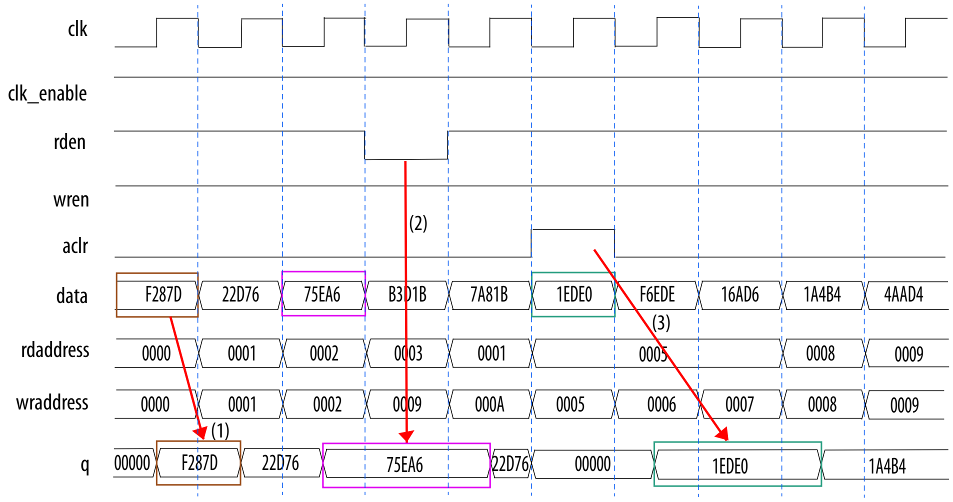
Figure 8. Simplified Block Diagram of Coherent Read Memory Circuitry



Notes:
1. N = (rden_reg && wren && (rdaddress_reg == wraddress)) ? 1 : 0.
2. C = (rden_reg && wren_reg && (rdaddress_reg == wraddress_reg)) ? 1 : 0.
3. FWD2 = (forwarding_stage2) ? 1 : 0.

Figure 9. Coherent Read Memory Behavior for Unregistered Output

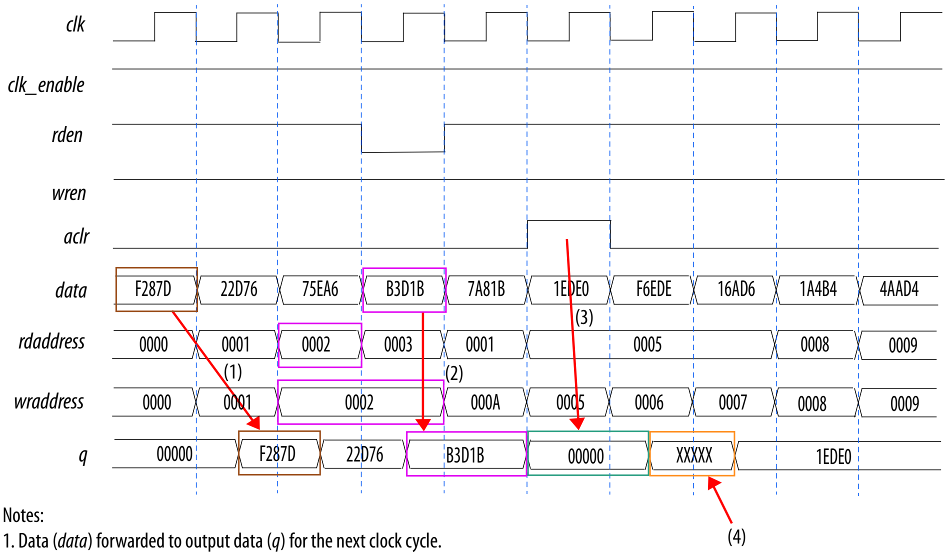
This figure shows the waveform of the coherent read memory when the output is unregistered.



Notes:
1. Data (*data*) forwarded to output data (*q*) at the same clock cycle.
2. *rden* is low, *q* holds the forwarded value.
3. One clock cycle is needed to recover the *q* after clear (*aclr*)
for the unregistered condition.

Figure 10. Coherent Read Memory Behavior for Registered Output

This figure shows the waveform of the coherent read memory when the output is registered.



Notes:

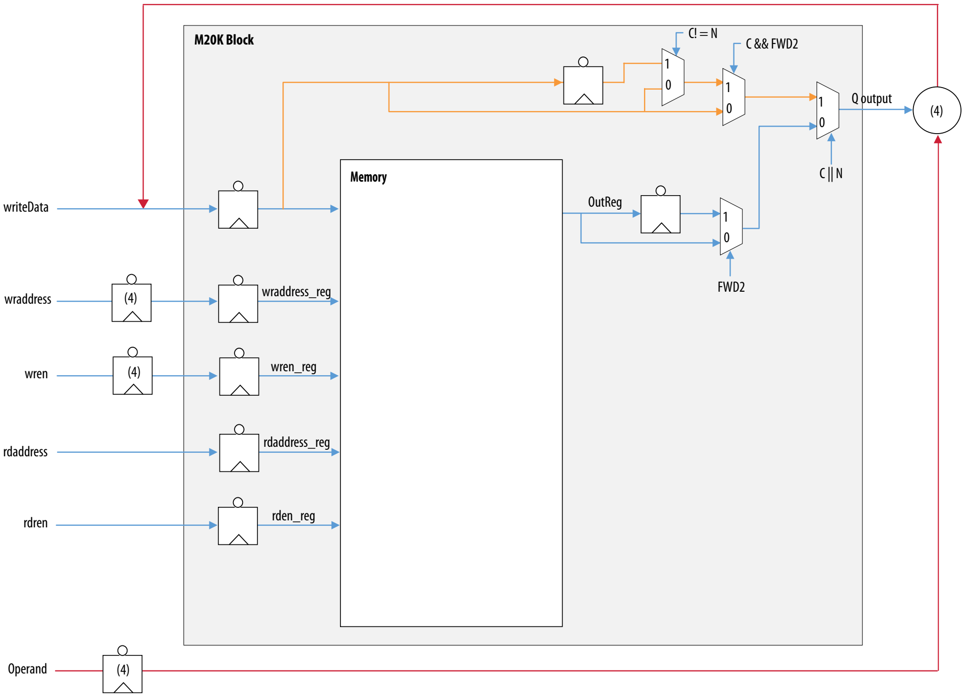
1. Data (*data*) forwarded to output data (*q*) for the next clock cycle.
2. *data* forwarded to *q* at the same clock cycle as current *wraddress* is the same as previous *rdaddress*.
3. When clear (*aclr*) is asserted, the M20K block clears *q*.
4. At this interval, *q* will latch from the M20K block, which is a Don't Care value, due to the read-during-write operation.

2.6.1. Forwarding Logic

In pipelining, you can use forwarding logic to perform data forwarding to reduce instruction cycles.

With coherent read feature and forwarding logic, you can coherently read out the data, perform operations (arithmetic or logical or both) on top of the data content, and write the data back to the same memory location within a single clock cycle.

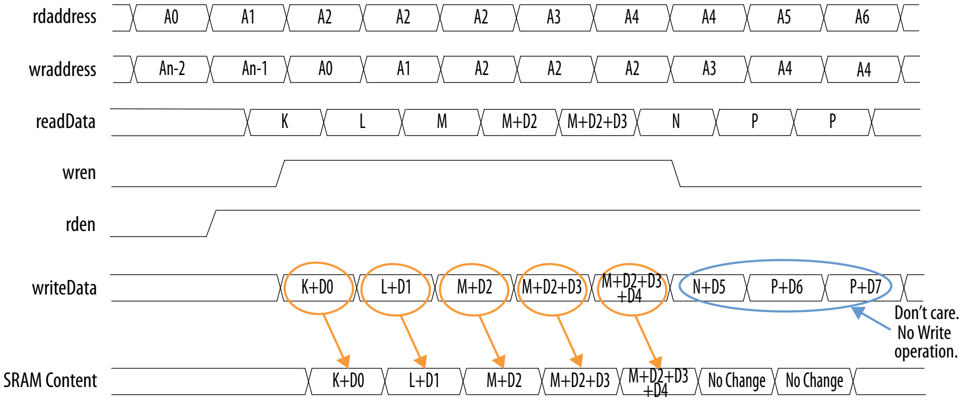
Figure 11. Example Forwarding Logic with Simplified Coherent Read Memory Circuitry



Notes:
1. N = (rden_reg && wren && (rdaddress_reg == wraddress)) ? 1 : 0.
2. C = (rden_reg && wren_reg && (rdaddress_reg == wraddress_reg)) ? 1 : 0.
3. FWD2 = (forwarding_stage2) ? 1 : 0.
4. External user logic added to achieve data forwarding.

Figure 12. Pipelining Waveform When Output of M20K Blocks is Unregistered

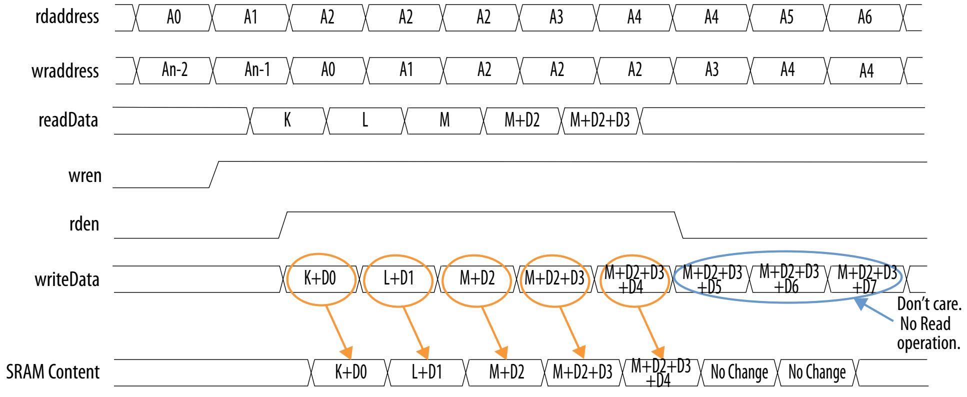
This figure shows the waveform of the pipelining with the read enable (`rden`) signal is high.



Note: All plus signs shown in this figure are example arithmetic operations performed on the data.

Figure 13. Pipelining Waveform When Output of M20K Blocks is Registered

This figure shows the waveform of the pipelining with the write enable (`wren`) signal is high.



Note: All plus signs shown in this figure are example arithmetic operations performed on the data.

With the coherent read feature enabled and forwarding logic implemented, the output of M20K blocks can be either unregistered or registered. To match the latency of the coherency circuitry within the hardware boundary of the M20K blocks, you may need to manually add the additional pipeline registers on the `wren` and `wraddress` paths, which is described in the following table:

Table 9. Pipeline Registers Requirements

Output Register	Additional Pipeline Registers on `wren` and `wraddress`
Unregistered	0
Registered	1

2.7. Freeze Logic

The freeze logic feature specifies whether to implement clock-enable circuitry for use in a partial reconfiguration region.

This feature is applicable only to the RAM modes:

- Single-port RAM
- Dual-port RAM
- Quad-port RAM

You have the option to turn on **Implement clock-enable circuitry for use in a partial reconfiguration** to enable the freeze logic feature in the parameter editors of the RAM IP cores.

2.8. True Dual Port Dual Clock Emulator

The true dual port (TDP) dual clock emulator feature emulates a TDP dual clock mode. This feature provides backward compatibility with Intel Arria® 10 devices, which supports TDP dual clock mode.

This feature is supported only in the following conditions:

- Two read/write ports operation mode.
- Customize clocks for A and B ports clocking mode.

Note: You must turn on **Emulate TDP dual clock mode** to enable the TDP dual clock emulator feature in the parameter editors of the dual-port RAM IP core. Refer to Table 25 on page 41 for more information about how to enable this feature.

The TDP dual clock emulator consists of two DCFIFOs and a single RAM block. The DCFIFO handles clock domain crossing (CDC) issues for the control signals and is a temporary buffer for data storage before and after being processed by the RAM block.

Due to the non-deterministic latency caused by different clock frequencies, a valid signal is introduced to identify whether the output data is valid. When the valid signal is asserted, it indicates that you should adhere to the correct output data. If the valid signal is de-asserted, discard the output data.

Table 10. Differences between Intel Arria 10 TDP Dual Clock Mode and Intel Stratix 10 Emulated TDP Dual Clock Mode

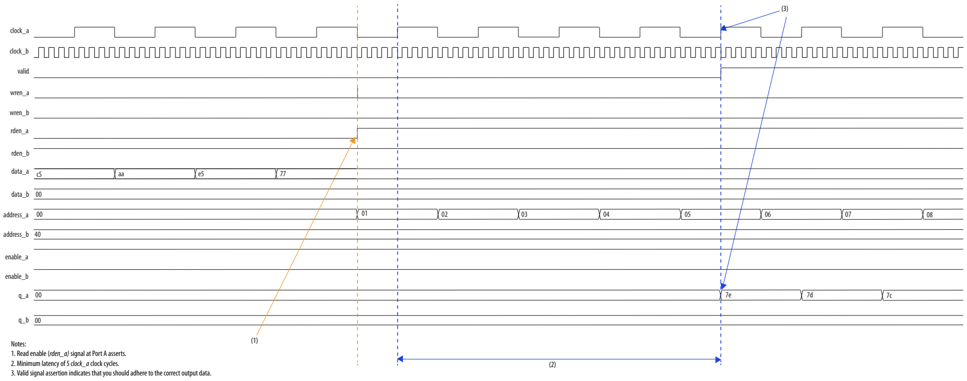
Signal	Intel Arria 10 TDP Dual Clock Mode	Intel Stratix 10 Emulated TDP Dual Clock Mode
clocken	Supported	Supported
rden	Supported	Supported
wren	Supported	Supported
aclr	Supported	—
sclr	—	—
byteena	Supported	—

The clock connection to Port A must be a slow clock (clock A) and the clock connection to Port B must be a fast clock (clock B), with a clock frequency ratio of clock B divided by clock A is greater than or equal to seven. This clock frequency ratio ensures a minimum latency of 5 clock cycle for Port A. The latency will not be guaranteed if the ratio is less than 7.

When you engage the TDP dual clock emulator feature, port A and port B will have different latency. The latency for port A decreases as the difference between the two clock frequencies increase, with a minimum latency of five clock cycles. Port B latency is fixed to two clock cycles, with the output registers always enabled for this configuration.

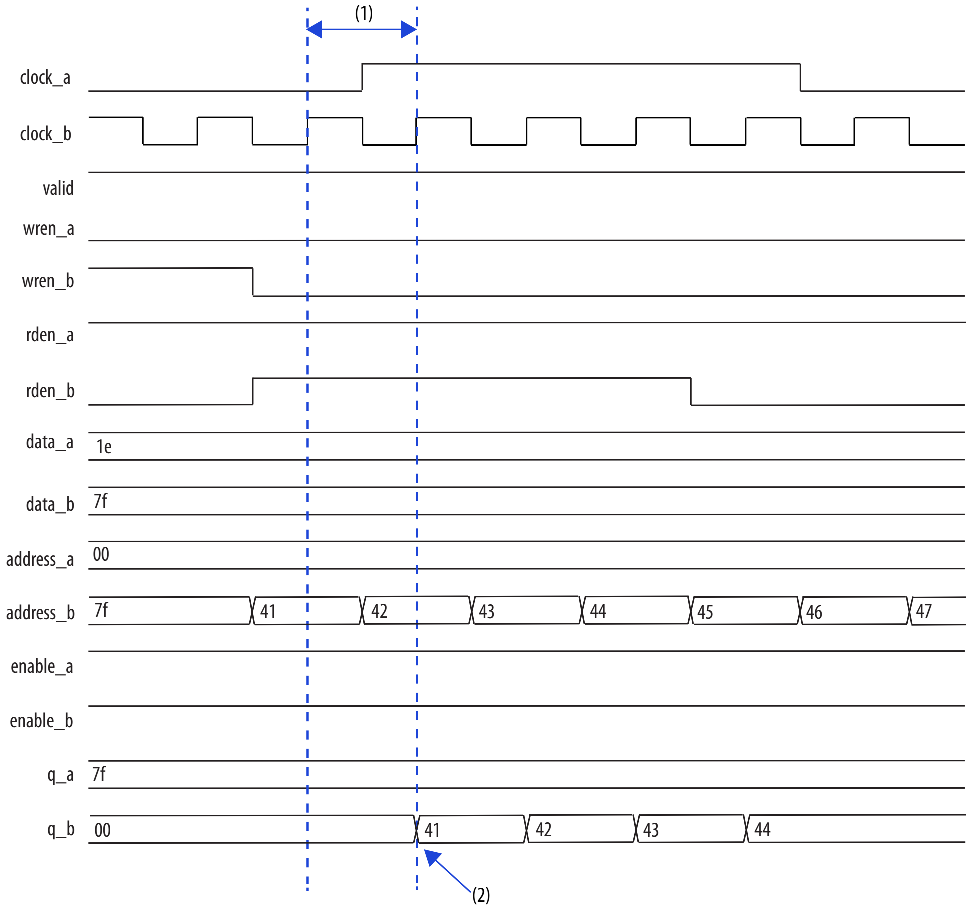
The following figures show the timing diagrams for the TDP dual clock emulator feature.

Figure 14. Output Condition of Port A



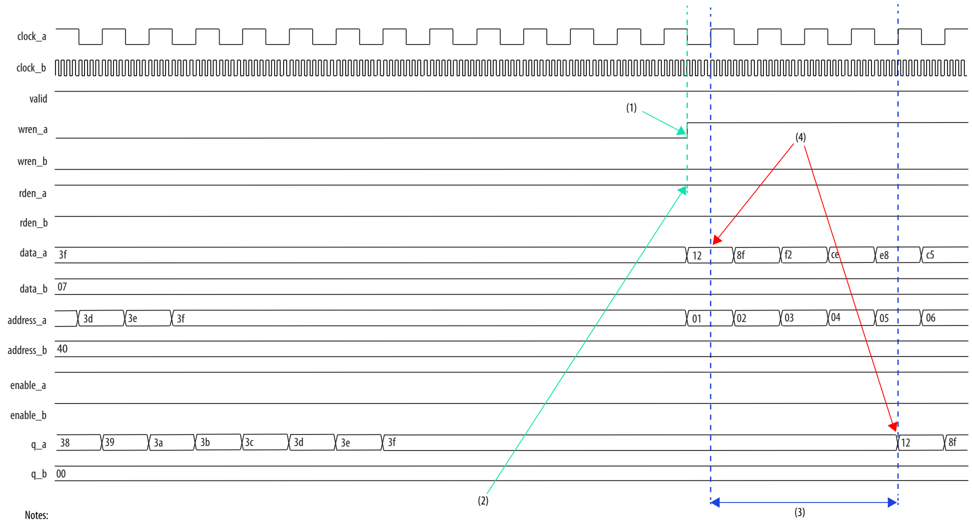
Notes:
1. Read enable (*rden_a)* signal at Port A asserts.
2. Minimum latency of 5 *clock_a* clock cycles.
3. Valid signal assertion indicates that you should adhere to the correct output data.

Figure 15. Output Condition of Port B



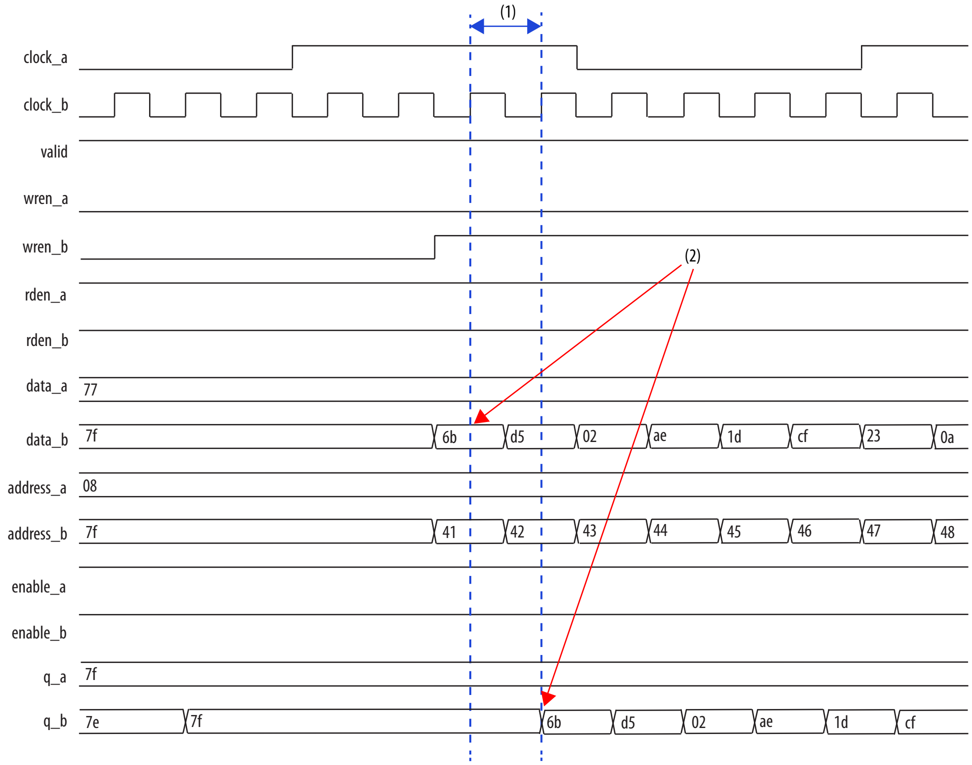
Notes:
1. Latency of 2 *clock_b* clock cycle.
2. Valid data output (*q_b*) at Port B.

Figure 16. Read-During-Write Condition of Port A



Notes:
1. Write enable (*wren_a*) signal at Port A asserts.
2. Read enable signal (*rden_a*) signal at Port A is already high.
3. Latency of 5 *clock_a* clock cycles.
4. Port A same port RDW occurs and the flow through value appears as data output (*q_a*) at Port A after 5 *clock_a* clock cycles.

Figure 17. Read-During-Write Condition of Port B



Notes:
1. Latency of 2 *clock_b* clock cycles.
2. Port B same port RDW occurs and the flow through value appears as data output (*q_b*) at Port B after 2 *clock_b* clock cycles.

2.9. Intel Stratix 10 Supported Embedded Memory IP Cores

Table 11. Intel Stratix 10 Memory IP Cores

This table lists and describes the IP cores that are supported in the Intel Stratix 10 embedded memory blocks.

IP Core	Supported Memory Mode	M20K Support	MLAB Support	Description
RAM: 1-PORT Intel FPGA IP	Single-port RAM	Yes	Yes	You can perform only one read or one write operation at a time. Use the read enable port to control the RAM output ports behavior during a write operation: • To retain the previous values that are held during the most recent active read enable—create a read-enable port and perform the write operation with the read enable port deasserted. • To show the new data being written, the old data at that address, or a *Don't Care* value when read-during-write occurs at the same address location—do not create a read-enable signal, or activate the read enable during a write operation.
RAM: 2-PORT Intel FPGA IP	Simple dual-port RAM	Yes	Yes	You can simultaneously perform one read and one write operations to different locations where the write operation happens on port A and the read operation happens on port B.
RAM: 2-PORT Intel FPGA IP	True dual-port RAM	Yes	No	You can perform any combination of two port operations: two reads, two writes, or one read and one write at single clocking mode.
RAM: 4-PORT Intel FPGA IP	Simple quad-port RAM	Yes	No	You can simultaneously perform two read and two write operations to different locations where the write addresses are specified at `address_a` and `address_b` signal/port, and the read addresses are specified at `address2_a` and `address2_b` signal/port.
ROM: 1-PORT Intel FPGA IP	Single-port ROM	Yes	Yes	Only one address port is available for read operation. You can use the memory blocks as ROM. • Initialize the ROM contents of the memory blocks using a **.mif** or **.hex**. • The address lines of the ROM are registered on M20K blocks but can be unregistered on MLABs. • The outputs can be registered or unregistered. • The output registers can be asynchronously or synchronously cleared. • The ROM read operation is identical to the read operation in the single-port RAM configuration.
ROM: 2 PORT Intel FPGA IP	Dual-port ROM	Yes	No	The dual-port ROM has almost similar functional ports as single-port ROM. The difference is dual-port ROM has an additional address port for read operation. You can use the memory blocks as a ROM. • Initialize the ROM contents of the memory blocks using a **.mif** or **.hex**. • The address lines of the ROM are registered on M20K blocks. • The outputs can be registered or unregistered. • The output registers can be asynchronously or synchronously cleared. • The ROM read operation is identical to the read operation in the true dual-port RAM configuration.

continued...

IP Core	Supported Memory Mode	M20K Support	MLAB Support	Description
Shift Register (RAM-based) Intel FPGA IP	—	Yes	Yes	Use the memory blocks as a shift register block to save logic cells and routing resources. This mode is useful in DSP applications that require local data storage such as finite impulse response (FIR) filters, pseudo-random number generators, multi-channel filtering, and auto- and cross- correlation functions. Traditionally, the local data storage is implemented with standard flip-flops that exhaust many logic cells for large shift registers. The input data width (`w`), the length of the taps (`m`), and the number of taps (`n`) determine the size of a shift register (`w × m × n`). You can cascade memory blocks to implement larger shift registers.
FIFO Intel FPGA IP FIFO2 Intel FPGA IP	—	Yes	Yes	You can use the memory blocks as FIFO buffers. Use the SCFIFO and DCFIFO functions to implement single- and dual-clock asynchronous FIFO buffers in your design. For designs with many small and shallow FIFO buffers, the MLABs are ideal for the FIFO mode. However, the MLABs do not support mixed-width FIFO mode.

Caution: To avoid corrupting the memory contents, do not violate the setup time or hold time on any of the embedded memory block input registers during read and write operations. This limitation is applicable if you use the memory blocks in single-port RAM, simple dual-port RAM, true dual-port RAM, simple quad-port RAM, or ROM mode.

Related Information

- RAM-Based Shift Register (ALTSHIFT_TAPS) IP Core User Guide
- Intel Stratix 10 Embedded Memory IP Core References on page 37

2.10. Intel Stratix 10 Embedded Memory Clocking Modes

Each Intel Stratix 10 embedded memory operation mode has supporting clocking modes.

Table 12. Memory Blocks Clocking Modes Supported for Each Memory Mode

Clocking Mode	Memory Mode					
	Single-Port	Simple Dual-Port	True Dual-Port	Simple Quad-Port	Single-Port ROM	Dual-Port ROM
Single clock mode	Yes	Yes	Yes	Yes	Yes	Yes
Read/write clock mode	N/A	Yes	N/A [1]	N/A	N/A	N/A
Input/output clock mode	Yes	Yes	Yes	N/A [2]	Yes	Yes

[1] The read/write clock mode is done through emulated true dual-port. For more information about the emulated true dual-port, refer to the *True Dual Port Dual Clock Emulator* section.

[2] Both input and output modes share the same clock.

Note: The clock enable signals are supported for write address, byte enable, and data input registers on MLAB blocks.

2.10.1. Single Clock Mode

In the single clock mode, a single clock, together with a clock enable, controls all registers of the embedded memory block.

2.10.2. Read/Write Clock Mode

In the read/write clock mode, a separate clock is available for each read and write port.

- A read clock controls the data-output, read-address, and read-enable registers.
- A write clock controls the data-input, write-address, write-enable, and byte enable registers.

2.10.3. Input/Output Clock Mode

In input/output clock mode, a separate clock is available for each input and output port.

- An input clock controls all registers related to the data input to the embedded memory block including data, address, byte enables, read enables, and write enables.
- An output clock controls the data output registers.

2.10.4. Asynchronous/Synchronous Clears in Clocking Modes

In all clocking modes, asynchronous and synchronous clears are available only for output latches and output registers.

For the independent (read/write and input/output) clock modes, the asynchronous and synchronous clears are available on both ports.

2.10.5. Output Read Data in Simultaneous Read/Write

If you perform a simultaneous read/write to the same address location using the read/write clock mode, the output read data is unknown. If you require the output read data to be a known value, use single-clock or input/output clock mode and select the appropriate read-during-write behavior in the parameter editors of the RAM/ROM IP cores.

2.10.6. Independent Clock Enables in Clocking Modes

Independent clock enables are supported in the following clocking modes:

- Read/write clock mode—supported for both read and write clocks.
- Input/output clock mode—supported for the registers of both ports.

To save power, you can control the shutdown of a particular register using the clock enables.

2.11. Intel Stratix 10 Embedded Memory Configurations

Table 13. Supported Embedded Memory Block Configurations

This table lists the maximum configurations supported for the Intel Stratix 10 embedded memory blocks.

Embedded Memory Block	Depth (bits)	Programmable Width
MLAB	32	×16, ×18, or ×20
M20K	512	×32 or ×40 *Note:* For simple dual-port only.
	1024	×16 or ×20 *Note:* For simple dual-port and true dual-port.
	2048	×8 or ×10 *Note:* For simple dual-port, true dual-port, and simple quad-port.
eSRAM	2048×42 [3]	x72 [3]

Related Information

eSRAM Intel FPGA IP on page 59

2.11.1. Mixed-Width Port Configurations

The mixed-width port configuration is supported only in simple dual-port RAM memory operation modes.

Note: MLABs do not support mixed-width port configurations.

Table 14. Supported Mixed-width Ratios for Intel Stratix 10

Operation Mode	Mixed-width Ratio	
	Without Byte Enable	With Byte Enable
Simple dual-port	1, 2, 4, 8, 16, and 32 *Note:* 8, 16, and 32 are emulated. For emulated ratio, use the `.mif` dimension of the larger width port.	1, 2, and 4
True dual-port	1	1
Simple quad-port	1	1

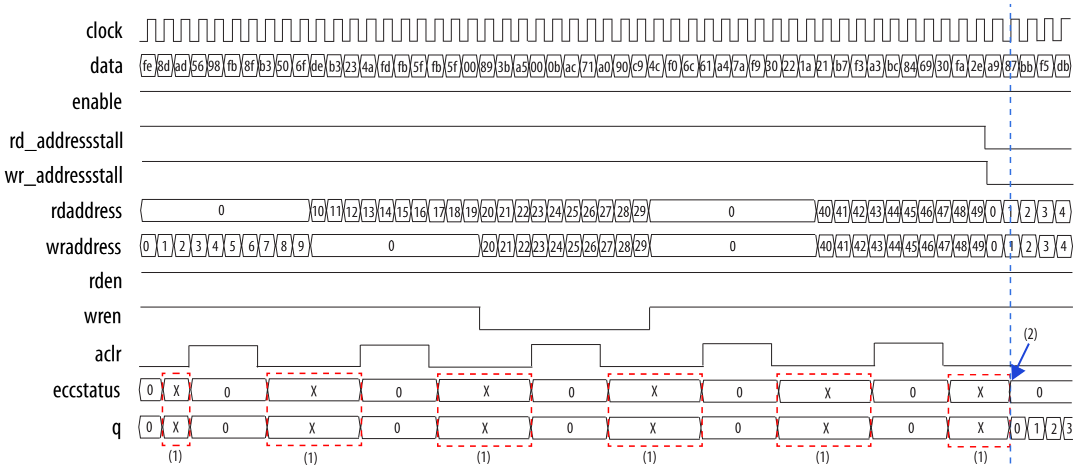
2.12. Initial Value of Read and Write Address Registers

In Intel Stratix 10 devices, the M20K blocks does not have freeze register (`frzreg`) in hardware to clear the address registers after entering user mode. This results in a non-deterministic address value in hardware before you can send any valid address. Hence, the address registers have been initialized to ‘X’ in the simulation model.

[3] The eSRAM channel depth and width can be programmably reduced to realize power savings. Refer to *eSRAM Intel FPGA IP* section for further details.

The figure below is a waveform illustrating the behavior of the values of the address registers initialized to ‘X’ for a simple dual-port RAM with registered output.

Figure 18. Simple Dual-Port RAM with Registered Output Timing Diagram



Notes:

1. Output data *q* and *eccstatus* are *dont_care* since the stalled read address value is non-deterministic and remains *dont_care* even after the output register is asynchronously cleared.
2. Output data *q* and *eccstatus* are now deterministic value since the read address is now a deterministic value. Output unregistered design will observe this behavior one clock cycle earlier.

3. Intel Stratix 10 Embedded Memory Design Considerations

There are several considerations that require your attention to ensure the success of your Intel Stratix 10 designs.

Note: Unless noted otherwise, these considerations apply to all variants of the Intel Stratix 10 device family.

3.1. Consider the Memory Block Selection

The Intel Quartus Prime software automatically partitions user-defined memory into the embedded memory blocks based on your design's speed and size constraints. For example, the Intel Quartus Prime software may spread out the memory across multiple available memory blocks to increase the performance of your design.

To assign the memory to a specific block size manually, use the parameter editor of the on-chip memory IP cores.

For the MLABs, you can implement single-port SRAM through emulation using the Intel Quartus Prime software. Emulation minimizes additional use of logic resources.

Because of the dual purpose architecture of the MLAB, the block has only data input registers, output registers, and write address registers. The MLABs gain read address registers from the ALMs.

Note:

1. For Intel Stratix 10 devices, the Resource Property Editor and the Timing Analyzer report the location of the M20K block as *EC_X<number>_Y<number>_N<number>*, even though the assigned location allowed is *M20K_X<number>_Y<number>_N<number>*. Embedded Cell (EC) is the sub-location of the M20K block.
2. When you select **AUTO** memory block type with clock enable port connected in the parameter editors of the RAM IP cores, the fitter will always choose M20K instead of MLAB.

3.2. Consider the Concurrent Read Behavior

The Intel Stratix 10 embedded memory blocks provide both corrupting and non-corrupting hardware behaviors using dual concurrent write operation on the same address. This feature is applicable if you use the memory blocks in true dual-port and single quad-port modes.

By default, the memory blocks will corrupt upon the dual concurrent write at the same address. To show a non-corrupting hardware behavior in the memory blocks, include the user-defined option “ENA_NON_CORRUPT=1” in the simulator setup script.

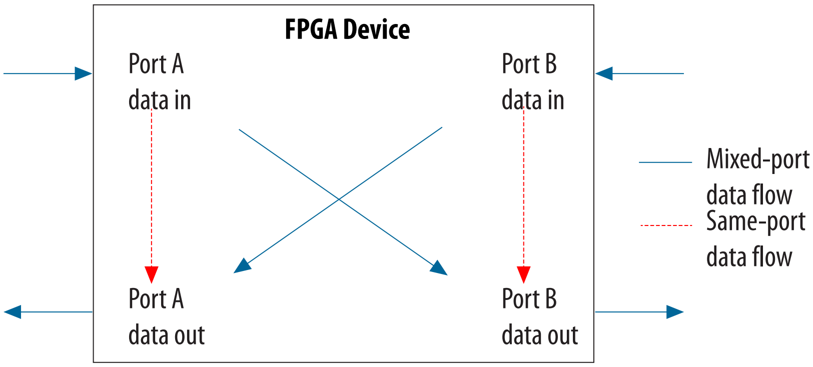
When the dual concurrent write occurs, the physical emulation uses a time-division multiplexing method to multiplex Port A and Port B together under the same data width. In this sequence, the value of Port B will be written first, followed by the value of Port A at the same address. This results the value of Port A being written to the memory.

3.3. Customize Read-During-Write Behavior

Customize the read-during-write behavior of the memory blocks to suit your design requirements.

Figure 19. Read-During-Write Data Flow

This figure shows the difference between the two types of read-during-write operations available: same port and mixed port.



If you are configuring same-port or mixed-port read-during-write mode with byte enable permutation in simple quad-port RAM, add `enable_vcs_sqp_be_rdw = 1` define flag in the VCS simulator.

3.3.1. Same-Port Read-During-Write Mode

The same-port read-during-write mode applies to a single-port RAM, simple quad-port RAM or the same port of a true dual-port RAM.

Table 15. Output Modes for Embedded Memory Blocks in Same-Port Read-During-Write Mode

This table lists the available output modes if you select the embedded memory blocks in the same-port read-during-write mode.

Output Mode	Memory Type	Description
New Data	M20K	The new data is available on the rising edge of the same clock cycle on which the new data is written.
Don't Care	M20K, MLAB	The RAM produces *Don't Care* values for a read-during-write operation. *Note:* For QUAD_PORT operating mode, the *Don't Care* mode is the only output mode for a same-port read-during-write-operation.

Figure 20. Same-Port Read-During-Write: New Data Mode

This figure shows sample functional waveforms of same-port read-during-write behavior in the *New Data* mode.

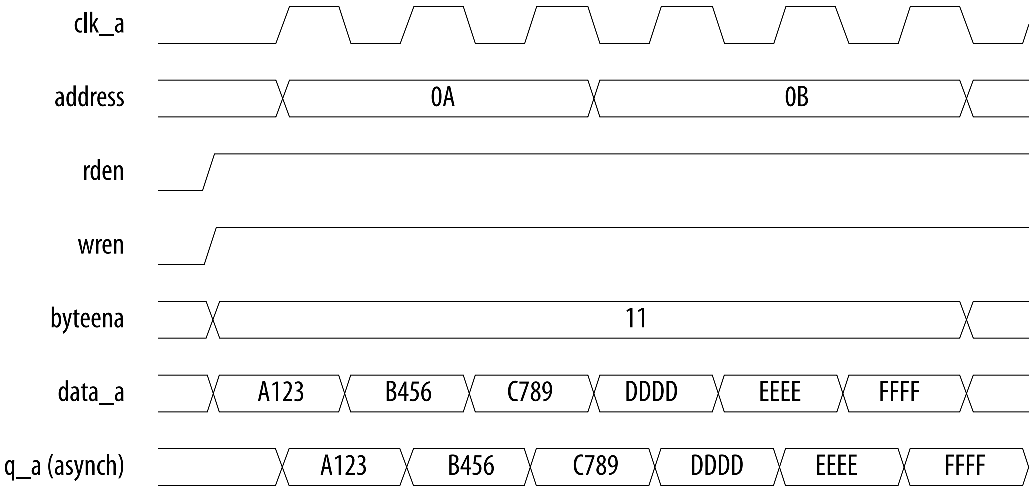
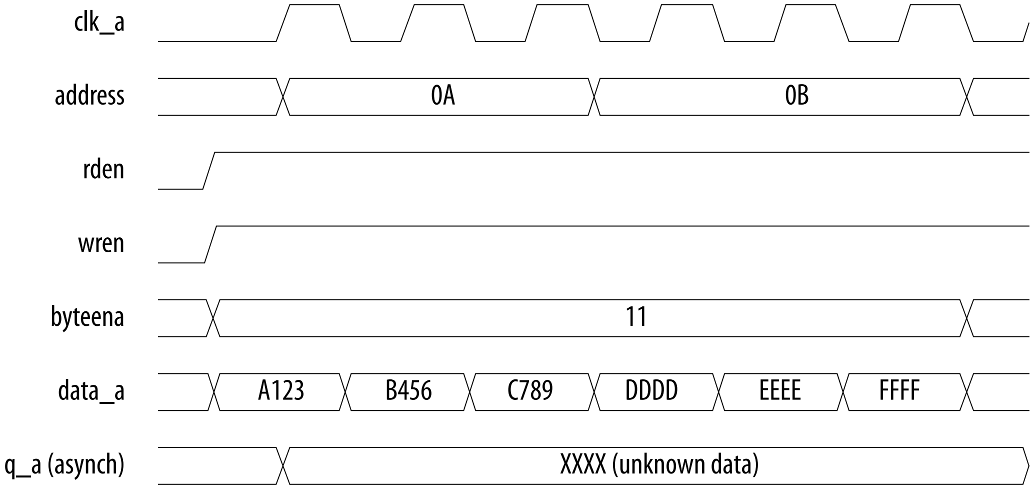


Figure 21. Same-Port Read-During-Write: Don't Care Mode

This figure shows sample functional waveforms of same-port read-during-write behavior in the *Don't Care* mode.



3.3.2. Mixed-Port Read-During-Write Mode

The mixed-port read-during-write mode applies to simple dual-port RAM mode. Two ports perform read and write operations on the same memory address using the same clock: one port reading from the address, and the other port writing to it.

Table 16. Output Modes for RAM in Mixed-Port Read-During-Write Mode

Output Mode	Memory Type	Description
New Data	MLAB	A read-during-write operation to different ports causes the MLAB registered output to reflect the *New Data* on the next rising edge after the data is written to the MLAB memory.

continued...

Output Mode	Memory Type	Description
		This mode is available only if the output is registered.
Old Data	M20K, MLAB	A read-during-write operation to different ports causes the RAM output to reflect the *Old Data* value at that particular address. For MLAB, this mode is available only if the output is registered.
Don't Care	M20K, MLAB	The RAM produces *Don't Care* or *Unknown* value. • For M20K, the Intel Quartus Prime software does not analyze the timing between write and read operations. • For MLAB, the Intel Quartus Prime software does not analyze the timing between write and read operations by default. To enable this behavior: — Turn off the **Do not analyze the timing between write and read operation. Metastability issues are prevented by never writing and reading at the same address at the same time** option in the embedded memory IP core parameter editor. or — Turn on the **MLAB Add Timing Constraints For Mixed-Port Feed-Through Mode Setting Don't Care** option in the **Advanced Fitter Setting**. *Note:* In M20K's true dual port operation, you will experience getting new data value during the mix-port read-during-write mode in simulation. When you set the output mode as *Don't Care*, the simulation value should treat it as a junk value.
New_a_old_b	M20K	This mode applicable only in simple-quad port for M20K where the read-during-write operation to different ports causes the RAM output to reflect new data at port A and old data at port B.

Table 17. Mixed Port Read-During-Write Output Behaviors

This table lists and describes the output behaviors of the mixed-port read-during-write mode. These behaviors are applicable only for MLAB blocks.

RAM: 2-PORT Intel FPGA IP Settings		Output Behavior		
Parameter	Enabled Parameter Options	altera_syncram Parameter (`read_during_write_mode_mixed_ports`)	Output Data when Read-During-Write	MLAB Atom (visible in Chip Planner)
Mixed Port Read-During-Write for Single Input Clock RAM	Old memory contents appear	`old_data`	Old data [4]	New Data
	New data	`new_data`	New data	New Data
	I do not care (The outputs will be undefined)	`constrained_dont_care`	Don't care [5]	Constrained Don't Care

continued...

[4] Old data is achieved through external soft logic as the MLAB blocks only natively supports new data.

[5] The output data is don't care because the IP does not guarantee metastability for the output data when read-during-write.

RAM: 2-PORT Intel FPGA IP Settings		Output Behavior		
Parameter	Enabled Parameter Options	altera_syncram Parameter (read_during_write_mode_mixed_ports)	Output Data when Read-During-Write	MLAB Atom (visible in Chip Planner)
How should the q_a and q_b outputs behave when reading a memory location that is being written from the other ports?	- I do not care (The outputs will be undefined) - Do not analyze the timing between write and read operation. Metastability issues are prevented by never writing and reading at the same address at the same time.	`dont_care`	Don't care [5]	Don't Care

Figure 22. Mixed-Port Read-During-Write: *New Data* Mode

This figure shows a sample functional waveform of mixed-port read-during-write behavior for the *New Data* mode.

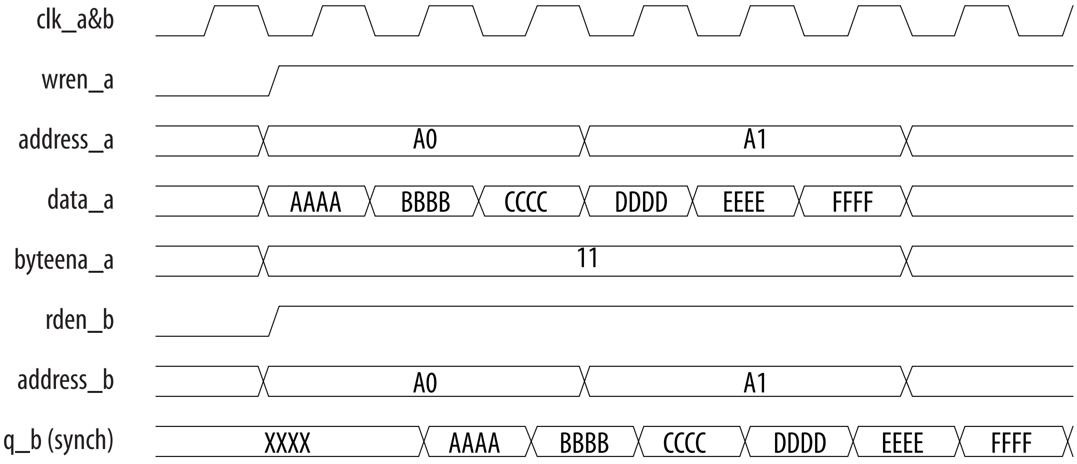


Figure 23. Mixed-Port Read-During-Write: *Old Data* Mode

This figure shows a sample functional waveform of mixed-port read-during-write behavior for the *Old Data* mode.

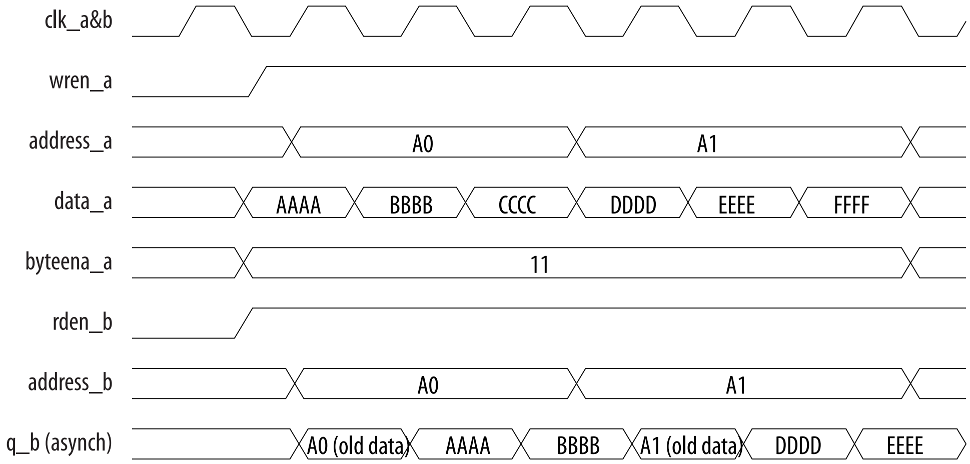


Figure 24. Mixed-Port Read-During-Write: *Don't Care* Mode

This figure shows a sample functional waveform of mixed-port read-during-write behavior for the *Don't Care* mode. This behavior is only applicable for M20K blocks.

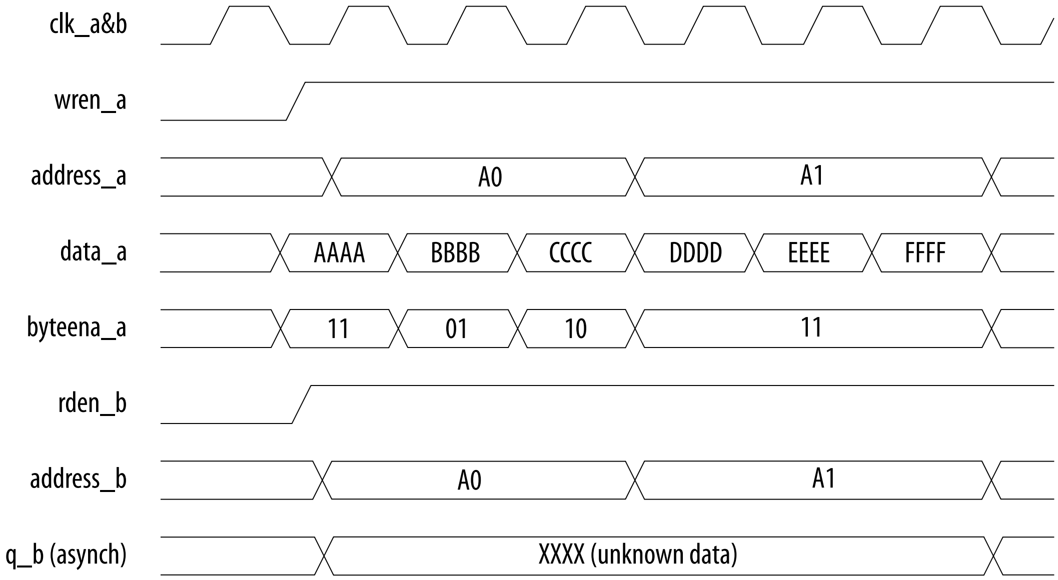
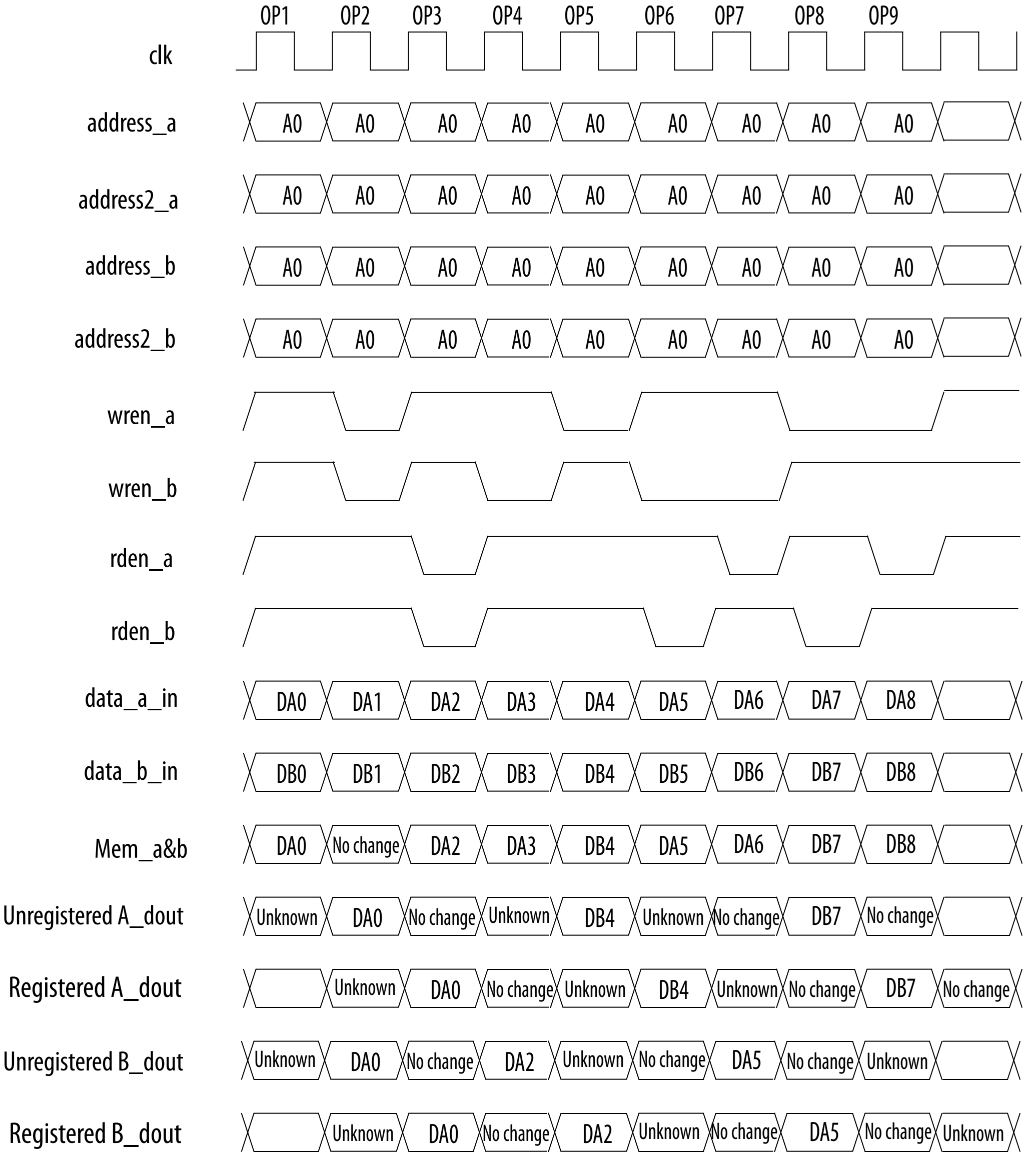


Figure 25. Mixed-Port Read-During-Write: *New_a_old_b* Mode

This figure shows a sample functional waveform of mixed-port read-during-write behavior for the *New_a_old_b* mode.


OP1 OP2 OP3 OP4 OP5 OP6 OP7 OP8 OP9
clk
address_a
A0 A0 A0 A0 A0 A0 A0 A0 A0
address2_a
A0 A0 A0 A0 A0 A0 A0 A0 A0
address_b
A0 A0 A0 A0 A0 A0 A0 A0 A0
address2_b
A0 A0 A0 A0 A0 A0 A0 A0 A0
wren_a
wren_b
rden_a
rden_b
data_a_in
DA0 DA1 DA2 DA3 DA4 DA5 DA6 DA7 DA8
data_b_in
DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7 DB8
Mem_a&b
DA0 No change DA2 DA3 DB4 DA5 DA6 DB7 DB8
Unregistered A_dout
Unknown DA0 No change Unknown DB4 Unknown No change DB7 No change
Registered A_dout
Unknown DA0 No change Unknown DB4 Unknown No change DB7 No change
Unregistered B_dout
Unknown DA0 No change DA2 Unknown No change DA5 No change Unknown
Registered B_dout
Unknown DA0 No change DA2 Unknown No change DA5 No change Unknown

Notes:

1. When the same-port read-during-write and mixed-port read-during-write behaviors exist simultaneously (OP1), the single quad-port will honor the same-port read-during-write behavior.
2. When the same-port read-during-write behavior happens, the output should be unknown.

3.4. Consider Power-Up State and Memory Initialization

Consider the power-up state of the different types of memory blocks if your design logic evaluates the initial power-up values.

Table 18. Initial Power-Up Values of Embedded Memory Blocks

Memory Type	Output Registers	Power-Up Value
MLAB	Used	Zero (cleared)
	Bypassed	Read memory contents
M20K	Used	Zero (cleared)
	Bypassed	Zero (cleared)
eSRAM	Used	Undefined

By default, the Intel Quartus Prime software initializes the embedded memory block in Intel Stratix 10 devices to zero, unless you specify in the memory contents in a `.mif`.

The MLAB and M20K embedded memory blocks support initialization with a `.mif`. You can create `.mif` files in the Intel Quartus Prime software and specify their use with the on-chip memory IP core when you create an instance of a memory in your design. Even if a memory is pre-initialized (for example, using a `.mif`), the memory still powers up with its output cleared.

3.5. Reduce Power Consumption

Reduce alternating current (AC) power consumption of each memory block in your design:

- Use the Intel Stratix 10 memory block clock enables to control the clocking of each embedded memory block.
- Use the read enable signal to ensure that read operations occur only when necessary. If your design does not require read-during-write, you can reduce power consumption by deasserting the read enable signal during write operations and when there are no memory operations.
- Use the Intel Quartus Prime software to automatically place any unused embedded memory blocks in low-power mode to reduce static power.

3.6. Avoid Providing Non-Deterministic Input

When running the embedded memory simulation model, you must ensure that you do not provide “X” or `dont_care` as inputs to the simulation model. Providing “X” or don’t_care may result in unexpected behavior in simulation.

3.7. Including the Reset Release Intel FPGA IP in Your Design

When using the eSRAM Intel FPGA IP, Intel requires that you either use the Reset Release Intel FPGA IP or the `INIT_DONE` signal route back through a pin to hold this IP in reset until configuration is complete.

To hold the eSRAM Intel FPGA IP in reset, connect the `c<`*channel_number*`>_sd_n_0` signal for this IP.

Related Information

Intel Stratix 10 Configuration User Guide
More information about the Reset Release Intel FPGA IP.

4. Intel Stratix 10 Embedded Memory IP Core References

You can access the features of the Intel Stratix 10 Embedded Memory using the On Chip Memory IP cores in the Intel Quartus Prime software.

The On Chip Memory IP cores include:

- RAM: 1-Port Intel FPGA IP—instantiates the single-port RAM
- RAM: 2-Port Intel FPGA IP—instantiates the dual-port and bidirectional-port RAM
- RAM: 4-Port Intel FPGA IP—instantiates the quad-port RAM
- ROM: 1-Port Intel FPGA IP—instantiates the single-port ROM
- ROM: 2-Port Intel FPGA IP—instantiates the dual-port and bidirectional-port ROM
- eSRAM (Embedded Synchronous Random Access Memory) Intel FPGA IP—instantiates the native eSRAM block
- FIFO (First-In-First-Out) Intel FPGA IP—instantiates the FIFO Intel FPGA IP core
- FIFO2 Intel FPGA IP—instantiates the FIFO2 Intel FPGA IP core

The information for each IP core parameter is described in the parameter editors in the Intel Quartus Prime software.

Related Information

- Introduction to Intel IP Cores
  Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- Creating Version-Independent IP and Qsys Simulation Scripts
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- Project Management Best Practices
  Guidelines for efficient management and portability of your project and IP files.

4.1. On Chip Memory RAM and ROM Intel FPGA IPs

On Chip Memory Intel FPGA IPs	Features
RAM: 1-PORT Intel FPGA IP	• Non-simultaneous read and write operations from a single address. • Read enable port to specify the behavior of the RAM output ports during a write operation, to overwrite or retain existing value. • Emulates single-port ROM using DUAL_PORT configuration for block RAM.
RAM: 2-PORT Intel FPGA IP	Simple dual-port RAM • Simultaneous one read and one write operations to different locations. • Supports error correction code (ECC). • Emulates single-port RAM using DUAL_PORT configuration for block RAM.

continued...

On Chip Memory Intel FPGA IPs	Features
	True dual-port RAM • Simultaneous two reads. • Simultaneous two writes. • Simultaneous one read and one write at two different clock frequencies.
RAM: 4-PORT Intel FPGA IP	• Simultaneous two reads and two writes to different locations.
ROM: 1-PORT Intel FPGA IP	• One port for read-only operations.
ROM: 2-PORT Intel FPGA IP	• Two ports for read-only operations. • Emulates dual-port ROM using BIDIR_DUAL_PORT configuration for block RAM.

4.1.1. Release Information for RAM and ROM Intel FPGA IPs

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP version (X.Y.Z) number may change from one Intel Quartus Prime software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 19. RAM: 1-PORT Intel FPGA IP Current Release Information

Item	Description
IP Version	20.1.0
Intel Quartus Prime Version	20.1
Release Date	2020.04.13

Table 20. RAM: 2-PORT Intel FPGA IP Current Release Information

Item	Description
IP Version	20.1.0
Intel Quartus Prime Version	20.1
Release Date	2020.04.13

Table 21. RAM: 4-PORT Intel FPGA IP Current Release Information

Item	Description
IP Version	20.1.0
Intel Quartus Prime Version	20.1
Release Date	2020.04.13

Table 22. ROM: 1-PORT Intel FPGA IP Current Release Information

Item	Description
IP Version	20.1.0
Intel Quartus Prime Version	20.1
Release Date	2020.04.13

Table 23. ROM: 2-PORT Intel FPGA IP Current Release Information

Item	Description
IP Version	20.1.0
Intel Quartus Prime Version	20.1
Release Date	2020.04.13

4.1.2. RAM: 1-PORT Intel FPGA IP Parameters

This table lists the parameters for the RAM: 1-PORT Intel FPGA IP core.

Table 24. RAM: 1-PORT Intel FPGA IP Parameters Description

Parameter	Legal Values	Description
Parameter Settings: Widths/Blk Type/Clks		
How wide should the ‘q’ output bus be?	—	Specifies the width of the ‘q’ output bus.
How many words of memory?	—	Specifies the number of bit words.
What should the memory block type be?	Auto, MLAB, M20K, LCs	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
Set the maximum block depth to	• Auto: Auto, 32, 64, 128, 256, 512, 1024, 2048, 4096 • MLAB: Auto, 32 • M20K: Auto, 512, 1024, 2048 • LCs: Auto	Specifies the maximum block depth in words.
How should the memory be implemented?	• Use default logic cell style • Use Stratix M512 emulation logic cell style	Specifies the logic cell implementation method. • Select **Use default logic cell style** if you prefer smaller and faster memory capacity. • Select **Use Stratix M512 emulation logic cell style** if you prefer the memory to be compatible to the Stratix M512 emulation style.
What clocking method would you like to use?	• Single clock • Dual clock: use separate ‘input’ and ‘output’ clocks	Specifies the clocking method to use. • **Single clock**—A single clock and a clock enable controls all registers of the memory block. • **Dual clock: use separate ‘input’ and ‘output’ clocks**—An input and an output clock controls all registers related to the data input and output to/from the memory block including data, address, byte enables, read enables, and write enables.

continued...

Parameter		Legal Values	Description
Parameter Settings: Regs/Clken/Byte Enable/Aclrs			
Which ports should be registered? The following options are available: • 'data' and 'wren' input ports • 'address' input port • 'q' output port		On/Off	Specifies whether to register the input and output ports.
Create one clock enable signal for each clock signal. *Note:* All registered ports are controlled by the enable signal(s).		On/Off	Specifies whether to turn on the option to create one clock enable signal for each clock signal.
More Options	Use clock enable for port A input registers	On/Off	Specifies whether to use clock enable for port A input registers.
	Use clock enable for port A output registers	On/Off	Specifies whether to use clock enable for port A output registers.
	Create an 'addressstall_a' input port.	On/Off	Specifies whether to create an `addressstall_a` input port. You can create this port to act as an extra active low clock enable input for the address registers.
Create byte enable for port A		On/Off	Specifies whether to create a byte enable for port A. Turn on this option if you want to mask the input data so that only specific bytes, nibbles, or bits of data are written. To enable byte enable for port A and port B, the data width ratio has to be 1 or 2 for the RAM: 1-PORT and RAM: 2-PORT Intel FPGA IP cores.
What is the width of a byte for byte enables?		• MLAB: 5 or 10 • Other memory block types: 8 or 9 • M20K: 8, 9, or 10	Specifies the byte width of the byte enable port. The width of the data input port must be divisible by the byte size.
Create an 'aclr' asynchronous clear for the registered ports. 'q' port		On/Off	Specifies whether the registered ports are affected by an asynchronous clear port.
Create an 'sclr' synchronous clear for the registered ports. 'q' port		On/Off	Specifies whether the registered ports are affected by a synchronous clear port.
Create a 'rden' read enable signal		On/Off	Turn on if you want to create a read enable signal.
Parameter Settings: Read During Write Option			
What should the 'q' output be when reading from a memory location being written to?		Don't Care, Old Data	Specifies the output behavior when read-during-write occurs. **Don't Care**—The RAM outputs "don't care" or "unknown" values for read-during-write operation. **Old Data**—The RAM outputs reflect the old data at that address before the write operation proceeds.
Get x's for write masked bytes instead of old data when byte enable is used		On/Off	Turn on this option to obtain 'X' on the masked byte.

continued...

Parameter	Legal Values	Description
Parameter Settings: Mem Init		
Do you want to specify the initial content of the memory?	• No, leave it blank • Yes, use this file for the memory content data	Specifies the initial content of the memory. To initialize the memory to zero, select **No, leave it blank**. To use a memory initialization file (`.mif`) or a hexadecimal (Intel-format) file (`.hex`), select **Yes, use this file for the memory content data.**
Initialize memory content data to XX..X on power-up in simulation	On/Off	—
Implement clock-enable circuitry for use in a partial reconfiguration region	On/Off	Specifies whether to implement clock-enable circuitry for use in a partial reconfiguration region.
Allow In-System Memory Content Editor to capture and update content independently of the system clock	On/Off	Specifies whether to allow In-System Memory Content Editor to capture and update content independently of the system clock.
The 'Instance ID' of this RAM is	NONE	Specifies the RAM ID.
Parameter Settings: Performance Optimization		
Enable Force To Zero	On/Off	Specifies whether to set the output to zero when you deassert the read enable signal. Enabling this feature helps improve the glue logic performance when the selected memory depth is larger than a single memory block.

4.1.3. RAM: 2-PORT Intel FPGA IP Parameters

This table lists the parameters for the RAM: 2-PORT Intel FPGA IP core.

Table 25. RAM: 2-PORT Intel FPGA IP Parameter Settings

Parameter	Legal Values	Description
Parameter Settings: General		
How will you be using the dual port RAM?	Operation mode: • With one read port and one write port • With two read /write ports	Specifies how you use the dual port RAM.
How do you want to specify the memory size?	Type: • As a number of words • As a number of bits	Determines whether to specify the memory size in words or bits.
Parameter Settings: Widths/ Blk Type		
How many words of memory?	—	Specifies the number of words.
Use different data widths on different ports	On/Off	Specifies whether to use different data widths on different ports.
When you select **With one read port and one write port** or **With two read/write ports**, the following options are available:	—	Specifies the width of the input and output ports.

continued...

Parameter	Legal Values	Description
• How wide should the 'q_a' output bus be? • How wide should the 'data_a' input bus be? • How wide should the 'q_b' output bus be?		
Ram block type	Auto, MLAB, M20K, LCs	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
Set the maximum block depth to	• Auto: Auto, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384 • MLAB: Auto, 32 • M20K: Auto, 512, 1024, 2048 • LCs: Auto	Specifies the maximum block depth in words.
How should the memory be implemented?	• Use default logic cell style • Use Stratix M512 emulation logic cell style	Specifies the logic cell implementation method. • Select default logic cell style if you prefer smaller and faster memory capacity. • Select Stratix M512 emulation logic cell style if you prefer the memory to be compatible to the Stratix M512 emulation style. *Note:* This option is applicable only when you choose LCs memory type.
Parameter Settings: Clks/Rd, Byte En		
What clocking method would you like to use?	• Single clock • Dual clock: use separate 'input' and 'output' clocks • Dual clock: use separate 'read' and 'write' clocks • Customize clocks for A and B ports	Specifies the clocking method to use. • **Single clock**—A single clock and a clock enable controls all registers of the memory block. • **Dual Clock: use separate 'input' and 'output' clocks**—An input and an output clock controls all registers related to the data input and output to/from the memory block including data, address, byte enables, read enables, and write enables. • **Dual clock: use separate 'read' and 'write' clock**—A write clock controls the data-input, write-address, and write-enable registers while the read clock controls the data-output, read-address, and read-enable registers. • **Dual clock: use separate clocks for A and B ports**—Clock A controls all registers on the port A side; clock B controls all registers on the port B side. Each port also supports independent clock enables for both port A and port B registers, respectively. • **Customize clocks for A and B ports**—To use this option, the **Emulate TDP dual clock mode** option must also be enabled.

continued...

Parameter	Legal Values	Description
When you select **With two read/ write ports** and **Customize clocks for A and B ports** clocking method, the following option is available: Emulate TDP dual clock mode	—	Specifies whether to emulate a TDP dual clock mode. The clock connection to Port A must be a slow clock and the clock connection to Port B must be a fast clock.
When you select **With one read port and one write port**, the following option is available: Create a 'rden' read enable signal	—	Specifies whether to create a read enable signal for port B.
When you select **With two read/ write ports**, the following option is available: Create a 'rden_a' and 'rden_b' read enable signals		Specifies whether to create a read enable signal for port A and B.
Create byte enable for port A	—	Specifies whether to create a byte enable for port A and B. Turn on these options if you want to mask the input data so that only specific bytes, nibbles, or bits of data are written. To enable byte enable for port A and port B, the data width ratio has to be 1 or 2 for the RAM: 1-PORT and RAM: 2-PORT Intel FPGA IP cores. The option to create a byte enable for port B is only available when you select the **With two read/write ports** option.
Create byte enable for port B	—	
What is the width of a byte for byte enables?	• MLAB: 5 or 10 • Other memory block types: 8 or 9 • M20K: 8, 9, or 10	Specifies the width of a byte for byte enables. This option is only available when you select the **Create byte enable for port A** and/or **Create byte enable for port B** option(s).
Enable Error Correction Check (ECC)	On/Off	Specifies whether to enable the ECC feature that corrects single bit errors, double adjacent bit errors, and triple adjacent bit errors at the output of the memory.
Enable ECC Pipeline Registers	On/Off	Specifies whether to enable the ECC Pipeline Registers before the output decoder to achieve that same performance as non-ECC mode at the expense of one cycle of latency.
Enable ECC Encoder Bypass	On/Off	Specifies whether to enable the ECC encoder bypass feature that allows you to selectively insert parity bits into the memory through eccencparity port.
Enable Coherent Read	On/Off	Specifies whether to enable the coherent read feature to present with coherent memory read. This feature allows you to read out current memory content, perform operation on top of the content, and write back to the same location in the same cycle.
Parameter Settings: Regs/Clkens/Aclrs		
Which ports should be registered?	On/Off	Specifies whether to register the read or write input and output ports.

continued...

Parameter	Legal Values	Description
When you select **With one read port and one write port**, the following options are available: • All write input ports • raddress port • q_b port When you select **With two read/ write ports**, the following options are available: • All write input ports • raddress port • q_a port • q_b port		
Clock Enables When you select **With one read port and one write port**, the following option is available: • Use different clock enables for registers • Use clock enable for write input registers • Use clock enable for read input registers • Use clock enable for output registers When you select **With two read / write ports**, the following options are available: • Use different clock enables for registers • Use clock enable for port A input registers • Use clock enable for port A output registers • Use clock enable for port B input registers • Use clock enable for port B output registers	On/Off	Specifies whether to create clock enables for read and write registers. *Note:* For MLAB blocks, a non-deterministic output may be produced in the first clock cycle when you select the **Use clock enable for output registers** option.
Addressstalls When you select **With one read port and one write port**, the following option is available: • Create a 'addressstall_a' input port.	On/Off	Specifies whether to create clock enables for address registers. You can create these ports to act as an extra active low clock enable input for the address registers.
Aclr Options When you select **With one read port and one write port**, the following option is available: • q_b port When you select **With two read / write ports**, the following options are available: • q_a port • q_b port	On/Off	Specifies whether to create an asynchronous clear port for the registered ports. Specifies whether the 'q_a' and 'q_b' ports are cleared by the aclr port.
Sclr Options	On/Off	Specifies whether to create a synchronous clear port for the registered ports. Specifies whether the 'q_a', and 'q_b' ports are cleared by the sclr port.

continued...

Parameter	Legal Values	Description
When you select **With one read port and one write port**, the following option is available: • q_b port When you select **With two read / write ports**, the following options are available: • q_a port • q_b port		*Note:* For MLAB blocks, a non-deterministic output may be produced in the first clock cycle when you select the **q_b port** option.
Parameter Settings: Output 1 (This tab is only available when you select one read port and one write ports)		
How should the q_a and q_b outputs behave when reading a memory location that is being written from the other port?	• New Data • Old memory contents appear • I do not care (The outputs will be undefined)	Specifies the output behavior when read-during-write occurs. • **New Data**—New data is available on the rising edge of the same clock cycle on which it was written. • **Old memory contents appear**—The RAM outputs reflect the old data at that address before the write operation proceeds. • **I do not care**—This option functions differently when you turn it on depending on the following memory block type you select: — When you set the memory block type to **Auto**, **M20K**, or any other block RAM, the RAM outputs 'don't care' or "unknown" values for read-during-write operation without analyzing the timing path. — When you set the memory block type to **MLAB** (for LUTRAM), the RAM outputs 'don't care' or 'unknown' values for read-during-write operation but analyzes the timing path to prevent metastability.
Do not analyze the timing between write and read operation. Metastability issues are prevented by never writing and reading at the same address at the same time.	On/Off	Turn on this option when you want the RAM to output 'don't care' or unknown values for read-during-write operation without analyzing the timing path. This option is only available for LUTRAM and is enabled when you set memory block type to **MLAB.**
Parameter Settings: Output 2 (This tab is only available when you select two read/ write ports)		
What should the 'q_a' output be when reading from a memory location being written to?	• New data • Old Data	Specifies the output behavior when read-during-write occurs. • **New Data**—New data is available on the rising edge of the same clock cycle on which it was written. • **Old Data**—The RAM outputs reflect the old data at that address before the write operation proceeds.
What should the 'q_b' output be when reading from a memory location being written to?		
Get x's for write masked bytes instead of old data when byte enable is used	On/Off	Turn on this option to obtain 'X' on the masked byte.
Parameter Settings: Mem Init		

continued...

Parameter	Legal Values	Description
Do you want to specify the initial content of the memory?	• No, leave it blank • Yes, use this file for the memory content data	Specifies the initial content of the memory. To initialize the memory to zero, select **No, leave it blank.** To use a memory initialization file (**.mif**) or a hexadecimal (Intel-format) file (**.hex**), select **Yes, use this file for the memory content data**.
Initialize memory content data to XX..X on power-up in simulation	On/Off	—
The initial content file should conform to which port's dimensions?	PORT_A, PORT_B	If you select to use the initial content file for memory content data, select the port the file should conform to.
Implement clock-enable circuitry for use in a partial reconfiguration region	On/Off	Specifies whether to implement clock-enable circuitry for use in a partial reconfiguration region. Implement clock-enable circuitry for use in a partial reconfiguration region
Parameter Settings: Performance Optimization		
Enable Force to Zero	On/Off	Specifies whether to set the output to zero when you deassert the read enable signal. Enabling this feature helps improve the glue logic performance when the selected memory depth is larger than a single memory block.

4.1.4. RAM: 4-PORT Intel FPGA IP Parameters

This table lists the parameters for the RAM: 4-PORT Intel FPGA IP core.

Table 26. RAM: 4-PORT Intel FPGA IP Parameter Settings

Parameter	Legal Values	Description
Parameter Settings: Widths/ Blk Type		
How many words of memory?	—	Specifies the number of bit words.
How wide should the 'q_a' and 'q_b' output bus be?	—	Specifies the width of the input and output ports.
RAM block type	Auto, M20K	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
Set the maximum block depth to	• Auto: Auto, 512, 1024, 2048 • M20K: Auto, 512, 1024, 2048	Specifies the maximum block depth in words.
Parameter Settings: Clks/Rd, Byte En		
What clocking method would you like to use?	Single clock	Specifies the clocking method to use. **Single clock**—A single clock and a clock enable controls all registers of the memory block.
Create 'rden_a' and 'rden_b' read enable signals	—	Specifies whether to create a read enable signal for ports A and B.

continued...

Parameter	Legal Values	Description
What is the width of a byte for byte enables?	M20K: 5, 8, 9, 10	Specifies the byte width of the byte enable port. The width of the data input port must be divisible by the byte size.
Parameter Settings: Regs/Clkens/Aclrs		
Which ports should be registered? Input registers: • All write input ports • ‘raddress’ port Output registers: • ‘q_a’ port • ‘q_b’ port	On/Off	Specifies whether to register the read or write input and output ports.
Use clock enable for input and output registers.	On/Off	Specifies whether to turn on the option to create one clock enable signal for the input and output registers.
Create an ‘aclr’ asynchronous clear for the output ports. Output Aclrs: • ‘q_a‘ port • ‘q_b‘ port	On/Off	Specifies whether to create an asynchronous clear port for the output ports. Output Aclrs: • **q_a port**—Specifies whether the q_a port is cleared by the aclr port. • **q_b port**—Specifies whether the q_b port is cleared by the aclr port.
Create an ‘sclr’ synchronous clear for the output ports. Output Sclrs: • ‘q_a’ port • ‘q_b’ port	On/Off	Specifies whether to create a synchronous clear port for the output ports. Output Sclrs: • **q_a port**—Specifies whether the q_a port is cleared by the sclr port. • **q_b port**—Specifies whether the q_b port is cleared by the sclr port.
Parameter Settings: Output 1		
How should the ‘q_a’ and ‘q_b’ outputs behave when reading a memory location that is being written from the other port? The output of port A will be ‘NEW’ while the output of port B will be ‘OLD’	On/Off	Specifies the output behavior when read-during-write occurs.
Parameter Settings: Output 2		
What should the ‘q_a’ output be when reading from a memory location being written to?	Don't Care	Specifies the output behavior when read-during-write occurs.
What should the ‘q_b’ output be when reading from a memory location being written to?		
Parameter Settings: Mem Init		
Do you want to specify the initial content of the memory?	• No, leave it blank • Yes, use this file for the memory content data	Specifies the initial content of the memory. To initialize the memory to zero, select **No, leave it blank.**

continued...

Parameter	Legal Values	Description
		To use a memory initialization file (**.mif**) or a hexadecimal (Intel-format) file (**.hex**), select **Yes, use this file for the memory content data**.
Initialize memory content data to XX..X on power-up simulation	On/Off	—
The initial content file should conform to which port's dimensions?	PORT_A, PORT_B	If you select to use the initial content file for memory content data, select the port the file should conform to.
Implement clock-enable circuitry for use in a partial reconfiguration region	On/Off	Specifies whether to implement clock-enable circuitry for use in a partial reconfiguration region.
Parameter Settings: Performance Optimization		
Enable Force-to-Zero	On/Off	Specifies whether to set the output to zero when you deassert the read enable signal. Enabling this feature helps improve the glue logic performance when the selected memory depth is larger than a single memory block.

4.1.5. ROM: 1-PORT Intel FPGA IP Parameters

This table lists the parameters for the ROM: 1-PORT Intel FPGA IP core.

Table 27. ROM: 1-PORT Intel FPGA IP Parameter Settings

Parameter	Legal Values	Description
Parameter Settings: General Page		
How wide should the ‘q’ output bus be?	—	Specifies the width of the ‘q’ output bus.
How many words of memory?	—	Specifies the number of words.
What should the memory block type be?	Auto, MLAB, M20K	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
Set the maximum block depth to	• MLAB: Auto, 32 • M20K: Auto, 512, 1024, 2048	Specifies the maximum block depth in words.

continued...

Parameter	Legal Values	Description
What clocking method would you like to use?	- Single clock - Dual clock: use separate 'input' and 'output' clocks	Specifies the clocking method to use. - **Single clock**—A single clock and a clock enable controls all registers of the memory block - **Dual clock: use separate 'input' and 'output' clocks**—The input clock controls the address registers and the output clock controls the data-out registers. There are no write-enable, byte-enable, or data-in registers in ROM mode.
Parameter Settings: Regs/Clken/Aclrs		
Which ports should be registered? The following options are available: - 'address' input port - 'q' output port	On/Off	Specifies whether to register the input and output ports.
Use clock enable for port A input registers	On/Off	Specifies whether to use clock enable for port A input registers.
Use clock enable for port A output registers	On/Off	Specifies whether to use clock enable for port A output registers.
Create an 'addressstall_a' input port	On/Off	Specifies whether to create an addressstall_a input port. You can create this port to act as an extra active low clock enable input for the address registers.
Create an 'aclr' asynchronous clear for the registered ports. The following option is available: - 'q' port	On/Off	Specifies whether the registered ports be affected by an asynchronous clear port.
Create a 'sclr' asynchronous clear for the registered ports. - 'q' port	On/Off	Specifies whether the q port be affected by a synchronous clear port.
Create an 'rden' read enable signal	On/Off	Specifies whether to create a read enable signal.
Parameter Settings: Mem Init		
Do you want to specify the initial content of the memory?	- No, leave it blank - Yes, use this file for the memory content data	Specifies the initial content of the memory. In ROM mode, you must specify a memory initialization file (**.mif**) or a hexadecimal (Intel-format) file (**.hex**). The **Yes, use this file for the memory content data** option is turned on by default.
The initial content file should conform to which port's dimensions?	PORT_A	The initial content file for memory content data only conforms to port A.

continued...

Parameter	Legal Values	Description
Allow In-System Memory Content Editor to capture and update content independently of the system clock	On/Off	Specifies whether to allow In-System Memory Content Editor to capture and update content independently of the system clock
The 'Instance ID' of this ROM is	NONE	Specifies the ROM ID.
Parameter Settings: Performance Optimization		
Enable Force-to-Zero	On/Off	Specifies whether to set the output to zero when you deassert the read enable signal. Enabling this feature helps improve the glue logic performance when the selected memory depth is larger than a single memory block.

4.1.6. ROM: 2-PORT Intel FPGA IP Parameters

This table lists the parameters for the ROM: 2-PORT Intel FPGA IP core.

Table 28. ROM: 2-PORT Intel FPGA IP Parameter Settings

Parameter	Legal Values	Description
Parameter Settings: Widths/Blk Type		
How do you want to specify the memory size?	• As a number of words • As a number of bits	Determines whether to specify the memory size in words or bits.
How many words of memory?	32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536	Specifies the number of words.
Use different data widths on different ports	On/Off	Specifies whether to use different data widths on different ports.
How wide should the 'q_a' output bus be?	—	Specifies the width of the 'q_a' and 'q_b' output ports.
How wide should the 'q_b' output bus be?		
RAM block type	Auto, M20K	Specifies the memory block type. The types of memory block that are available for selection depends on your target device
Set the maximum block depth to:	• Auto: Auto, 512, 1024, 2048, 4096 • M20K: Auto, 512, 1024, 2048	Specifies the maximum block depth in words. This option is enabled only when you choose **Auto** as the memory block type.
Parameter Settings: Clks/Rd		

continued...

Parameter	Legal Values	Description
What clocking method would you like to use?	• Single clock • Dual clock: use separate 'input' and 'output' clocks • Customize clocks for A and B ports	Specifies the clocking method to use. • **Single**—A single clock and a clock enable controls all registers of the memory block • **Dual clock: use separate 'input' and 'output' clocks**—The input clock controls the address registers and the output clock controls the data-out registers. There are no write-enable, byte-enable, or data-in registers in ROM mode. • **Customize clocks for A and B ports**—Clock A controls all registers on the port A side; clock B controls all registers on the port B side. Each port also supports independent clock enables for both port A and port B registers, respectively.
Create a 'rden_a' and 'rden_b' read enable signals	On/Off	Specifies whether to create read enable signals.
Parameter Settings: Regs/Clkens/Aclrs		
Which ports should be registered? Read output ports	On/Off	Specifies whether to register the read output ports.
More Options Registered Q Output Ports • 'q_a' port • 'q_b' port	On/Off	Turn on if you want the registered 'q_a' and 'q_b' ports to be affected by the asynchronous clear signal. • **q_a port**—Specifies whether to register the 'a_b' output port. • **q_b port**—Specifies whether to register the 'q_b' output port.
Use clock enable for port A input registers	On/Off	Specifies whether to use clock enable for port A input registers.
Use clock enable for port A output registers	On/Off	Specifies whether to use clock enable for port A output registers.
Use clock enable for port B input registers	On/Off	Specifies whether to use clock enable for port B input registers.
Use clock enable for port B output registers	On/Off	Specifies whether to use clock enable for port B output registers.
Aclr Options • 'q_a' port • 'q_b' port	On/Off	Specifies whether the registered ports should be cleared by the asynchronous clear port.
Sclr Options • 'q_a' port • 'q_b' port	On/Off	Specifies whether the registered ports should be cleared by the synchronous clear port.
Parameter Settings: Mem Init		
Do you want to specify the initial content of the memory?	• No, leave it blank • Yes, use this file for the memory content data	Specifies the initial content of the memory. In ROM mode, you must specify a memory initialization file (**.mif**) or a hexadecimal (Intel-format) file (**.hex**). The **Yes, use this file for the memory content data** option is turned on by default.

continued...

Parameter	Legal Values	Description
The initial content file should conform to which port's dimensions?	• PORT_A • PORT_B	Specifies whether the initial content file conforms to port A or port B.
Parameter Settings: Performance Optimization		
Enable Force-to-Zero	On/Off	Specifies whether to set the output to zero when you deassert the read enable signal. Enabling this feature helps improve the glue logic performance when the selected memory depth is larger than a single memory block.

4.1.7. RAM and ROM Interface Signals

Table 29. Interface Signals of the Intel Stratix 10 RAM and ROM Intel FPGA IP Cores

Signal	Direction	Required	Description
data_a	Input	Optional	Data input to port A of the memory. The data_a port is required for all RAM operation modes: • SINGLE_PORT • DUAL_PORT • BIDIR_DUAL_PORT • QUAD_PORT
address_a	Input	Yes	Address input to port A of the memory. The address_a signal is required for all operation modes.
address2_a	Input	Yes (for simple quad-port)	Read address input to port A of the memory. The address2_a signal is required if the operation_mode parameter is set to QUAD_PORT.
wren_a	Input	Optional	Write enable input for address_a port. The wren_a signal is required all RAM operation modes: • SINGLE_PORT • DUAL_PORT • BIDIR_DUAL_PORT • QUAD_PORT
rden_a	Input	Optional	Read enable input for address_a port. The rden_a signal is supported depending on your selected memory mode and memory block.
byteena_a	Input	Optional	Byte enable input to mask the data_a port so that only specific bytes, nibbles, or bits of the data are written. The byteena_a port is not supported in the following conditions: • If implement_in_les parameter is set to ON • If operation_mode parameter is set to ROM
addressstall_a	Input	Optional	Address clock enable input to hold the previous address of address_a port for provided that the addressstall_a port is high.
q_a	Output	Yes	Data output from port A of the memory.

continued...

Signal	Direction	Required	Description
			The q_a port is required if the operation_mode parameter is set to any of the following values: • SINGLE_PORT • BIDIR_DUAL_PORT • QUAD_PORT • ROM The width of q_a port must be equal to the width of data_a port.
data_b	Input	Optional	Data input to port B of the memory. The data_b port is required if the operation_mode parameter is set to BIDIR_DUAL_PORT and QUAD_PORT.
address_b	Input	Optional	Address input to port B of the memory. The address_b port is required if the operation_mode parameter is set to the following values: • DUAL_PORT • BIDIR_DUAL_PORT • QUAD_PORT
address2_b	Input	Yes (for simple quad-port)	Read address input to port B of the memory. The address2_b is required if the operation_mode parameter is set to QUAD_PORT.
wren_b	Input	Yes	Write enable input for address_b port. The wren_b port is required if operation_mode is set to BIDIR_DUAL_PORT and QUAD_PORT.
rden_b	Input	Optional	Read enable input for address_b port. The rden_b port is supported depending on your selected memory mode and memory block
byteena_b	Input	Optional	Byte enable input to mask the data_b port so that only specific bytes, nibbles, or bits of the data are written. The byteena_b port is not supported in the following conditions: • If implement_in_les parameter is set to ON • If operation_mode parameter is set to SINGLE_PORT, DUAL_PORT, or ROM
q_b	Output	Yes	Data output from port B of the memory. The q_b port is required if the operation_mode is set to the following values: • DUAL_PORT • BIDIR_DUAL_PORT • QUAD_PORT The width of q_b port must be equal to the width of data_b port.
clock0	Input	Yes	The following describes which of your memory clock must be connected to the clock0 port, and port synchronization in different clocking modes:

continued...

Signal	Direction	Required	Description
			• Single clock: Connect your single source clock to `clock0` port. All registered ports are synchronized by the same source clock. • Read/Write: Connect your read clock to `clock0` port. All registered ports related to write operation, such as `data_a` port, `address_a` port, `wren_a` port, and `byteena_a` port are synchronized by the write clock. • Input Output: Connect your input clock to `clock0` port. All registered input ports are synchronized by the input clock. • Independent clock: Connect your port A clock to `clock0` port. All registered input and output ports of port A are synchronized by the port A clock.
`clock1`	Input	Optional	The following describes which of your memory clock must be connected to the `clock1` port, and port synchronization in different clocking modes: • Single clock: Not applicable. All registered ports are synchronized by `clock0` port. • Read/Write: Connect your read clock to `clock1` port. All registered ports related to read operation, such as `address_b` port and `rden_b` port are synchronized by the read clock. • Input Output: Connect your output clock to `clock1` port. All the registered output ports are synchronized by the output clock. • Independent clock: Connect your port B clock to `clock1` port. All registered input and output ports of port B are synchronized by the port B clock.
`clocken0`	Input	Optional	Clock enable input for `clock0` port.
`clocken1`	Input	Optional	Clock enable input for `clock1` port.
`eccstatus`	Output	Optional	A bit wide error correction status port. Indicate whether the data that is read from the memory has an error in single-bit with correction, fatal error with no correction, or no error bit occurs. The `eccstatus` port is supported if all the following conditions are met: • `operation_mode` parameter is set to `DUAL_PORT` • `ram_block_type` parameter is set to `M20K` • `width_a` and `width_b` parameter have the same value • Byte enable is not used
`eccencbypass`	Input	Optional	When active, this port allow user to inject parity flip bits through `eccencparity` ports. When inactive, parity flip bits will be generated using internal ecc encoder. This port can only be used when `enable_ecc_encoder_bypass` is set to "TRUE".
`eccencparity`	Input	Optional	When `eccencbypass` is active, user can inject 8-bit parity flip through `eccencparity` port. This port can be used only when `enable_ecc_encoder_bypass` is set to "TRUE".
`data`	Input	Yes	Data input to the memory. The data port is required and the width must be equal to the width of the q port.
`wraddress`	Input	Yes	Write address input to the memory.
`wren`	Input	Yes	Write enable input for `wraddress` port. The `wren` port is required.

continued...

Signal	Direction	Required	Description
rdaddress	Input	Yes	Read address input to the memory.
rden	Input	Optional	Read enable input for rdaddress port.
byteena	Input	Optional	Byte enable input to mask the data port so that only specific bytes, nibbles, or bits of data are written. It is supported in Intel Stratix 10 devices when you set the ram_block_type parameter to MLAB.
wraddressstall	Input	Optional	Write address clock enable input to hold the previous write address of wraddress port for as long as the wraddressstall port is high.
rdaddressstall	Input	Optional	Read address clock enable input to hold the previous read address of rdaddress port for as long as the rdaddressstall port is high.
q	Output	Yes	Data output from the memory.
inclock	Input	Yes	The following describes which of your memory clock must be connected to the inclock port, and port synchronization in different clocking modes: • Single clock: Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock. • Read/Write: Connect your write clock to inclock port. All registered ports related to write operation, such as data port, wraddress port, wren port, and byteena port are synchronized by the write clock. • Input/Output: Connect your input clock to inclock port. All registered input ports are synchronized by the input clock.
outclock	Input	Yes	The following describes which of your memory clock must be connected to the outclock port, and port synchronization in different clocking modes: • Single clock: Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock. • Read/Write: Connect your read clock to outclock port. All registered ports related to read operation, such as rdaddress port and rdren port are synchronized by the read clock. • Input/Output: Connect your output clock to outclock port. The registered q port is synchronized by the output clock.
inclocken	Input	Optional	Clock enable input for inclock port.
outclocken	Input	Optional	Clock enable input for outclock port.
aclr	Input	Optional	Asynchronously clear the output ports. The asynchronous clear effect on the registered ports can be controlled through their corresponding clear parameter, such as outdata_aclr_a and outdata_aclr_b.
sclr	Input	Optional	Synchronously clear the output ports. The synchronous clear effect on the registered ports can be controlled through their corresponding parameter, such as outdata_sclr_a and outdata_sclr_b.

Note: When running the embedded memory simulation model, you must ensure that you do not provide “X” or dont_care as inputs to the simulation model. Providing “X” or don’t_care may result in unexpected behavior in simulation.

4.1.8. Changing Parameter Settings Manually

When the IP core has been generated using the IP Parameter Editor, you can use this flow to change of the parameter settings within the specified memory mode. However, to change the memory mode, use the IP Parameter Editor to configure and regenerate the IP core.

Follow these steps to change the parameter settings manually:

1. Locate the Verilog design file: *<project directory>*/*<project name_software version>*/synth/*<projectName_coreName_QuartusVersion_random>*.v.
2. Change the parameter settings in the design file. Ensure that you use only legal parameter values as specified in Parameters and Signals topic. Failing to do so results in compilation errors.
3. Compile the design using the Intel Quartus Prime software.

For example, the following codes enable the ECC feature and specify the initialization file.

```
altera_syncram_component.enable_ecc = "TRUE",
altera_syncram_component.ecc_pipeline_stage_enabled = "FALSE",
altera_syncram_component.init_file = "mif1.mif",
```

To disable the ECC feature and specify a different .mif file, make the following changes.

```
altera_syncram_component.enable_ecc = "FALSE",
altera_syncram_component.ecc_pipeline_stage_enabled = "FALSE",
altera_syncram_component.init_file = "mif2.mif",
```

4.1.8.1. RAM and ROM Parameter Settings

Table 30. Parameters for altera_syncram

Use the parameter list when editing the design file manually.

Name	Legal Values	Description
operation_mode	**SINGLE_PORT** **DUAL_PORT** **BIDIR_DUAL_PORT** **QUAD_PORT** **ROM**	Operation mode of the memory block.
width_a	—	Data width of port A.
widthad_a	—	Address width of port A.
widthad2_a		Address 2 width of port A.
numwords_a	—	Number of data words in the memory block for port A.
outdata_reg_a	**UNREGISTERED** **CLOCK1** **CLOCK0**	Clock for the data output registers of port A.

continued...

Name	Legal Values	Description
outdata_aclr_a	**NONE** **CLEAR1** **CLEAR0**	Asynchronous clear for data output registers of port A. When the outdata_reg_a parameter is set to **UNREGISTERED**, this parameter specifies the clearing parameter for the output latch.
outdata_sclr_a	**NONE** **SCLEAR**	Synchronous clear for data output registers of port A. When the outdata_reg_a parameter is set to **NONE**, this parameter specifies the clearing parameter for the output latch.
address_aclr_a	**NONE**	Option to clear the address input registers of port A.
width_byteena_a	—	Width of the byte-enable bus of port A. The width must be equal to the value of width_a divided by the byte size. The default value of 1 is only allowed when byte-enable is not used.
width_b	—	Data width of port B.
widthad_b	—	Address width of port B.
widthad2_b	—	Address 2 width of port B.
numwords_b	—	Number of data words in the memory block for port B.
outdata_reg_b	**UNREGISTERED** **CLOCK1** **CLOCK0**	Clock for the data output registers of port B.
indata_reg_b	**CLOCK1** **CLOCK0**	Clock for the data input registers of port B.
address_reg_b	**CLOCK1** **CLOCK0**	Clock for the address registers of port B.
byteena_reg_b	**CLOCK1** **CLOCK0**	Clock for the byte-enable registers of port B.
outdata_aclr_b	**NONE** **CLEAR1** **CLEAR0**	Asynchronous clear for data output registers of port B. When the outdata_reg_b parameter is set to **UNREGISTERED**, this parameter specifies the clearing parameter for the output latch.
outdata_sclr_b	**NONE** **SCLEAR**	Synchronous clear for data output registers of port B. When the outdata_reg_b parameter is set to **NONE**, this parameter specifies the clearing parameter for the output latch.
address_aclr_b	**NONE**	Option to clear the address input registers of port B.
width_byteena_b	—	Width of the byte-enable bus of port B. The width must be equal to the value of width_b divided by the byte size. The default value of 1 is only allowed when byte-enable is not used.
intended_device_family	**"Stratix 10"**	Parameter used for simulation purpose.
ram_block_type	**AUTO** **M20K** **MLAB**	The memory block type.
byte_size	**5**	The byte size for the byte-enable mode.

continued...

Name	Legal Values	Description
	8 **9** **10**	
read_during_write_mode_mixed_ports	**DONT_CARE** **CONSTRAINT_DONT_CARE** **NEW_DATA** **OLD_DATA** **NEW_A_OLD_B**	The behavior for the read-during-write mode. • The default value is **DONT_CARE**. • The value of **NEW_DATA** is supported only when the read address and output data are registered by the write clock in the LUTRAM mode. • The value of **CONSTRAINED_DONT_CARE** is supported only in the LUTRAM mode. • The value of **NEW_A_OLD_B** is supported only when the operation_mode parameter is set to **QUAD_PORT**.
init_file	*.mif *.hex	The initialization file.
init_file_layout	**PORT_A** **PORT_B**	The layout of the initialization file.
maximum_depth	—	The depth of the memory block slices.
clock_enable_input_a	**NORMAL** **BYPASS**	The clock enable for the input registers of port A.
clock_enable_output_a	**NORMAL** **BYPASS**	The clock enable for the output registers of port A.
clock_enable_input_b	**NORMAL** **BYPASS**	The clock enable for the input registers of port B.
clock_enable_output_b	**NORMAL** **BYPASS**	The clock enable for the output registers of port B.
read_during_write_mode_port_a	**NEW_DATA_NO_NBE_READ** **NEW_DATA_WITH_NBE_READ** **OLD_DATA** **DONT_CARE**	The read-during-write behavior for port A.
read_during_write_mode_port_b	**NEW_DATA_NO_NBE_READ** **NEW_DATA_WITH_NBE_READ** **OLD_DATA** **DONT_CARE**	The read-during-write behavior for port B.
enable_ecc	**TRUE** **FALSE**	Enables or disables the ECC feature.
ecc_pipeline_stage_enabled	**TRUE** **FALSE**	• Specifies whether to enable ECC Pipeline Registers before the output decoder to achieve the same performance as non-ECC mode at the expense of one cycle of latency. • The parameter enable_ecc must set to **TRUE** if this parameter is set to **TRUE**. • The parameter outdata_reg_b cannot set to UNREGISTERED if this parameter is set to **TRUE**. • The default value is **FALSE**.

continued...

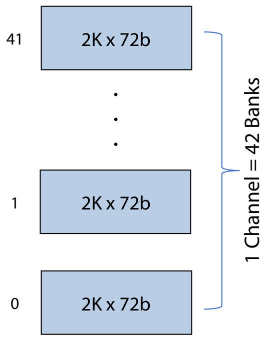
Name	Legal Values	Description
`enable_ecc_encoder_bypass`	**TRUE** **FALSE**	Enables or disables the ECC Encoder Bypass feature. • The parameter `enable_ecc` must set to **TRUE** if this parameter is set to **TRUE**.
`enable_coherent_read`	**TRUE** **FALSE**	Enables or disables the coherent read feature. • The default value is **FALSE**.
`enable_force_to_zero`	**TRUE** **FALSE**	Enables or disables the Force-to-Zero feature. • The default value is **FALSE**.
`width_eccencparity`	**8**	The width of the `eccencparity` signal.
`optimization_option`	**AUTO**	Specifies how the RAM block would be optimized. • If **AUTO** is selected, the fitter determines whether the RAM block is in High_Speed or Low_Power mode. • The RAM block type must be M20K when High_Speed or Low_Power is selected.

4.2. eSRAM Intel FPGA IP

The basic building block of the eSRAM Intel FPGA IP core is a bank, which consists of an array of 2K x 72-bit SRAM blocks.

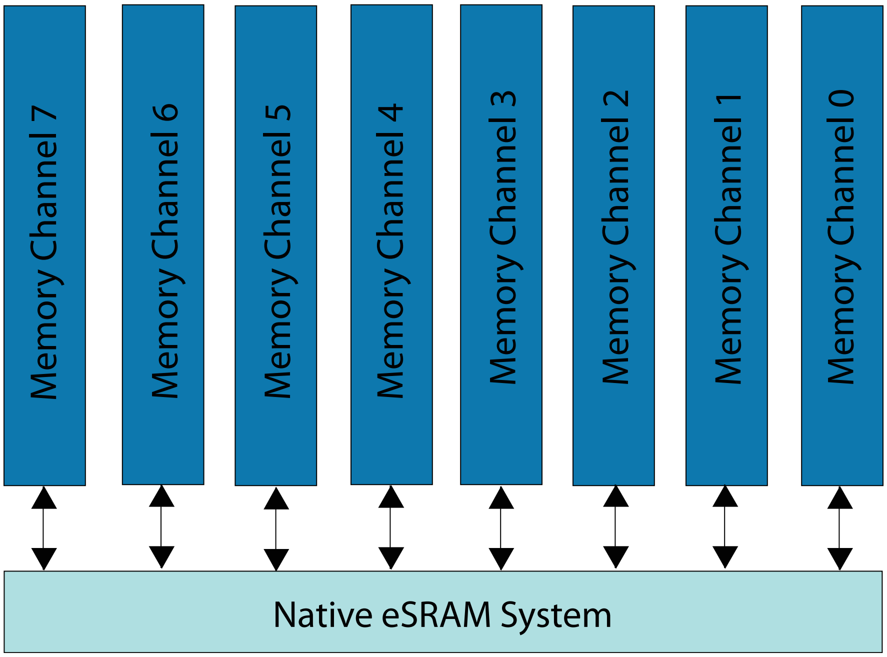
42 eSRAM banks combine to form a channel.

Figure 26. eSRAM Channel



Eight memory channels combine to form an eSRAM system.

Figure 27. eSRAM System



Related Information

Intel Stratix 10 Embedded Memory Configurations on page 27

4.2.1. Release Information for eSRAM Intel FPGA IP

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP version (X.Y.Z) number may change from one Intel Quartus Prime software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 31. eSRAM Intel FPGA IP Current Release Information

Item	Description
IP Version	19.1.4
Intel Quartus Prime Version	20.2
Release Date	2020.06.22

4.2.2. eSRAM System Features

An eSRAM system provides features for handling simultaneous read and write requests, ensuring data integrity and coherency, and maximizing power efficiency.

A given eSRAM system can achieve a maximum frequency of 750 MHz. The number of available eSRAM systems depends on the Intel Stratix 10 device in use.

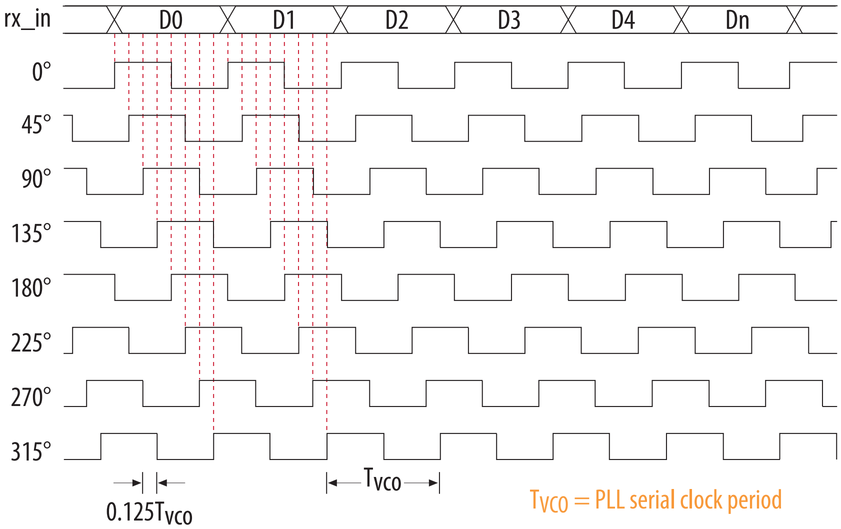
Every memory channel within an eSRAM system has one write port and one read port, which can handle simultaneous read and write requests. Each channel has access to only its own banks, thus ensuring that each channel is independent from its neighbors.

The eSRAM system has an error correction code (ECC) which you can enable at the cost of some user-accessible data capacity. The ECC can improve data integrity by encoding write data with extended Hamming code and decoding read data for Single-bit Error Correction, Double-bit Error Detection (SECDED). Write latency and read latency are the same whether ECC is enabled or not.

There is a data coherency feature called Write Forwarding which you can enable to handle simultaneous write and read access to the same eSRAM memory location. The write data on the write port is forwarded to the read port and not read from the targeted SRAM bank. The write data is still written into the targeted eSRAM bank.

A low power mode can conserve static power at the cost of 1 clock cycle. In addition, each channel can power down unused banks, for additional power savings.

The eSRAM system includes a PLL which natively drives the clock domains necessary for eSRAM operation.



4.2.2.1. eSRAM Specifications

The following table summarizes the specifications of the eSRAM Intel FPGA IP core.

Table 32. eSRAM Specifications

Feature	Detail	Value	Description
Clock Frequency [6]	-1 -2 -3	200 MHz - 750 MHz 200 MHz - 640 MHz 200 MHz - 500 MHz [7]	—
Bank Capacity	without ECC with ECC	144 Kb 128 Kb	Each bank is (2048) 2K x 72 bits
Banks per Channel	—	42	—
Channel Capacity	without ECC with ECC	5.90625 Mb 5.25 Mb	—
Channels per eSRAM	—	8	—
eSRAM Capacity	without ECC with ECC	47.25 Mb 42 Mb	—
Interface Data Width	without ECC with ECC	x72	Maximum width
Read Latency [8]	Normal Low Power	10 +2 [9] 11 + 2 [9]	These latencies are fixed, whether ECC is enabled or not.
Write Latency	—	0 +1 [10]	There is a zero cycle latency for write commands issued to the eSRAM.
Power (per eSRAM system)	Industrial Extended	1.15 W - 1.5 W 2.28 W - 3.31 W	Low Power mode to Normal mode.

4.2.2.2. eSRAM Usage Model

The eSRAM configuration is deemed static after FPGA configuration. You cannot reconfigure the eSRAM after it enters user mode.

All 8 memory channels have an interface to a shared set of 3 fabric sectors. The fitter chooses which sector interfaces with core logic because not all the sectors are available for each eSRAM.

[6] The input clock source for eSRAM must not exceed 20 ps peak-to-peak, or 1.42 ps RMS at 1e-12 BER, 1.22 ps at 1e-16 BER.

[7] In Speed Grade 3 devices, the following clock frequency range is not supported:
- 466.51 MHz - 499.99 MHz
- 233.26 MHz - 249.99 MHz

[8] Read latency is measured from a read command being presented to the interface to valid read data being returned.

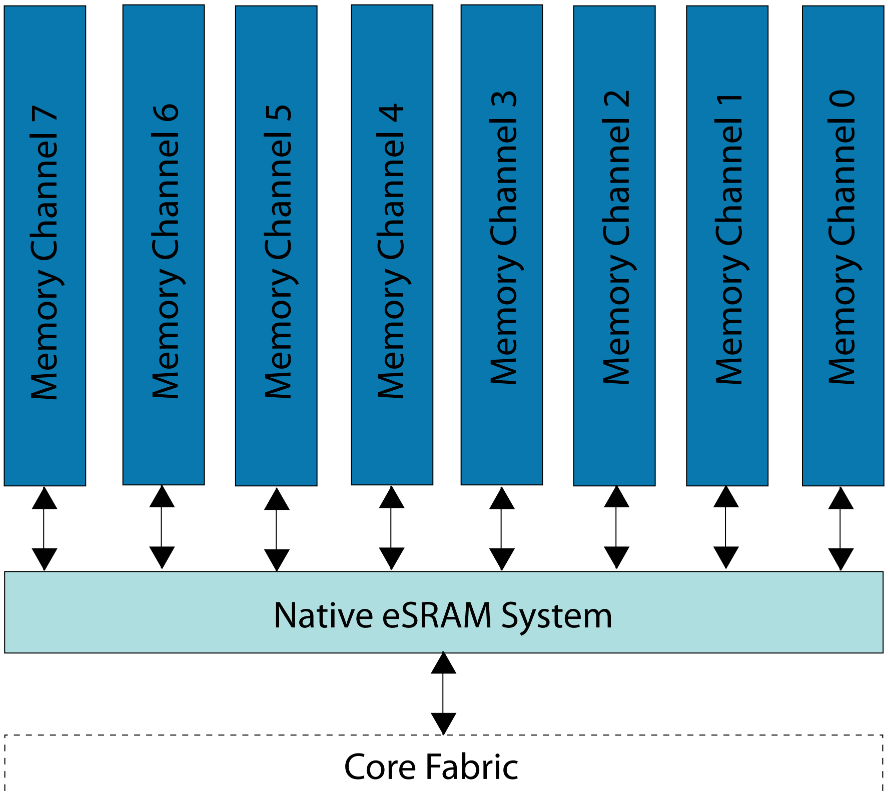
[9] +2 on read latency is added due to registers interfacing with eSRAM required to meet routing and timing requirement.

[10] +1 on read latency is added due to registers interfacing with eSRAM required to meet routing and timing requirement.

The reference clock (`refclk`) is only support LVDS standard. When setting an instance assignment, use the correct standard for `refclk`. An instance assignment must be set to use the correct standard for `refclk`:

```
set_instance_assignment -name IO_STANDARD LVDS -to refclk
```

Figure 28. eSRAM Interface With Core Logic



There is a maximum of 17 address bits available. Address bits [10:0] are the 11 bits used to target the 2K entries in a bank. Address bits [16:11] are the 6 bits used to target a certain bank in a channel. Because there are only 42 banks in a channel, the threshold address you can target is [16:11] = 6'b101001 (41st bank relative to the 0th Bank).

Note: eSRAM bits cannot be reset while in user mode and hence do not have a reset requirement.

Each of the 8 memory channels that make up an eSRAM can power down unused banks. You are responsible for selecting the desired capacity in the eSRAM Intel FPGA IP core as the unused banks are powered down by default.

4.2.3. eSRAM Intel FPGA IP Parameters

The parameters allow you to select the channels that you want to implement.

Table 34. eSRAM Intel FPGA IP Parameter Editor: General Tab

Parameter	Legal Values	Description
Interface		
Interface • Enable Channel 0 • Enable Channel 1 • Enable Channel 2 • Enable Channel 3 • Enable Channel 4 • Enable Channel 5 • Enable Channel 6 • Enable Channel 7	On/Off	Specifies the channel to be enabled for eSRAM. There are 8 channels per eSRAM. • **Enable Channel 0**—This option enables Channel 0 for eSRAM. • **Enable Channel 1**—This option enables Channel 1 for eSRAM. • **Enable Channel 2**—This option enables Channel 2 for eSRAM. • **Enable Channel 3**—This option enables Channel 3 for eSRAM. • **Enable Channel 4**—This option enables Channel 4 for eSRAM. • **Enable Channel 5**—This option enables Channel 5 for eSRAM. • **Enable Channel 6**—This option enables Channel 6 for eSRAM. • **Enable Channel 7**—This option enables Channel 7 for eSRAM.
PLL		
PLL Reference Clock Frequency	—	Specifies the PLL reference clock frequency to the eSRAM PLL. The valid ranges is 10 - 325 MHz for any device's speed grade.
PLL Desired Clock Frequency	—	Specifies the PLL desired output clock frequency which is the frequency to the eSRAM. The valid ranges is 200 - 750 MHz depending on the speed grade of your device.

Table 35. eSRAM Intel FPGA IP Core Parameter Editor: Channel Tab

Parameter	Legal Values	Description
Channel Width and Depth		
How wide should the data bus be?	—	Specifies the width of the data bus. • Normal mode: 1 to 72 bits • ECC enable only: 1 to 64 bits • Both ECC and ECC Encoder and Decoder Bypass enabled: 72 bits only
How many words of memory?	—	Specifies how many memory banks to use out of the possible 42 banks available per eSRAM channel. Banks are specified in increments of 2048 words, where each 2048 words equals one bank. The number of banks specified determines the address width available to the user. Banks that are not used are powered off and cannot be activated after parameterization. *Note:* If you attempt to address a bank that has not been enabled, any resulting data will be random and without value.

continued...

Parameter	Legal Values	Description
Channel Features		
Enable ECC Encoder and Decoder	On/Off	Enables the ECC encoder and decoder, which assists in maintaining the integrity of data written to and read from the eSRAM. *Note:* When you enable the ECC encoder and decoder, the maximum data bus width decreases from 72 bits to 64 bits. The 8 bit difference is used in the parity calculations required by the ECC encoder and decoder.
Enable Dynamic ECC Encoder and Decoder Bypass	On/Off	Enables users to dynamically bypass the ECC encoder and/or decoder, by asserting `eccencbypass` or `eccdecbypass`. This feature is useful for debugging purposes.
Enable Write Forwarding	On/Off	Enables write forwarding, which ensures data coherency when writing to and reading from the same address in the eSRAM. Write forwarding takes the data present on the write port and forwards it to the read port as read data. Write-forwarded read data requires the same duration of time as a regular read. Read logic does not use data stored in the targeted address, but the data is still written to the address.
Enable Low Power Mode	On/Off	Enables Low Power mode, which reduces power consumption by placing all eSRAM memory banks into a state of light sleep. When a bank is targeted for access, it is awakened one cycle prior to the access. The bank returns to a state of light sleep after the access is completed. Low Power mode does not alter the content of a memory bank. One drawback of Low Power mode is that it increases read latency from 10+2 to 11+2.

4.2.4. eSRAM Intel FPGA IP Interface Signals

The following table lists the input and output signals of the eSRAM Intel FPGA IP interface.

Table 36. eSRAM Intel FPGA IP Input and Output Signals

Signal	Direction	Width	Description
refclk	Input	1	Provide a PLL reference clock. This clock must be stable and free-running at device power-up for a successful configuration.
esram2f_clk	Output	1	Core clock provided by the eSRAM to the fabric. Use this clock to drive core logics that are interfacing with the eSRAM. Otherwise, proper cross-clock domain circuitry is expected.
c<*channel_number*>_data_0	Input	1-72	• 72 for clear channel data, or • 64 when ECC is enabled, or • 72 when ECC Bypass is enabled, the MSB (most significant bit) of data (data[71:64]) represents the parity bits.
c<*channel_number*>_wraddress_0	Input	Range from 17–11	Write address of the memory. Dependent on how many banks are enabled in the channel. *Note:* Writing to an invalid address does nothing, because the targeted bank is not powered.
c<*channel_number*>_wren_n_0	Input	1	Active low write enable input for the wraddress port.
c<*channel_number*>_rdaddress_0	Input	Range from 17–11	Read address of the memory. Dependent on how many banks are enabled in the channel. *Note:* If you attempt to read from an invalid address, the data returned is random and of no value.
c<*channel_number*>_rden_n_0	Input	1	Active low read enable input for the rdaddress port.
c<*channel_number*>_q_0	Output	72 or 64	• 72 for clear channel data, or • 64 when ECC is enabled, or • 72 when ECC Bypass is enabled, the MSB of output (q[71:64]) represents the parity bits.
ECC Enabled			
c<*channel_number*>_error_detect_0	Output	1	Asserts when an ECC error occurred on the read data retrieved from the eSRAM.
c<*channel_number*>_error_correct_0	Output	1	Asserts when an ECC error is successfully corrected. The memory content is not updated with the corrected data.
Dynamic ECC Bypass Enabled			
c<*channel_number*>_eccencbypass_0	Input	1	Dynamically bypass the ECC Encoder. When active, this port allows user to inject parity bits through 8-bits MSB from data port (c<channel_number>_data_0[71:64]). When inactive, parity bits will be generated using internal ECC Encoder.

continued...

Signal	Direction	Width	Description
			This port can only be used when `c<channel_number>_ecc_byp_enable` parameter is set to "TRUE".
`c<channel_number>_eccdecbypass_0`	Input	1	Dynamically bypass the ECC Decoder. 8-bits MSB from output port (`c<channel_number>_q_0[73:64]`) represents the parity bits. Parity bits are not checked and the `c<channel_number>_error_detect_0` and `c<channel_number>_error_correct_0` signals should not assert. This port can only be used when `c<channel_number>_ecc_byp_enable` parameter is set to "TRUE".
Additional Options			
`c<channel_number>_sd_n_0`	Input	1	Active low signal that dynamically shuts down channels. This signal shuts down power to periphery (active low) and memory core of the banks within the channel, with no memory data retention. In addition to the channels that are statically shut down when choosing the number of channels to use in an eSRAM system, you can also dynamically shut down channels at run time. *Note:* Memory contents are not retained when a channel is shut down.
`iopll_lock2core`	Output	1	eSRAM IOPLL lock status. • High—Locked • Low—Unlocked or lock loss.

4.2.5. eSRAM Intel FPGA IP Simulation Walk Through

The IOPLL is included in the eSRAM Intel FPGA IP core to drive its clock domains for operation. The testbench should wait for the IOPLL to be locked before starting any simulation to ensure the clock entering the eSRAM is always stable. During the waiting period for the IOPLL to lock, the eSRAM will not function properly due to unstable clock frequency. In hardware, the testbench does not need to check the IOPLL lock signal because the IOPLL lock signal is asserted at the configuration stage, which is handled by firmware. The wait for the IOPLL lock is only needed to perform in software simulation.

You can check the LOCK signal from the output port `iopll_lock2core` in the eSRAM IP design. Simulation can only start after the `iopll_lock2core` signal goes from LOW to HIGH.

Note: Before starting the simulation, you must provide sufficient delay (for example, 10 us) for the clock to be stable enough after the eSRAM's IOPLL is locked (`iopll_lock2core` signal goes from LOW to HIGH).

4.2.6. eSRAM Timing Diagrams

The timing diagrams show the signal behavior of the eSRAM in normal and low power modes.

Figure 29. eSRAM Timing Diagram in Normal Mode

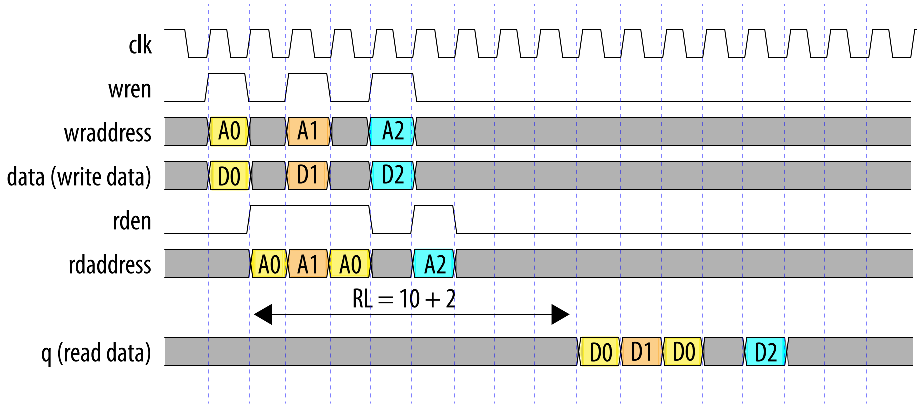
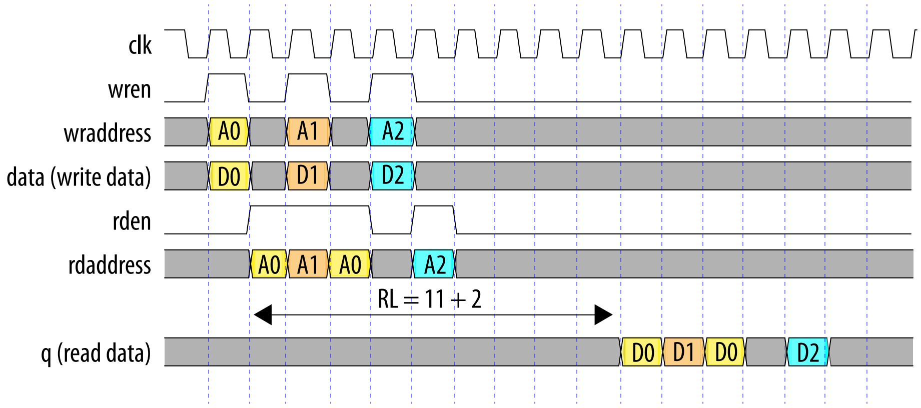


Figure 30. eSRAM Timing Diagram in Low-Power Mode



4.3. FIFO Intel FPGA IP

Intel provides FIFO Intel FPGA IP core through the parameterizable single-clock FIFO (SCFIFO) and dual-clock FIFO (DCFIFO) functions.

The FIFO functions are mostly applied in data buffering applications that comply with the first-in-first-out data flow in synchronous or asynchronous clock domains.

The specific names of the FIFO functions are as follows:

- SCFIFO: single-clock FIFO
- DCFIFO: dual-clock FIFO (supports same port widths for input and output data)
- DCFIFO_MIXED_WIDTHS: dual-clock FIFO (supports different port widths for input and output data)

Note: The term “DCFIFO” refers to both the DCFIFO and DCFIFO_MIXED_WIDTHS IP cores, unless specified.

Related Information

FIFO Intel FPGA IP Core User Guide

4.3.1. Release Information for FIFO Intel FPGA IP

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP version (X.Y.Z) number may change from one Intel Quartus Prime software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 37. FIFO Intel FPGA IP Current Release Information

Item	Description
IP Version	19.1
Intel Quartus Prime Version	19.1
Release Date	2019.04.13

4.3.2. Configuration Methods

Table 38. Configuration Methods

You can configure and build the FIFO Intel FPGA IP core with methods shown in the following table.

Method	Description
Using the FIFO parameter editor.	Intel recommends using this method to build your FIFO Intel FPGA IP core. It is an efficient way to configure and build the FIFO Intel FPGA IP core. The FIFO parameter editor provides options that you can easily use to configure the FIFO Intel FPGA IP core. You can access the FIFO Intel FPGA IP core parameter editor in **Basic Functions ➤ On Chip Memory ➤ FIFO** of the IP catalog.[16]
Manually instantiating the FIFO Intel FPGA IP core.	Use this method only if you are an expert user. This method requires that you know the detailed specifications of the IP core. You must ensure that the input and output ports used, and the parameter values assigned are valid for the FIFO Intel FPGA IP core you instantiate for your target device.

4.3.3. Specifications

[16] Do not use dcfifo or scfifo as the entity name for your FIFO Platform Designer system.

4.3.3.1. Verilog HDL Prototype

You can locate the Verilog HDL prototype in the Verilog Design File (`.v`) `altera_mf.v` in the `<Intel Quartus Prime installation directory>\eda\sim_lib` directory.

4.3.3.2. VHDL Component Declaration

The VHDL component declaration is located in the `<Intel Quartus Prime installation directory>\libraries\vhdl\altera_mf\altera_mf_components.vhd`

4.3.3.3. VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

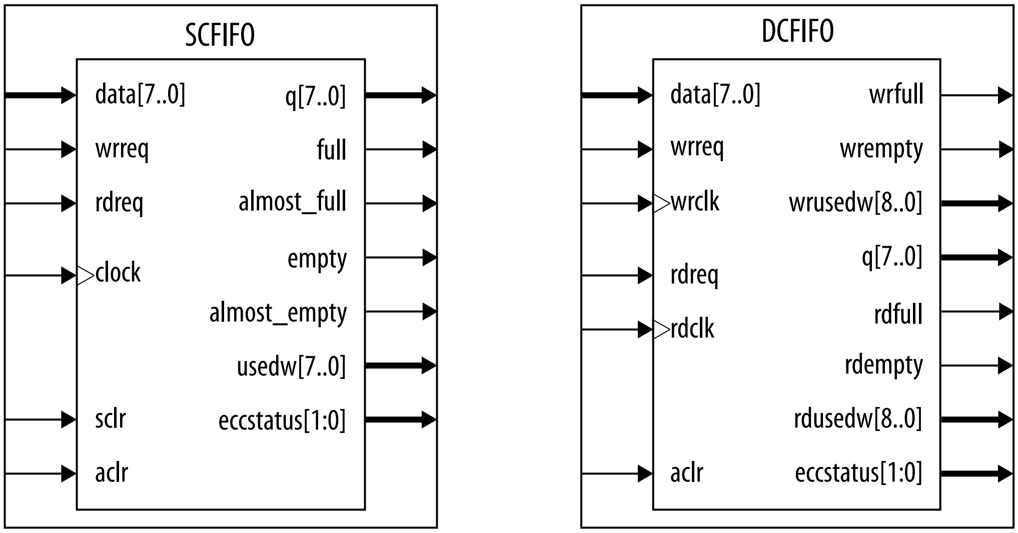
```
LIBRARY altera_mf;

USE altera_mf.altera_mf_components.all;
```

4.3.3.4. FIFO Signals

This section provides diagrams of the SCFIFO and DCFIFO blocks of the FIFO Intel FPGA IP core to help in visualizing their input and output ports. This section also describes each port in detail to help in understanding their usages, functionality, or any restrictions. For better illustrations, some descriptions might refer you to a specific section in this user guide.

Figure 31. SCFIFO and DCFIFO Input and Output Signals



For the SCFIFO block, the read and write signals are synchronized to the same clock; for the DCFIFO block, the read and write signals are synchronized to the `rdclk` and `wrclk` clocks respectively. The prefixes `wr` and `rd` represent the signals that are synchronized by the `wrclk` and `rdclk` clocks respectively.

Table 39. Input and Output Ports Description

This table lists the signals of the FIFO Intel FPGA IP core. The term “series” refers to all the device families of a particular device.

Port	Type	Required	Description
clock [17]	Input	Yes	Positive-edge-triggered clock.
wrclk [18]	Input	Yes	Positive-edge-triggered clock. Use to synchronize the following ports: • data • wrreq • wrfull • wrempty • wrusedw
rdclk [18]	Input	Yes	Positive-edge-triggered clock. Use to synchronize the following ports: • q • rdreq • rdfull • rdempty • rdusedw
data [19]	Input	Yes	Holds the data to be written in the FIFO Intel FPGA IP core when the wrreq signal is asserted. If you manually instantiate the FIFO Intel FPGA IP core, ensure the port width is equal to the lpm_width parameter.
wrreq [19]	Input	Yes	Assert this signal to request for a write operation. Ensure that the following conditions are met: • Do not assert the wrreq signal when the full (for SCFIFO) or wrfull (for DCFIFO) port is high. Enable the overflow protection circuitry or set the overflow_checking parameter to **ON** so that the FIFO Intel FPGA IP core can automatically disable the wrreq signal when it is full. • The wrreq signal must meet the functional timing requirement based on the full or wrfull signal. • Do not assert the wrreq signal during the deassertion of the aclr signal. Violating this requirement creates a race condition between the falling edge of the aclr signal and the rising edge of the write clock if the wrreq port is set to high. For both the DCFIFO functions that target Stratix series, you have the option to automatically add a circuit to synchronize the aclr signal with the wrclk clock, or set the write_aclr_synch parameter to **ON**. Use this option to ensure that the restriction is obeyed.
rdreq [19]	Input	Yes	Assert this signal to request for a read operation. The rdreq signal acts differently in normal mode and show-ahead mode.

continued...

[17] Only applicable for the SCFIFO function.

[18] Applicable for both of the DCFIFO functions.

[19] Applicable for the SCFIFO, DCFIFO, and DCFIFO_MIXED_WIDTH functions.

Port	Type	Required	Description
			Ensure that the following conditions are met: • Do not assert the rdreq signal when the empty (for SCFIFO) or rdempty (for DCFIFO) port is high. Enable the underflow protection circuitry or set the underflow_checking parameter to **ON** so that the FIFO Intel FPGA IP core can automatically disable the rdreq signal when it is empty. • The rdreq signal must meet the functional timing requirement based on the empty or rdempty signal.
sclr [17] aclr [19]	Input	No	Assert this signal to clear all the output status ports, but the effect on the q output may vary for different FIFO configurations. There are no minimum number of clock cycles for aclr signals that must remain active.
q [19]	Output	Yes	Shows the data read from the read request operation. For the SCFIFO function and DCFIFO function, the width of the q port must be equal to the width of the data port. If you manually instantiate the FIFO functions, ensure that the port width is equal to the lpm_width parameter. For the DCFIFO_MIXED_WIDTHS function, the width of the q port can be different from the width of the data port. If you manually instantiate the FIFO function, ensure that the width of the q port is equal to the lpm_width_r parameter. The FIFO function supports a wide write port with a narrow read port, and vice versa. However, the width ratio is restricted by the type of RAM block, and in general, are in the power of 2.
full [17] wrfull [18] rdfull [18]	Output	No	When asserted, the FIFO Intel FPGA IP core is considered full. Do not perform write request operation when the FIFO Intel FPGA IP core is full. In general, the rdfull signal is a delayed version of the wrfull signal. The rdfull signal functions as a combinational output instead of a derived version of the wrfull signal. Therefore, you must always refer to the wrfull port to ensure whether or not a valid write request operation can be performed, regardless of the target device.
empty [17] wrempty [18] rdempty [18]	Output	No	When asserted, the FIFO Intel FPGA IP core is considered empty. Do not perform read request operation when the FIFO Intel FPGA IP core is empty. In general, the wrempty signal is a delayed version of the rdempty signal. The wrempty signal functions as a combinational output instead of a derived version of the rdempty signal. Therefore, you must always refer to the rdempty port to ensure whether or not a valid read request operation can be performed, regardless of the target device.
almost_full [17]	Output	No	Asserted when the usedw signal is greater than or equal to the almost_full_value parameter. It is used as an early indication of the full signal.
almost_empty [17]	Output	No	Asserted when the usedw signal is less than the almost_empty_value parameter. It is used as an early indication of the empty signal. [20]
usedw [17] wrusedw [18]	Output	No	Show the number of words stored in the FIFO.

continued...

[20] Under certain condition, the SCFIFO asserts the empty signal without ever asserting the almost_empty signal. Refer to SCFIFO ALMOST_EMPTY Functional Timing on page 77 for more details.

Port	Type	Required	Description
rdusedw [18]			Ensure that the port width is equal to the lpm_widthu parameter if you manually instantiate the SCFIFO function or the DCFIFO function. For the DCFIFO_MIXED_WIDTH function, the width of the wrusedw and rdusedw ports must be equal to the LPM_WIDTHU and lpm_widthu_r parameters respectively. The FIFO Intel FPGA IP core shows full even before the number of words stored reaches its maximum value. Therefore, you must always refer to the full or wrfull port for valid write request operation, and the empty or rdempty port for valid read request operation regardless of the target device. *Note:* Stored data may not be available for reading. Refer to FIFO Output Status Flag and Latency on page 78 for "wrreq to empty" and "rdreq to empty" latency to ensure that the data is ready before reading the FIFO.
eccstatus [21]	Output	No	A 2-bit wide error correction status port. Indicate whether the data that is read from the memory has an error in single-bit with correction, fatal error with no correction, or no error bit occurs. • 00: No error • 01: Illegal • 10: A correctable error occurred and the error has been corrected at the outputs; however, the memory array has not been updated. • 11:An uncorrectable error occurred and uncorrectable data appears at the output.

The DCFIFO functionrdempty output may momentarily glitch when the aclr input is asserted. To prevent an external register from capturing this glitch incorrectly, ensure that one of the following is true:

- The external register must use the same reset which is connected to the aclr input of the DCFIFO function, or
- The reset connected to the aclr input of the DCFIFO function must be asserted synchronous to the clock which drives the external register.

The output latency information of the FIFO Intel FPGA IP core is important, especially for the q output port, because there is no output flag to indicate when the output is valid to be sampled.

4.3.3.5. FIFO Parameter Settings

Table 40. FIFO Parameters

Parameter	Type	Required	Description
lpm_width	Integer	Yes	Specifies the width of the data and q ports for the SCFIFO function and DCFIFO function. For the DCFIFO_MIXED_WIDTHS function, this parameter specifies only the width of the data port.
lpm_width_r [22]	Integer	Yes	Specifies the width of the q port for the DCFIFO_MIXED_WIDTHS function.

continued...

[21] Not applicable for the DCFIFO_MIXED_WIDTHS function.

[22] Only applicable for the DCFIFO_MIXED_WIDTHS function.

Parameter	Type	Required	Description
`lpm_widthu`	Integer	Yes	Specifies the width of the `usedw` port for the SCFIFO function, or the width of the `rdusedw` and `wrusedw` ports for the DCFIFO function. For the DCFIFO_MIXED_WIDTHS function, it only represents the width of the `wrusedw` port.
`lpm_widthu_r` [22]	Integer	Yes	Specifies the width of the `rdusedw` port for the DCFIFO_MIXED_WIDTHS function.
`lpm_numwords`	Integer	Yes	Specifies the depths of the FIFO you require. The value must be at least **4**. The value assigned must comply to the following equation: 2^LPM_WIDTHU
`lpm_showahead`	String	Yes	Specifies whether the FIFO is in normal mode (`OFF`) or show-ahead mode (`ON`). For more details, refer to SCFIFO and DCFIFO Look-Ahead Mode section. If you set the parameter to **ON**, you may reduce performance.
`lpm_type`	String	No	Identifies the library of parameterized modules (LPM) entity name. The values are **SCFIFO** and **DCFIFO**.
`lpm_hint`	String	No	This is a legacy parameter that is used to set the following: 1. Maximum depth. You can set the maximum depth desired with "MAXIMUM_DEPTH=`<depth>`". Allow the Intel Quartus Prime software to automatically choose the maximum depth of the RAM used in SCFIFO by ignoring this parameter. 2. RAM block type. You can select the RAM block type with "RAM_BLOCK_TYPE=<M20K|MLAB|AUTO>". The default value is **AUTO**. 3. Disable Embedded Timing Constraint. This is a compulsory parameter to be set for DCFIFO, which disables the legacy `set_false_path` that resides in FIFO Intel FPGA IP. Add "DISABLE_EMBEDDED_TIMING_CONSTRAINT=TRUE" to `lpm_hint` for proper timing analysis on the synchronizer path within DCFIFO IP. Options should be concatenated with a comma in between and enclosed with double quotations. For example: `lpm_hint = "MAXIMUM_DEPTH=512, RAM_BLOCK_TYPE=M20K"`
`overflow_checking`	String	No	Specifies whether or not to enable the protection circuitry for overflow checking that disables the `wrreq` port when the FIFO Intel FPGA IP core is full. The values are **ON** or **OFF**. If omitted, the default is **ON**.
`underflow_checking`	String	No	Specifies whether or not to enable the protection circuitry for underflow checking that disables the `rdreq` port when the FIFO Intel FPGA IP core is empty. The values are **ON** or **OFF**. If omitted, the default is **ON**. Note that reading from an empty SCFIFO gives unpredictable results.
`enable_ecc` [23]	String	No	Specifies whether to enable the error checking and correcting (ECC) feature that corrects single bit errors, double adjacent bit errors, and detects triple adjacent bit errors at the output of the memory. This option is only available using M20K memory block type.

continued...

[23] Not applicable for the DCFIFO_MIXED_WIDTHS function.

Parameter	Type	Required	Description
			The ECC is disabled by default.
`delay_wrusedw` [24]	String	No	Specify the number of register stages that you want to internally add to the `rdusedw` or `wrusedw` port using the respective parameter. The default value of 1 adds a single register stage to the output to improve its performance. Increasing the value of the parameter does not increase the maximum system speed. It only adds additional latency to the respective output port.
`add_usedw_msb_bit` [24]	String	No	Increases the width of the `rdusedw` and `wrusedw` ports by one bit. By increasing the width, it prevents the FIFO Intel FPGA IP core from rolling over to zero when it is full. The values are ON or OFF. If omitted, the default value is OFF.
`rdsync_delaypipe` [24] `wrsync_delaypipe` [24]	Integer	No	Specify the number of synchronization stages in the cross clock domain. The value of the `rdsync_delaypipe` parameter relates the synchronization stages from the write control logic to the read control logic; the `wrsync_delaypipe` parameter relates the synchronization stages from the read control logic to the write control logic. Use these parameters to set the number of synchronization stages if the clocks are not synchronized, and set the `clocks_are_synchronized` parameter to **FALSE**. The actual synchronization stage implemented relates variously to the parameter value assigned, depends on the target device. The values of these parameters are internally reduced by two. Thus, the default value of 3 for these parameters corresponds to a single synchronization stage; a value of 4 results in two synchronization stages, and so on. Choose at least 4 (two synchronization stages) for metastability protection.
`use_eab`	String	No	Specifies whether or not the FIFO Intel FPGA IP core is constructed using the RAM blocks. The values are **ON** or **OFF**. Setting this parameter value to **OFF** yields the FIFO Intel FPGA IP core implemented in ALMs regardless of the type of the TriMatrix memory block type assigned to the `ram_block_type` parameter. This parameter is enabled by default. FIFO will be implemented using RAM blocks specified in `ram_block_type`.
`write_aclr_synch` [24]	String	No	Specifies whether or not to add a circuit that causes the `aclr` port to be internally synchronized by the `wrclk` clock. Adding the circuit prevents the race condition between the `wrreq` and `aclr` ports that could corrupt the FIFO Intel FPGA IP core. The values are **ON** or **OFF**. If omitted, the default value is **OFF**.
`read_aclr_synch` [24]	String	No	Specifies whether or not to add a circuit that causes the `aclr` port to be internally synchronized by the `rdclk` clock. Adding the circuit prevents the race condition between the `rdreq` and `aclr` ports that could corrupt the FIFO Intel FPGA IP core.

continued...

[24] Only applicable for the DCFIFO function.

Parameter	Type	Required	Description
			The values are **ON** or **OFF**. If omitted, the default value is **OFF**.
clocks_are_synchronized [24]	String	No	Specifies whether or not the write and read clocks are synchronized which in turn determines the number of internal synchronization stages added for stable operation of the FIFO. The values are **TRUE** and **FALSE**. If omitted, the default value is **FALSE**. You must only set the parameter to **TRUE** if the write clock and the read clock are always synchronized and they are multiples of each other. Otherwise, set this to **FALSE** to avoid metastability problems. If the clocks are not synchronized, set the parameter to **FALSE**, and use the rdsync_delaypipe and wrsync_delaypipe parameters to determine the number of synchronization stages required.
ram_block_type	String	No	Specifies the target device's Trimatrix Memory Block to be used. To get the proper implementation based on the RAM configuration that you set, allow the Intel Quartus Prime software to automatically choose the memory type by ignoring this parameter and set the use_eab parameter to **ON**. This gives the compiler the flexibility to place the memory function in any available memory resource based on the FIFO depth required. Types of RAM block type available; Auto (default), MLAB, M20K and M144K.
add_ram_output_register	String	No	Specifies whether to register the q output. The values are **ON** and **OFF**. If omitted, the default value is **OFF**.
almost_full_value [25]	Integer	No	Sets the threshold value for the almost_full port. When the number of words stored in the FIFO Intel FPGA IP core is greater than or equal to this value, the almost_full port is asserted.
almost_empty_value [25]	Integer	No	Sets the threshold value for the almost_empty port. When the number of words stored in the FIFO Intel FPGA IP core is less than this value, the almost_empty port is asserted.
allow_wrcycle_when_full [25]	String	No	Allows you to combine read and write cycles to an already full SCFIFO, so that it remains full. The values are **ON** and **OFF**. If omitted, the default is **OFF**. Use only this parameter when the OVERFLOW_CHECKING parameter is set to **ON**.
intended_device_family	String	No	Specifies the intended device that matches the device set in your Intel Quartus Prime project. Use only this parameter for functional simulation.

4.3.4. FIFO Functional Timing Requirements

The wrreq signal is ignored (when FIFO is full) if you enable the overflow protection circuitry in the FIFO Intel FPGA IP parameter editor, or set the OVERFLOW_CHECKING parameter to ON. The rdreq signal is ignored (when FIFO is empty) if you enable the underflow protection circuitry in the FIFO Intel FPGA IP core interface, or set the UNDERFLOW_CHECKING parameter to ON.

[25] Only applicable for the SCFIFO function.

If the protection circuitry is not enabled, you must meet the following functional timing requirements:

Table 41. Functional Timing Requirements

DCFIFO	SCFIFO
Deassert the `wrreq` signal in the same clock cycle when the `wrfull` signal is asserted.	Deassert the `wrreq` signal in the same clock cycle when the `full` signal is asserted.
Deassert the `rdreq` signal in the same clock cycle when the `rdempty` signal is asserted. You must observe these requirements regardless of expected behavior based on `wrclk` and `rdclk` frequencies.	Deassert the `rdreq` signal in the same clock cycle when the `empty` signal is asserted.

Figure 32. Functional Timing for the `wrreq` Signal and the `wrfull` Signal

This figure shows the behavior for the `wrreq` and the `wrfull` signals.

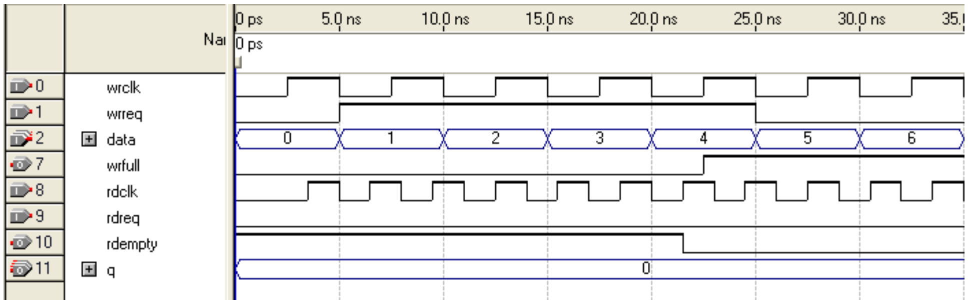
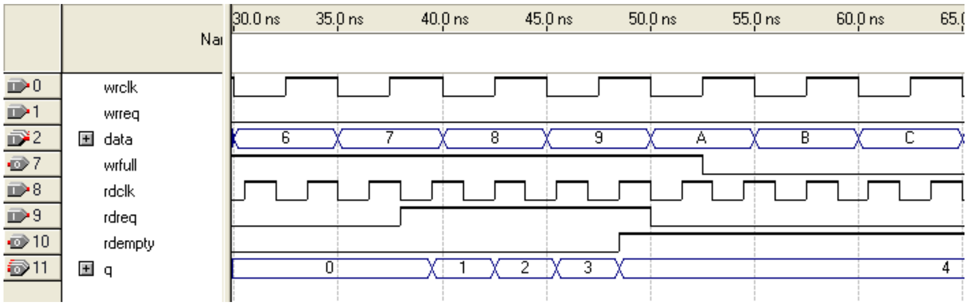


Figure 33. Functional Timing for the `rdreq` Signal and the `rdempty` Signal

This shows the behavior for the `rdreq` the `rdempty` signals.

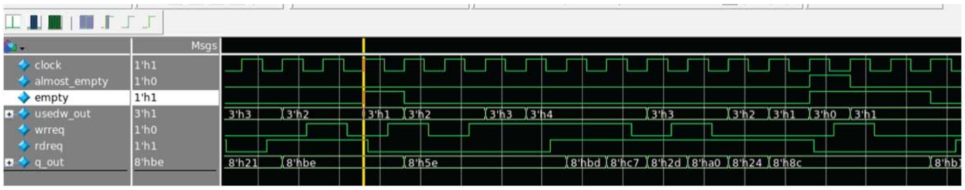


The required functional timing for the DCFIFO as described previously is also applied to the SCFIFO. The difference between the two modes is that for the SCFIFO, the `wrreq` signal must meet the functional timing requirement based on the `full` signal and the `rdreq` signal must meet the functional timing requirement based on the `empty` signal.

4.3.5. SCFIFO ALMOST_EMPTY Functional Timing

In SCFIFO, the `almost_empty` is asserted only when the `usedw` is less than the `almost_empty_value` that you set. The `almost_empty` signal does not consider the data readiness at the output. When the `almost_empty_value` is set too low, it is possible to observe that SCFIFO asserts the `empty` signal without asserting the `almost_emtpy` signal.

Figure 34. Example of empty Signal Assertion without Asserting almost_empty Signal



In this example, the almost_empty_value is 1 which means the almost_empty will assert when usedw is 0. There are three words in the FIFO before the read request is received. After the first read, the wrreq asserts and the rdreq signal remains high. The usedw remains at 2. In the next cycle, the wrreq de-asserts but there is another rdreq going on. The usedw decrease to 1 and the almost_emtpy signal remains low. However, the write data has not been written into the FIFO due to the write latency. The empty signal asserts to indicate the FIFO is empty.

4.3.6. FIFO Output Status Flag and Latency

The main concern in most FIFO design is the output latency of the read and write status signals.

Table 42. Output Latency of the Status Flags for SCFIFO

This table shows the output latency of the write signal (wrreq) and read signal (rdreq) for the SCFIFO according to the different output modes and optimization options.

Output Mode	Optimization Option [26]	Output Latency (in number of clock cycles)
Normal [27]	Speed	wrreq / rdreq to full: 1
		wrreq to empty: 2
		rdreq to empty: 1
		wrreq / rdreq to usedw[]: 1
		rdreq to q[]: 1
	Area	wrreq / rdreq to full: 1
		wrreq / rdreq to empty : 1
		wrreq / rdreq to usedw[] : 1
		rdreq to q[]: 1
Show-ahead [27]	Speed	wrreq / rdreq to full: 1
		wrreq to empty: 3
		rdreq to empty: 1
		wrreq / rdreq to usedw[]: 1

continued...

[26] Speed optimization is equivalent to setting the ADD_RAM_OUTPUT_REGISTER parameter to ON. Setting the parameter to OFF is equivalent to area optimization.

[27] Normal output mode is equivalent to setting the LPM_SHOWAHEAD parameter to OFF. For Show-ahead mode, the parameter is set to ON.

Output Mode	Optimization Option [26]	Output Latency (in number of clock cycles)
		`wrreq` to `q[]`: 3
		`rdreq` to `q[]`: 1
	Area	`wrreq` / `rdreq` to `full`: 1
		`wrreq` to `empty`: 2
		`rdreq` to `empty`: 1
		`wrreq` / `rdreq` to `usedw[]`: 1
		`wrreq` to `q[]`: 2
		`rdreq` to `q[]`: 1

Table 43. ALM Implemented RAM Mode for SCFIFO and DCFIFO

Output Mode	Optimization Option [28]	Output Latency (in number of clock cycles)
Normal [29]	Speed	`wrreq` / `rdreq` to `full`: 1
		`wrreq` to `empty`: 1
		`rdreq` to `empty`: 1
		`wrreq` / `rdreq` to `usedw[]`: 1
		`rdreq` to `q[]`: 1
	Area	`wrreq` / `rdreq` to `full`: 1
		`wrreq` / `rdreq` to `empty` : 1
		`wrreq` / `rdreq` to `usedw[]` : 1
		`rdreq` to `q[]`: 1
Show-ahead [29]	Speed	`wrreq` / `rdreq` to `full`: 1
		`wrreq` to `empty`: 1
		`rdreq` to `empty`: 1
		`wrreq` / `rdreq` to `usedw[]`: 1
		`wrreq` to `q[]`: 1
		`rdreq` to `q[]`: 1
	Area	`wrreq` / `rdreq` to `full`: 1
		`wrreq` to `empty`: 1
		`rdreq` to `empty`: 1

continued...

[26] Speed optimization is equivalent to setting the `ADD_RAM_OUTPUT_REGISTER` parameter to `ON`. Setting the parameter to `OFF` is equivalent to area optimization.

[28] Speed optimization is equivalent to setting the `ADD_RAM_OUTPUT_REGISTER` parameter to `ON`. Setting the parameter to `OFF` is equivalent to area optimization.

[29] Normal output mode is equivalent to setting the `LPM_SHOWAHEAD` parameter to `OFF`. For Show-ahead mode, the parameter is set to `ON`.

Output Mode	Optimization Option [28]	Output Latency (in number of clock cycles)
		wrreq / rdreq to usedw[]: 1
		wrreq to q[]: 1
		rdreq to q[]: 1

Table 44. Output Latency of the Status Flag for the DCFIFO

This table shows the output latency of the write signal (wrreq) and read signal (rdreq) for the DCFIFO.

Output Latency (in number of clock cycles)
wrreq to wrfull: 1 wrclk
wrreq to rdfull: 2 wrclk cycles + following n rdclk [30]
wrreq to wrempty: 1 wrclk
wrreq to rdempty: 2 wrclk [31] + following n rdclk [31]
wrreq to wrusedw[]: 2 wrclk
wrreq to rdusedw[]: 2 wrclk + following n + 1 rdclk [31]
wrreq to q[]: 1 wrclk + following 1 rdclk [31]
rdreq to rdempty: 1 rdclk
rdreq to wrempty: 1 rdclk + following n wrclk [31]
rdreq to rfull: 1 rdclk
rdreq to wrfull: 1 rdclk + following n wrclk [31]
rdreq to rdusedw[]: 2 rdclk
rdreq to wrusedw[]: 1 rdclk + following n + 1 wrclk [31]
rdreq to q[]: 1 rdclk

4.3.7. FIFO Metastability Protection and Related Options

The FIFO Intel FPGA IP parameter editor provides the total latency, clock synchronization, metastability protection, area, and f_{MAX} options as a group setting for the DCFIFO.

[28] Speed optimization is equivalent to setting the ADD_RAM_OUTPUT_REGISTER parameter to ON. Setting the parameter to OFF is equivalent to area optimization.

[30] The number of n cycles for rdclk and wrclk is equivalent to the number of synchronization stages and are related to the WRSYNC_DELAYPIPE and RDSYNC_DELAYPIPE parameters. For more information about how the actual synchronization stage (n) is related to the parameters set for different target device, refer to FIFO Metastability Protection and Related Options on page 80.

[31] This is applied only to Show-ahead output modes. Show-ahead output mode is equivalent to setting the LPM_SHOWAHEAD parameter to ON.

Table 45. DCFIFO Group Setting for Latency and Related Options

This table shows the available group setting.

Group Setting	Comment
Lowest latency but requires synchronized clocks	This option uses one synchronization stage with no metastability protection. It uses the smallest size and provides good f_{MAX}. Select this option if the read and write clocks are related clocks.
Minimal setting for unsynchronized clocks	This option uses two synchronization stages with good metastability protection. It uses the medium size and provides good f_{MAX}.
Best metastability protection, best f_{max} and unsynchronized clocks	This option uses three or more synchronization stages with the best metastability protection. It uses the largest size but gives the best f_{MAX}.

The group setting for latency and related options is available through the FIFO Intel FPGA IP parameter editor. The setting mainly determines the number of synchronization stages, depending on the group setting you select. You can also set the number of synchronization stages you desire through the WRSYNC_DELAYPIPE and RDSYNC_DELAYPIPE parameters, but you must understand how the actual number of synchronization stages relates to the parameter values set in different target devices.

The **number of synchronization stages** set is related to the value of the WRSYNC_DELAYPIPE and RDSYNC_DELAYPIPE pipeline parameters. For some cases, these pipeline parameters are internally scaled down by two to reflect the actual synchronization stage.

The following equation shows the relationship between the actual synchronization stage and the pipeline parameters:

Actual synchronization stage = value of pipeline parameter - 2

Note: The values assigned to WRSYNC_DELAYPIPE and RDSYNC_DELAYPIPE parameters are internally reduced by 2 to represent the actual synchronization stage implemented. Thus, the default value 3 for these parameters corresponds to a single synchronization pipe stage; a value of 4 results in 2 synchronization stages, and so on. Choose 4 (2 synchronization stages) for metastability protection.

The Timing Analyzer includes the capability to estimate the robustness of asynchronous transfers in your design, and to generate a report that details the mean time between failures (MTBF) for all detected synchronization register chains. This report includes the MTBF analysis on the synchronization pipeline you applied between the asynchronous clock domains in your DCFIFO. You can then decide the number of synchronization stages to use in order to meet the range of the MTBF specification you require.

4.3.8. FIFO Synchronous Clear and Asynchronous Clear Effect

The FIFO Intel FPGA IP core supports the synchronous clear (sclr) and asynchronous clear (aclr) signals, depending on the FIFO modes.The effects of these signals are varied for different FIFO configurations. The SCFIFO supports both synchronous and asynchronous clear signals while the DCFIFO support asynchronous clear signal and asynchronous clear signal that synchronized with the write and read clocks.

Note: For Intel Stratix 10 devices, you must assert either aclr or sclr upon power-up to guarantee correct functionality.

Table 46. Synchronous Clear and Asynchronous Clear in the SCFIFO

Mode	Synchronous Clear (sclr) [32]	Asynchronous Clear (aclr)
Effects on status ports	Deasserts the full and almost_full signals.	
	Asserts the empty and almost_empty signals.	
	Resets the usedw flag.	
Commencement of effects upon assertion	At the rising edge of the clock.	Immediate (except for the q output)
Effects on the q output for normal output modes	The read pointer is reset and points to the first data location. If the q output is not registered, the output shows the first data word of the SCFIFO; otherwise, the q output remains at its previous value.	The q output remains at its previous value.
Effects on the q output for show-ahead output modes	The read pointer is reset and points to the first data location. If the q output is not registered, the output remains at its previous value for only one clock cycle and shows the first data word of the SCFIFO at the next rising clock edge. [33] Otherwise, the q output remains at its previous value.	If the q output is not registered, the output shows the first data word of the SCFIFO starting at the first rising clock edge. Otherwise, the q output remains its previous value.

[32] The read and write pointers reset to zero upon assertion of either the sclr or aclr signal.

[33] The first data word shown after the reset is not a valid Show-ahead data. It reflects the data where the read pointer is pointing to because the q output is not registered. To obtain a valid Show-ahead data, perform a valid write after the reset.

Table 47. Asynchronous Clear in DCFIFO

Mode	Asynchronous Clear (aclr)	aclr (synchronize with write clock) [34] [35]	aclr (synchronize with read clock) [36] [36]
Effects on status ports	Deasserts the wrfull signal.	The wrfull signal is asserted while the write domain is clearing which nominally takes three cycles of the write clock after the asynchronous release of the aclr input.	The rdempty signal is asserted while the read domain is clearing which nominally takes three cycles of the read clock after the asynchronous release of the aclr input.
	Deasserts the rdfull signal.		
	Asserts the wrempty and rdempty signals.		
	Resets the wrusedw and rdusedw flags.		
Commencement of effects upon assertion	Immediate.		
Effects on the q output for normal output modes [38]	The output remains unchanged if it is not registered. If the port is registered, it is cleared.		
Effects on the q output for show-ahead output modes	The output shows 'X' if it is not registered. If the port is registered, it is cleared.		

4.3.8.1. Recovery and Removal Timing Violation Warnings when Compiling a DCFIFO

During compilation of a design that contains a DCFIFO, the Intel Quartus Prime software may issue recovery and removal timing violation warnings.

[34] The wrreq signal must be low when the DCFIFO comes out of reset (the instant when the aclr signal is deasserted) at the rising edge of the write clock to avoid a race condition between write and reset. If this condition cannot be guaranteed in your design, the aclr signal needs to be synchronized with the write clock. This can be done by setting the **Add circuit to synchronize 'aclr' input with 'wrclk'** option from the FIFO parameter editor, or setting the WRITE_ACLR_SYNCH parameter to ON.

[35] Even though the aclr signal is synchronized with the write clock, asserting the aclr signal still affects all the status flags asynchronously.

[36] The rdreq signal must be low when the DCFIFO comes out of reset (the instant when the aclr signal is deasserted) at the rising edge of the read clock to avoid a race condition between read and reset. If this condition cannot be guaranteed in your design, the aclr signal needs to be synchronized with the read clock. This can be done by setting the **Add circuit to synchronize 'aclr' input with 'rdclk'** option from the FIFO parameter editor, or setting the READ_ACLR_SYNCH parameter to ON.

[37] Even though the aclr signal is synchronized with the read clock, asserting the aclr signal affects all the status flags asynchronously.

[38] For Stratix series, the DCFIFO only supports registered q output in Normal mode, and unregistered q output in Show-ahead mode.

You may safely ignore warnings that represent transfers from `aclr` to the read side clock domain. To ensure that the design meets timing, enable the ACLR synchronizer for both read and write domains.

To enable the ACLR synchronizer for both read and write domains, on the **DCFIFO 2** tab of the FIFO Intel FPGA IP core, turn on **Asynchronous clear**, **Add circuit to synchronize 'aclr' input with 'wrclk'**, and **Add circuit to synchronize 'aclr' input with 'rdclk'**.

Note: For correct timing analysis, Intel recommends enabling the **Removal and Recovery Analysis** option in the Timing Analyzer tool when you use the `aclr` signal. The analysis is turned on by default in the Timing Analyzer tool.

When the **Add circuit to synchronize 'aclr' input with 'wrclk'** and **Add circuit to synchronize 'aclr' input with 'rdclk'** options are enabled, you can apply the following false path assignment on the reset path:

- `set_false_path -to *dcfifo:dcfifo_component| dcfifo_*:auto_generated|dffpipe_*:wraclr|dffe*a[0]`
- `set_false_path -to *dcfifo:dcfifo_component| dcfifo_*:auto_generated|dffpipe_*:rdaclr|dffe*a[0]`

4.3.9. SCFIFO and DCFIFO Show-Ahead Mode

You can set the read request/`rdreq` signal read access behavior by selecting normal or show-ahead mode.

For normal mode, the FIFO Intel FPGA IP core treats the `rdreq` port as a normal read request that only performs read operation when the port is asserted.

For show-ahead mode, the FIFO Intel FPGA IP core treats the `rdreq` port as a read-acknowledge that automatically outputs the first word of valid data in the FIFO Intel FPGA IP core (when the `empty` is low) without asserting the `rdreq` signal. Asserting the `rdreq` signal causes the FIFO Intel FPGA IP core to output the next data word, if available.

Figure 35. Normal Mode Waveform

Data appears after the `rdreq` asserted.

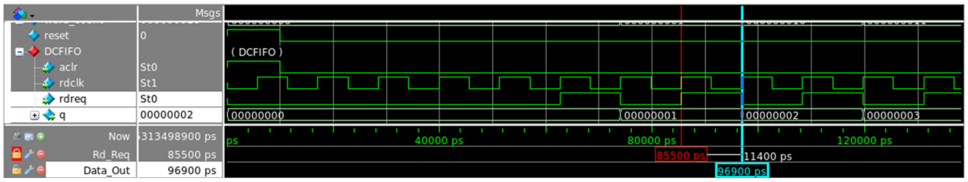
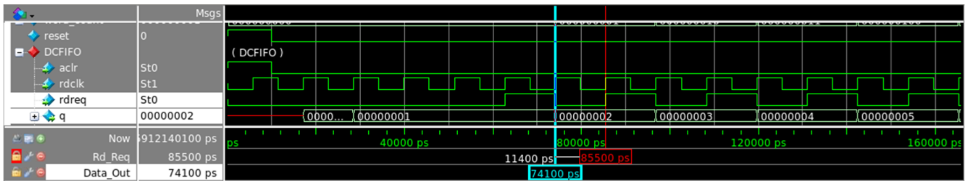


Figure 36. Show-Ahead Mode Waveform

Data appears before the `rdreq` asserted.



4.3.10. Different Input and Output Width

The DCFIFO_MIXED_WIDTHS function supports different write input data and read output data widths if the width ratio is valid. The FIFO parameter editor prompts an error message if the combinations of the input and the output data widths produce an invalid ratio. The supported width ratio in a power of 2 and depends on the RAM.

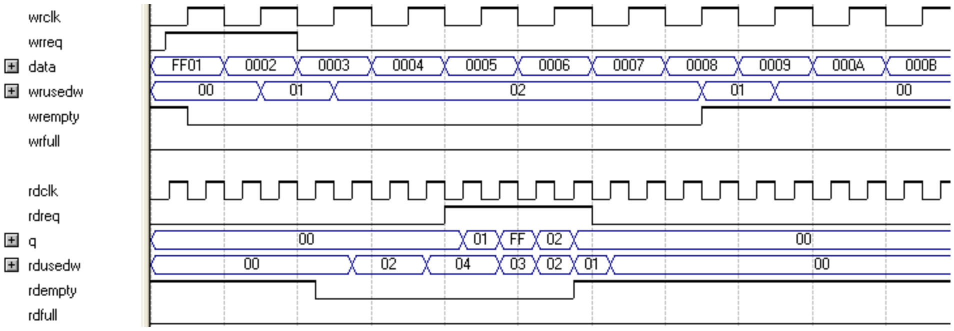
The IP core supports a wide write port with a narrow read port, and vice versa. The current supported mixed width ratios for Intel Stratix 10 devices are listed in the following table:

Table 48. Device Family Support for Width Ratios

Device Family	Valid Width Ratio
Intel Stratix 10	1, 2, 4, 8, 16, and 32

Figure 37. Writing 16-bit Words and Reading 8-bit Words

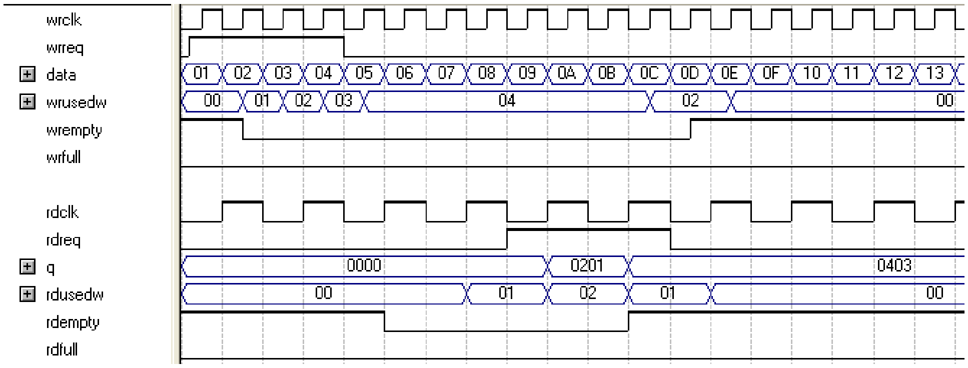
This figure shows an example of a wide write port (16-bit input) and a narrow read port (8-bit output).



In this example, the read port is operating at twice the frequency of the write port. Writing two 16-bit words to the FIFO buffer increases the `wrusedw` flag to two and the `rusedw` flag to four. Four 8-bit read operations empty the FIFO buffer. The read begins with the least-significant 8 bits from the 16-bit word written followed by the most-significant 8 bits.

Figure 38. Writing 8-Bit Words and Reading 16-Bit Words

This figure shows an example of a narrow write port (8-bit input) with a wide read port (16-bit output).



In this example, the read port is operating at half the frequency of the write port. Writing four 8-bit words to the FIFO buffer increases the `wrusedw` flag to four and the `rusedw` flag to two. Two 16-bit read operations empty the FIFO. The first and second

8-bit word written are equivalent to the LSB and MSB of the 16-bit output words, respectively. The `rdempty` signal stays asserted until enough words are written on the narrow write port to fill an entire word on the wide read port.

4.3.11. DCFIFO Timing Constraint Setting

The FIFO parameter editor provides the timing constraint setting for the DCFIFO function.

Table 49. DCFIFO Timing Constraint Setting Parameter in Intel Quartus Prime Software

Parameter	Description
Generate SDC File and disable embedded timing constraint [39]	Allows you to bypass embedded timing constraints that uses `set_false_path` in the synchronization registers. A user configurable SDC file is generated automatically when DCFIFO is instantiated from the IP Catalog. New timing constraints consist of `set_net_delay`, `set_max_skew`, `set_min_delay` and `set_max_delay` are used to constraint the design properly. *Note:* Intel recommends that you select this option for high frequency DCFIFO design to achieve timing closure. For more information, refer to User Configurable Timing Constraint on page 87.

4.3.11.1. Embedded Timing Constraint

When using the Intel Quartus Prime Timing Analyzer with a design that contains a DCFIFO block apply the following false paths to avoid timing failures in the synchronization registers:

- For paths crossing from the write into the read domain, apply a false path assignment between the `delayed_wrptr_g` and `rs_dgwp` registers:

  ```
  set_false_path -from [get_registers
  {*dcfifo*delayed_wrptr_g[*]}] -to [get_registers
  {*dcfifo*rs_dgwp*}]
  ```

- For paths crossing from the read into the write domain, apply a false path assignment between the `rdptr_g` and `ws_dgrp` registers:

  ```
  set_false_path -from [get_registers {*dcfifo*rdptr_g[*]}] -to
  [get_registers {*dcfifo*ws_dgrp*}]
  ```

The false path assignments are automatically added through the HDL-embedded Synopsis design constraint (SDC) commands when you compile your design. The related message is shown under the Timing Analyzer report.

Note: The constraints are internally applied but are not written to the Synopsis Design Constraint File (**.sdc)**. To view the embedded-false path, type `report_sdc` in the console pane of the Timing Analyzer GUI.

If you use the Intel Quartus Prime Timing Analyzer, the false paths are applied automatically for the DCFIFO.

[39] You can disable the embedded timing constraint with QSF setting in prior Intel Quartus Prime versions and other devices. Please refer to KDB link on the QSF assignment setting.

Note: If the DCFIFO is implemented in ALMs, you can ignore the cross-domain timing violations from the data path of the DFFE array (that makes up the memory block) to the `q` output register. To ensure the `q` output is valid, sample the output only after the `rdempty` signal is deasserted.

4.3.11.2. User Configurable Timing Constraint

DCFIFO contains multi-bit gray-coded asynchronous clock domain crossing (CDC) paths which derives the DCFIFO fill-level. In order for the logic to work correctly, the value of the multi-bit must always be sampled as 1-bit change at a given latching clock edge.

In the physical world, flip-flops do not have the same data and clock path insertion delays. It is important for you to ensure and check the 1-bit change property is properly set. You can confirm this using the Fitter and check using the Timing Analyzer.

Timing Analyzer will apply the following timing constraints for DCFIFO:

- Paths crossing from write into read domain are defined from the `delayed_wrptr_g` to `rs_dgwp` registers.

  -
    ```
    set from_node_list [get_keepers $hier_path|dcfifo_component|
    auto_generated|delayed_wrptr_g*]
    ```

  -
    ```
    set to_node_list [get_keepers $hier_path|dcfifo_component|
    auto_generated|rs_dgwp|dffpipe*|dffe*]
    ```
- Paths crossing from read into write domain are defined from the `rdptr_g` and `ws_dgrp` registers.

  -
    ```
    set from_node_list [get_keepers $hier_path|dcfifo_component|
    auto_generated|*rdptr_g*]
    ```

  -
    ```
    set to_node_list [get_keepers $hier_path|dcfifo_component|
    auto_generated|ws_dgrp|dffpipe*|dffe*]
    ```
- For the above paths which cross between write and read domain, the following assignments apply:

  -
    ```
    set_max_skew -from $from_node_list -to $to_node_list
    -get_skew_value_from_clock_period src_clock_period -
    skew_value_multiplier 0.8
    ```

  -
    ```
    set_min_delay -from $from_node_list -to $to_node_list -100
    ```

  -
    ```
    set_max_delay -from $from_node_list -to $to_node_list 100
    ```

  -
    ```
    set_net_delay -from $from_node_list -to $to_node_list -max
    -get_value_from_clock_period dst_clock_period -value_multiplier 0.8
    ```
- The following `set_net_delay` on cross clock domain nets are for metastability:.

  -
    ```
    set from_node_mstable_list [get_keepers $hier_path|dcfifo_component|
    auto_generated|ws_dgrp|dffpipe*|dffe*]
    set to_node_mstable_list [get_keepers $hier_path|dcfifo_component|
    auto_generated|ws_dgrp|dffpipe*|dffe*]
    ```

  -
    ```
    set from_node_mstable_list [get_keepers $hier_path|dcfifo_component|
    auto_generated|rs_dgwp|dffpipe*|dffe*]
    set to_node_mstable_list [get_keepers $hier_path|dcfifo_component|
    auto_generated|rs_dgwp|dffpipe*|dffe*]
    ```

  -
    ```
    set_net_delay -from $from_node_list -to $to_node_list -max -
    get_value_from_clock_period dst_clock_period -value_multiplier 0.8
    ```

Timing Analyzer will apply the following timing constraints for mix-width DCFIFO:

- Paths crossing from write into read domain are defined from the `delayed_wrptr_g` to `rs_dgwp` registers.

  -
    ```
    set from_node_list [get_keepers $hier_path|
    dcfifo_mixed_widths_component|auto_generated|delayed_wrptr_g*]
    ```

  -
    ```
    set to_node_list [get_keepers $hier_path|dcfifo_mixed_widths_component|
    auto_generated|rs_dgwp|dffpipe*|dffe*]
    ```
- Paths crossing from read into write domain are defined from the `rdptr_g` and `ws_dgrp` registers.

  -
    ```
    set from_node_list [get_keepers $hier_path|
    dcfifo_mixed_widths_component|auto_generated|*rdptr_g*]
    ```

  -
    ```
    set to_node_list [get_keepers $hier_path|dcfifo_mixed_widths_component|
    auto_generated|ws_dgrp|dffpipe*|dffe*]
    ```
- For the above paths which cross between write and read domain, the following assignments apply:

  -
    ```
    set_max_skew -from $from_node_list -to $to_node_list -
    get_skew_value_from_clock_period src_clock_period -
    skew_value_multiplier 0.8
    ```

  -
    ```
    set_min_delay -from $from_node_list -to $to_node_list -100
    ```

  -
    ```
    set_max_delay -from $from_node_list -to $to_node_list 100
    ```

  -
    ```
    set_net_delay -from $from_node_list -to $to_node_list -max -
    get_value_from_clock_period dst_clock_period -value_multiplier 0.8
    ```
- The following `set_net_delay` on cross clock domain nets are for metastability:

  -
    ```
    set from_node_mstable_list [get_keepers $hier_path|
    dcfifo_mixed_widths_component|auto_generated|ws_dgrp|dffpipe*|dffe*]
    set to_node_mstable_list [get_keepers $hier_path|
    dcfifo_mixed_widths_component|auto_generated|ws_dgrp|dffpipe*|dffe*]
    ```

  -
    ```
    set from_node_mstable_list [get_keepers $hier_path|
    dcfifo_mixed_widths_component|auto_generated|rs_dgwp|dffpipe*|dffe*]
    set to_node_mstable_list [get_keepers $hier_path|
    dcfifo_mixed_widths_component|auto_generated|rs_dgwp|dffpipe*|dffe*]
    ```

  -
    ```
    set_net_delay -from $from_node_list -to $to_node_list -max -
    get_value_from_clock_period dst_clock_period -value_multiplier 0.8
    ```

4.3.11.2.1. SDC Commands

Table 50. SDC Commands usage in the Intel Quartus Prime Fitter and Timing Analyzer

These SDC descriptions provided are overview for DCFIFO use case. For the exact SDC details, refer to the Intel Quartus Prime Timing Analyzer chapter in the Intel Quartus Prime Pro Edition Handbook.

SDC Command	Fitter	Timing Analyzer	Recommended Settings
`set_max_skew` [40]	To constraint placement and routing of flops in the multi-bit CDC paths to meet the specified skew requirement among bits.	To analyze whether the specified skew requirement is fully met. Both clock and data paths are taken into consideration.	Set to less than 1 launch clock.
`set_net_delay`	Similar to `set_max_skew` but without taking clock skews into considerations. To ensure the crossing latency is bounded.	To analyze whether the specified net delay requirement is fully met. Clock paths are not taken into consideration.	This is currently set to be less than 1 latch clock. [41]
`set_min_delay/` `set_max_delay`	To relax fitter effort by mimicking the `set_false_path` command but without overriding other SDCs. [42]	To relax timing analysis for the setup/hold checks to not fail. [43]	This is currently set to 100ns/-100ns for max/min. [44]

4.3.12. Coding Example for Manual Instantiation

This section provides a Verilog HDL coding example to create an instance of the DCFIFO. It is not a complete coding for you to compile, but it provides a guideline and some comments for the required structure of the instantiation. You can use the same structure to instantiate other IP cores but only with the ports and parameters that are applicable to the IP cores you instantiated.

Example 1. Verilog HDL Coding Example to Instantiate the DCFIFO

```
//module declaration
module dcfifo8x32 (aclr, data, …… ,wfull);
//Module's port declarations
input aclr;
input [31:0] data;
.
.
output wrfull;
//Module’s data type declarations and assignments
wire rdempty_w;
```

[40] It can have significant compilation time impact in older Quartus versions without Timing Analyzer 2.

[41] For advanced users, you can can fine-tune the value based on your design. For instance, if the designs are able to tolerate longer crossing latency (full and empty status will be delayed), this can be relaxed.

[42] Without `set_false_path` (which has the highest precedence and may result in very long insertion delays), Fitter will attempt to meet the default setup/hold which is extremely over constraint.

[43] Without `set_false_path`, the CDC paths will be analyzed for default setup/hold, which is extremely over constraint.

[44] Expect an approximately 100ns delay when you observe CDC paths compared to `set_false_path`.

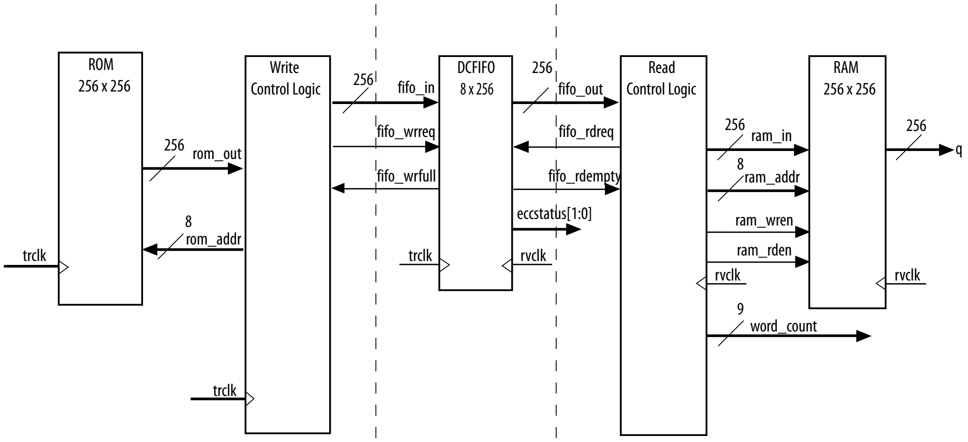
```
.
.
wire wrfull = wrfull_w; wire [31:0] q = q_w;
/*Instantiates dcfifo megafunction. Must declare all the ports available from
the megafunction and
define the connection to the module's ports.
Refer to the ports specification from the user guide for more information about
the megafunction's
ports*/
//syntax: <megafunction's name> <given an instance name>
dcfifo inst1 (
//syntax: .<dcfifo's megafunction's port>(<module's port/wire>)
.wrclk (wrclk),
.rdclk (rdclk),
.
.
.wrusedw ()); //left the output open if it's not used
/*Start with the keyword "defparam", defines the parameters and value
assignments. Refer to
parameters specifications from the user guide for more information about the
megafunction's
parameters*/
defparam
//syntax: <instance name>.<parameter> = <value>
inst1.intended_device_family = "Stratix 10",
inst1.lpm_numwords = 8,
.
.
inst1.wrsync_delaypipe = 4;
endmodule
```

4.3.13. Design Example

In this design example, the data from the ROM is required to be transferred to the RAM. Assuming the ROM and RAM are driven by non-related clocks, you can use the DCFIFO to transfer the data between the asynchronous clock domains effectively.

Figure 39. Component Blocks and Signal Interaction

This figure shows the component blocks and their signal interactions.



Note: The DCFIFO functions are with ECC feature enabled and implemented using M20K.

Note: Both the DCFIFO functions are only capable of handling asynchronous data transferring issues (metastable effects). You must have a controller to govern and monitor the data buffering process between the ROM, DCFIFO, and RAM. This design example provides you the write control logic (`write_control_logic.v`), and the read control logic (`read_control_logic.v`) which are compiled with the DCFIFO specifications that control the valid write or read request to or from the DCFIFO.

Note: This design example is validated with its functional behavior, but without timing analysis and gate-level simulation. The design coding such as the state machine for the write and read controllers may not be optimized. The intention of this design example is to show the use of the IP core, particularly on its control signal in data buffering application, rather than the design coding and verification processes.

To obtain the DCFIFO settings in this design example, refer to the parameter settings from the design file (`dcfifo8x32.v`).

The following sections include separate simulation waveforms to describe how the write and read control logics generate the control signal with respect to the signal received from the DCFIFO.

Note: For better understanding, refer to the signal names in the above figure when you go through the descriptions for the simulation waveforms.

Note: All signals in the following figures and tables have the following numerical format:

- Signal values in binary format: `reset`, `trclk`, `fifo_wrreq`, `fifo_wrfull`
- Signal values in HEX format: `rom_addr`, `rom_out`, `fifo_in`

Figure 40. Initial Write Operation to the DCFIFO Function

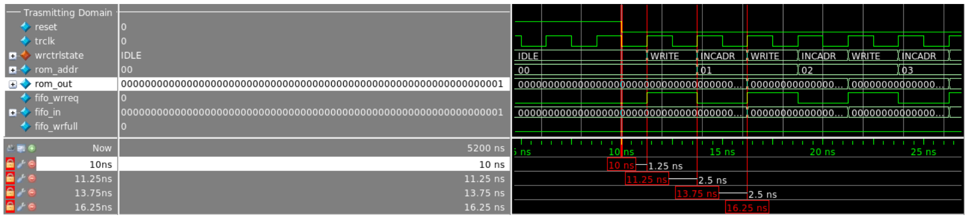


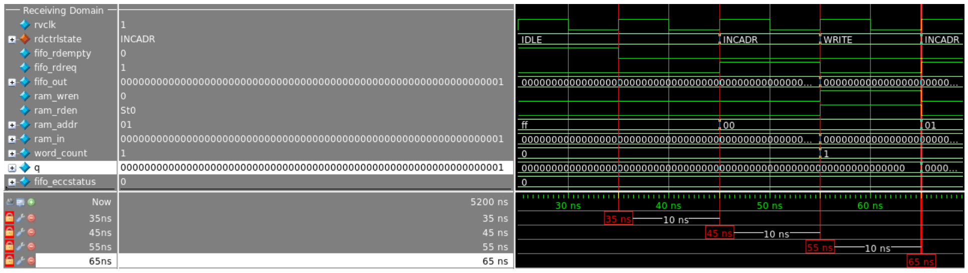
Table 51. Initial Write Operation to the DCFIFO Function Waveform Description

State	Description
`IDLE`	Before reaching 10 ns, the `reset` signal is high and causes the write controller to be in the `IDLE` state. In the `IDLE` state, the write controller drives the `fifo_wrreq` signal to low, and requests the data to be read from `rom_addr=00`. The ROM is configured to have an unregistered output, so that the `rom_out` signal immediately shows the data from the `rom_addr` signal regardless of the reset. This shortens the latency because the `rom_out` signal is connected directly to the `fifo_in` signal, which is a registered

continued...

State	Description
	input port in the DCFIFO. In this case, the data (0000000000000000000000000000000000000000000000000000000000000001) is always stable and pending to be written into the DCFIFO when the fifo_wrreq signal is high during the WRITE state.
WRITE	The write controller transitions from the IDLE state to the WRITE state if the fifo_wrfull signal is low after the reset signal is deasserted. In the WRITE state, the write controller drives the fifo_wrreq signal to high, and requests for write operation to the DCFIFO. The data is encoded through the embedded ECC block in the DCFIFO. The rom_addr signal is unchanged (00) so the data is stable for at least one clock cycle before the DCFIFO actually writes in the data at the next rising clock edge.
INCADR	The write controller transitions from the WRITE state to the INCADR state, if the rom_addr signal has not yet increased to ff (that is, the last data from the ROM has not been read out). In the INDADR state, the write controller drives the fifo_wrreq signal to low, and increases the rom_addr signal by 1 (00 to 01).
-	The same state transition continues as stated in IDLE and WRITE states, if the fifo_wrfull signal is low and the rom_addr signal not yet increased to ff.

Figure 41. Initial Read Operation from the DCFIFO Function



Receiving Domain	
rvclk	1
rdctrlstate	INCADR
fifo_rdempty	0
fifo_rdreq	1
fifo_out	0000000000000000000000000000000000000000000000000000000000000001
ram_wren	0
ram_rden	St0
ram_addr	01
ram_in	0000000000000000000000000000000000000000000000000000000000000001
word_count	1
q	0000000000000000000000000000000000000000000000000000000000000001
fifo_eccstatus	0
Now	5200 ns
35ns	35 ns
45ns	45 ns
55ns	55 ns
65ns	65 ns

Table 52. Initial Read Operation from the DCFIFO Function Waveform Description

State	Description
IDLE	Before reaching 35 ns, the read controller is in the IDLE state because the fifo_rdempty signal is high even when the reset signal is low (not shown in the waveform). In the IDLE state, the ram_addr = ff to accommodate the increment of the RAM address in the INCADR state, so that the first data read is stored at ram_addr = 00 in the WRITE state.
INCADR	The read controller transitions from the IDLE state to the INCADR state, if the fifo_rdempty signal is low. In the INCADR state, the read controller drives the fifo_rdreq signal to high, and requests for read operation from the DCFIFO. The data is decoded and the eccstatus shows the status of the data as no error detected (00), single-bit error detected and corrected(10), or uncorrectable error (11). The ram_addr signal is increased by one (ff to 00), so that the read data can be written into the RAM at ram_addr = 00.
WRITE	From the INCADR state, the read controller always transition to the WRITE state at the next rising clock edge. In the WRITE state, it drives the ram_wren signal to high, and enables the data writing into the RAM at ram_addr = 00. At the same time, the read controller drives the ram_rden signal to high so that the newly written data is output at q at the next rising clock edge. Also, it increases the word_count signal to 1 to indicate the number of words successfully read from the DCFIFO.
--	The same state transition continues as stated in INCADR and WRITE states, if the fifo_rdempty signal is low.

Figure 42. Write Operation when DCFIFO is FULL

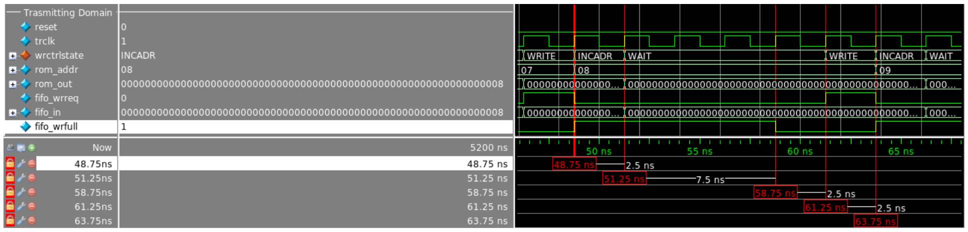


Table 53. Write Operation when DCFIFO is FULL Waveform Description

State	Description
INCADR	When the write controller is in the INCADR state, and the fifo_wrfull signal is asserted, the write controller transitions to the WAIT state in the next rising clock edge.
WAIT	In the WAIT state, the write controller holds the rom_addr signal (08) so that the respective data is written into the DCFIFO when the write controller transitions to the WRITE state. The write controller stays in WAIT state if the fifo_wrfull signal is still high. When the fifo_wrfull is low, the write controller always transitions from the WAIT state to the WRITE state at the next rising clock edge.
WRITE	In the WRITE state, then only the write controller drives the fifo_wrreq signal to high, and requests for write operation to write the data from the previously held address (08) into the DCFIFO. It always transitions to the INCADR state in the next rising clock edge, if the rom_addr signal has not yet increased to ff.
--	The same state transition continues as stated in INCADR, WAIT, and WRITE states, if the fifo_wrfull signal is high.

Figure 43. Completion of Data Transfer from ROM to DCFIFO

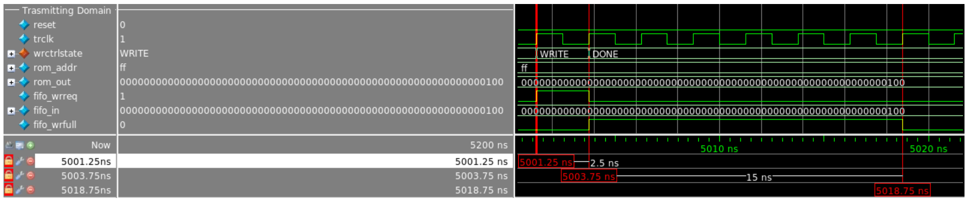
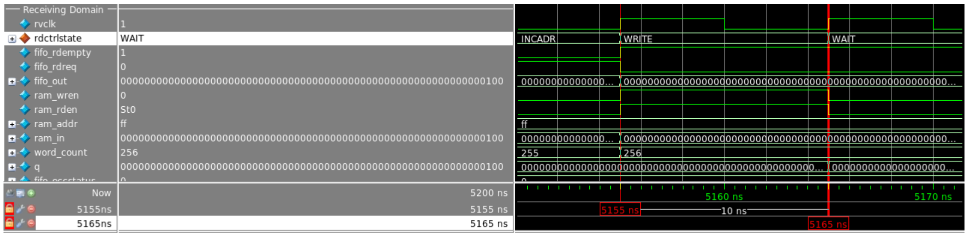


Table 54. Completion of Data Transfer from ROM to DCFIFO Waveform Description

State	Description
WRITE	When the write controller is in the WRITE state, and rom_addr = ff, the write controller drives the fifo_wrreq signal to high to request for last write operation to DCFIFO. The data 100 is the last data stored in the ROM to be written into the DCFIFO. In the next rising clock edge, the write controller transitions to the DONE state.
DONE	In the DONE state, the write controller drives the fifo_wrreq signal to low.
--	The fifo_wrfull signal is deasserted because the read controller in the receiving domain continuously performs the read operation. However, the fifo_wrfull signal is only deasserted sometime after the read request from the receiving domain. This is due to the latency in the DCFIFO (rdreq signal to wrfull signal).

Figure 44. Completion of Data Transfer from DCFIFO to RAM



```
Receiving Domain
rvclk          1
rdctrlstate    WAIT
fifo_rdempty   1
fifo_rdreq     0
fifo_out       0000000000000000000000000000000000000000000000000000000000000100
ram_wren       0
ram_rden       St0
ram_addr       ff
ram_in         0000000000000000000000000000000000000000000000000000000000000100
word_count     256
q              0000000000000000000000000000000000000000000000000000000000000100
Now            5200 ns
5155ns         5155 ns
5165ns         5165 ns
```

The `fifo_rdempty` signal is asserted to indicate that the DCFIFO is empty. The read controller drives the `fifo_rdreq` signal to low, and enables the write of the last data 100 at `ram_addr =ff`. The `word_count` signal is increased to 256 (in decimal) to indicate that all the 256 words of data from the ROM are successfully transferred to the RAM.

The last data written into the RAM is shown at the q output.

Note: To verify the results, compare the q outputs with the data in `rom_initdata.hex` file provided in the design example. Open the file in the Intel Quartus Prime software and select the word size as 256 bit. The q output must display the same data as in the file.

4.3.14. Gray-Code Counter Transfer at the Clock Domain Crossing

This section describes the effect of the large skew between Gray-code counter bits transfers at the clock domain crossing (CDC) with recommended solution. The gray-code counter is 1-bit transition occurs while other bits remain stable when transferring data from the write domain to the read domain and vice versa. If the destination domain latches on the data within the metastable range (violating setup or hold time), only 1 bit is uncertain and destination domain reads the counter value as either an old counter or a new counter. In this case, the DCFIFO still works, as long as the counter sequence is not corrupted.

The following section shows an example of how large skew between GNU C compiler (GCC) bits can corrupt the counter sequence. Taking a counter width with 3-bit wide and assuming it is transferred from write clock domain to read clock domain. Assume all the counter bits have 0 delay relative to the destination clock, excluding the `bit[0]` that has delay of 1 clock period of source clock. That is, the skew of the counter bits will be 1 clock period of the source clock when they arrived at the destination registers.

The following shows the correct gray-code counter sequence:

```
000,
001,
011,
010,
110....
```

which then transfers the data to the read domain, and on to the destination bus registers.

Because of the skew for bit[0], the destination bus registers receive the following sequence:

```
000,
000,
011,
011,
110....
```

Because of the skew, a 2-bit transition occurs. This sequence is acceptable if the timing is met. If the 2-bit transition occurs and both bits violate timing, it may result in the counter bus settled at a future or previous counter value, which will corrupt the DCFIFO.

Therefore, the skew must be within a certain skew to ensure that the sequence is not corrupted.

Note: Use the report_max_skew and report_net_delay reports in the Timing Analyzer for timing verification if you use the User Configurable Timing Constraint. For Embedded Timing Constraint, use the skew_report.tcl to analyze the actual skew and required skew in your design.

4.3.15. Guidelines for Embedded Memory ECC Feature

The Intel Stratix 10 FIFO Intel FPGA IP cores support embedded memory ECC for M20K memory blocks. The built-in ECC feature in the Intel Stratix 10 devices can perform:

- Single-error detection and correction
- Double-adjacent-error detection and correction
- Triple-adjacent-error detection

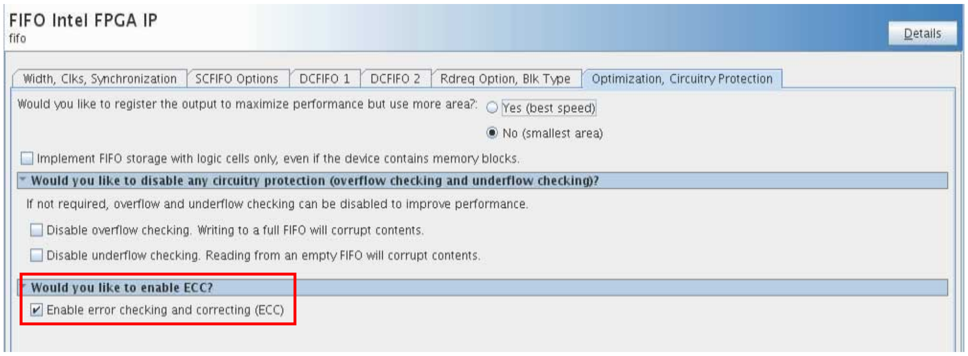
You can turn on FIFO Embedded ECC feature by enabling enable_ecc parameter in the FIFO Intel FPGA IP GUI.

Note: Embedded memory ECC feature is only available for M20K memory block type.

Note: The embedded memory ECC supports variable data width. When ECC is enabled, RAM combines multiple M20K blocks in the configuration of 32 (width) x 512 (depth) to fulfill your instantiation. The unused data width will be tied to the V_{CC} internally.

Note: The embedded memory ECC feature is not supported in mixed-width mode.

Figure 45. ECC Option in FIFO Intel FPGA IP GUI



Would you like to register the output to maximize performance but use more area?: ○ Yes (best speed) ● No (smallest area)

□ Implement FIFO storage with logic cells only, even if the device contains memory blocks.

Would you like to disable any circuitry protection (overflow checking and underflow checking)?

If not required, overflow and underflow checking can be disabled to improve performance.

□ Disable overflow checking. Writing to a full FIFO will corrupt contents.

□ Disable underflow checking. Reading from an empty FIFO will corrupt contents.

Would you like to enable ECC?

☑ Enable error checking and correcting (ECC)

When you enable the ECC feature, a 2-bit wide error correction status port (`eccstatus[1:0]`) will be created in the generated FIFO entity. These status bits indicate whether the data that is read from the memory has an error in single-bit with correction, fatal error with no correction, or no error bit.

- 00: No error
- 01: Illegal
- 10: A correctable error occurred and the error has been corrected at the outputs; however, the memory array has not been updated.
- 11: An uncorrectable error occurred and uncorrectable data appears at the output

4.3.16. FIFO Intel FPGA IP Parameters

Table 55. FIFO Intel FPGA IP Parameters Description

This table lists the parameters for the FIFO Intel FPGA IP core.

Parameter	Legal Values	Description
Parameter Settings: Width, Clk, Synchronization		
How wide should the FIFO be?	—	Specifies the width of the data and q ports.
How deep should the FIFO be? Note: You could enter arbitrary values for width	4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072	Specifies the depth of the FIFO, which is always a power of 2.
Do you want a common clock for reading and writing the FIFO?	• Yes, synchronize both reading and writing to 'clock'. Create one set of full/empty control signals. • No, synchronize reading and writing to 'rdclk' and 'wrclk', respectively. Create a set of full/empty control signals for for each clock.	—
Parameter Settings: SCFIFO Options		
Would you like to disable any circuitry protection?	On/Off	—

continued...

Parameter	Legal Values	Description
• full • empty • usedw[] (number of words in FIFO). Note: You can use the MSB to generate a half full flag. • almost full becomes true when usedw[] is greater than or equal to • almost empty becomes true when usedw[] is less than • Asyncronous clear • Syncronous clear (flush the FIFO)		
Parameter Settings: DCFIFO 1		
When you select **No, synchronize reading and writing to 'rdclk' and 'wrclk', respectively. Create a set of full/empty control signals for for each clock.**, the following options are available: Total latency, clock synchronization, metastability protection, area, and fmax options must be set as a group. Total latency is the sum of two write clock rising edges and the number of read clocks selected below. Which option(s) is most important to the DCFIFO? (Read clk sync stages, metastability protection, area, fmax) Which type of optimization do you want? • Lowest latency but requires synchronized clocks. 1 sync stage, no metastability protection, smallest size, good fmax. • Minimal setting for unsynchronized clocks. 2 sync stages, good metastability, medium size, good fmax. • Best metastability protection, best fmax, unsynchronized clocks. 3 or more sync stages, best metastability protection, largest size, best fmax.	On/Off	Specify total latency, clock synchronization, metastability protection, area, and fmax. • **Lowest latency but requires synchronized clocks**—This option uses one synchronization stage with no metastability protection. It uses the smallest size and provides good f_{MAX}. Select this option if the read and write clocks are related clocks. • **Minimal setting for unsynchronized clocks**—This option uses two synchronization stages with good metastability protection. It uses the medium size and provides good f_{MAX}. • **Best metastability protection, best fmax, unsynchronized clocks**—This option uses three or more synchronization stages with the best metastability protection. It uses the largest size but gives the best f_{MAX}.
More options When you select **Best metastability protection, best fmax, unsynchronized clock**, the following option is available: • How many sync stages?	3, 4, 5, 6, 7, 8, 9	Specifies the number syncronization stages.
Timing Constraint • Generate SDC file and disable embedded timing constraint	On/Off	Generate a SDC file with correct timing constraints. Embedded `set_false_path` assignment is disabled. The new timing constraints consist of `set_net_delay`, `set_max_skew`, `set_min_delay` and `set_max_delay`. For more infomation on the timing constraint usage, refer to user guide.
Parameter Settings: DCFIFO 2		
When you select **No, synchronize reading and writing to 'rdclk' and 'wrclk', respectively. Create a set of full/empty control signals for for each clock.**, the following options are available: Which optional output control signals do you want? usedw[] is the number of words in the FIFO.	On/Off	

continued...

Parameter	Legal Values	Description
Read-side • full • empty • usedw[] Note: These signals are syncronous to 'rdclk'.		
Write-side • full • empty • usedw[] Note: These signals are syncronous to 'wrclk'.		
More options • Add an extra MSB to usedw port(s). Note: You can use the MSB to generate a half-full flag. • Asynchronous clear • Add circuit to synchronize 'aclr' input with 'wrclk' • Add circuit to synchronize 'aclr' input with 'rdclk'	On/Off	
Parameter Settings: Rdreq Option, Blk Type		
Which kind of read access do you want with the 'rdreq' signal?	• Normal synchronous FIFO mode. • Show-ahead synchronous FIFO mode.	Specifies whether the FIFO is in Legacy mode or in Show-ahead mode. • **Normal synchronous FIFO mode**—The data becomes available after 'rdreq is asserted. 'rdreq' acts as a read request. • **Show-ahead synchronous FIFO mode**—The data becomes available before 'rdreq' is asserted. 'rdreq' acts as a read acknowledge. Note: This mode suffers a performance penalty.
What should the memory block type be?	• Auto • MLAB • M20K • M144K	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
Set the maximum block depth to:	Auto, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072	Specifies the maximum block depth in words.
Reduce RAM usage (decreases speed and increases number of Les). Available if data width is divisible by 9.	On/Off	
Parameter Settings: Optimization, Circuitry Protection		
Would you like to register the output to maximize performance but use more area?	• Yes (best speed) • No (smallest area)	Specifies whether to register the RAM output.
Implement FIFO storage with logic cells only, even if the device contains memory blocks.	On/Off	Specifies whether to implement FIFO storage with logic cells only.
Would you like to disable any circuitry protection (overflow checking and underflow checking)? If not required, overflow and underflow checking can be disabled to improve performance.	On/Off	Specifies whether to disable any circuitry protection for overflow

continued...

Parameter	Legal Values	Description
• Disable overflow checking. Writing to a full FIFO will corrupt contents. • Disable underflow checking. Reading from an empty FIFO will corrupt contents		
Would you like to enable ECC? • Enable error checking and correcting (ECC)	On/Off	Specifies whether to enable error checking and correcting feature.

4.3.17. Reset Scheme

During power up, the registers in Intel Stratix 10 devices are in undefined power and reset states. To guarantee correct functionality, reset the FIFO Intel FPGA IP core upon completion of configuration by asserting either the `sclr` or `aclr` signal. Reset is not required if `sclr` or `aclr` signal is not used in the FIFO Intel FPGA IP core.

4.4. FIFO2 Intel FPGA IP

Intel provides FIFO2 Intel FPGA IP core as an alternative solution for the FIFO Intel FPGA IP core for applications running at wide data width and very high operating frequencies (Fmax) to achieve high data bandwidth.

The FIFO functions in the FIFO2 Intel FPGA IP core are mostly applied in data buffering applications that comply with the first-in-first-out data flow in synchronous or asynchronous clock domains.

Note: The FIFO2 Intel FPGA IP core has no backward compatibility to the FIFO Intel FPGA IP core.

Table 56. Differences between FIFO and FIFO2 Intel FPGA IP Cores

Feature	Intel FPGA IP Cores	
	FIFO	FIFO2
Read latency	0 - 1 clock cycle after the `rdreq` signal is asserted.	3 - 4 clock cycle after the `rdreq` is asserted.
Read valid when	`r_empty` signal is low	`r_valid` signal is high
Show-ahead mode	Supported	Not supported
Depth (D) and width (W) configuration	Per user requirement	Multiple of hard memory block only (32W x 512D for M20K, 20W x 32D for MLAB)
Output data initial state	0	Unknown
Flushing	Not required	A minimum of 32 slow clock cycle flushings are required

Before any read-out operations, the applications data is first written (partially or entirely) into the FIFO2 Intel FPGA IP core. The data read operations can be in long continuous bursts or a single clock read. While there is no specific write or read limitation, the bandwidth utilization will be less efficient for short writes and/or reads due to incurred latencies.

The read interface of the FIFO2 Intel FPGA IP core is suitable for applications that does not perform back-pressure or for applications with a "cascaded" buffer further downstream.

For example,

- At MAC RX user interface, which typically cannot be back-pressured, which is equivalent to always read.
- Along MAX TX internal data path to harden Native PHY FIFO. The FIFO read operations may then be derived from the Native PHY FIFO partial full status.

User application can connect to the read interface of the FIFO2 Intel FPGA IP core directly to a small SCFIFO (or similar storage buffers) externally to change the read-to-data latency to be zero but at the expense of Fmax and resources.

In practice, all clocks run at several hundred MHz. This is because the FIFO2 Intel FPGA IP core is highly pipelined to run at very high Fmax, and is not be suitable for slow clocks due to the long latency.

4.4.1. Release Information for FIFO2 Intel FPGA IP

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP version (X.Y.Z) number may change from one Intel Quartus Prime software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 57. FIFO2 Intel FPGA IP Current Release Information

Item	Description
IP Version	19.1.0
Intel Quartus Prime Version	19.1
Release Date	2019.09.30

4.4.2. Configuration Methods

Table 58. Configuration Methods

You can configure and build the FIFO2 Intel FPGA IP cores with methods shown in the following table.

Method	Description
Using the FIFO2 parameter editor.	Intel recommends using this method to build your FIFO2 Intel FPGA IP cores. It is an efficient way to configure and build the FIFO2 Intel FPGA IP cores. The FIFO2 parameter editor provides options that you can easily use to configure the FIFO2 Intel FPGA IP core. You can access the FIFO2 Intel FPGA IP core parameter editor in **Basic Functions ➤ On Chip Memory ➤ FIFO2** of the IP catalog.[45]
Manually instantiating the FIFO2 Intel FPGA IP cores.	Use this method only if you are an expert user. This method requires that you know the detailed specifications of the IP cores. You must ensure that the input and output ports used, and the parameter values assigned are valid for the FIFO2 Intel FPGA IP cores you instantiate for your target device.

4.4.3. Fmax Target Measuring Methodology

The specified Fmax target is measured using the following conditions:

- IP is compiled as a stand-alone component, and is wrapped inside a wrapper.
- The wrapper has 1 non-resettable flop layer to register all wrapper input and output ports except for clocks.
- The wrapper flop layer must be preserved through synthesis attributes:

```
(* altera_attribute = {"-name DONT_MERGE_REGISTER ON; -name
PRESERVE_REGISTER ON; -name ADV_NETLIST_OPT_ALLOWED NEVER_ALLOW"} *)
```

- All wrapper ports except for clocks are set as virtual pins.

Note: This measurement is not intended to validate actual IP performance when integrated into a real system.

4.4.4. Performance Considerations

A wider FIFO is implemented using either multiple narrow instances or a single wide instance of these building blocks. You can choose based on empirical data or through parameters.

In the FIFO2 Intel FPGA IP core, the Fmax has higher priority than latency. To achieve the targeted Fmax, the design will be piped when necessary. Use the following estimated pipe stages (or latency) as guidelines:

Operation	Estimated Pipe Stages (Latency)
write to data available in storage	~2 read clocks
write pointer binary-to-gray conversion	~2 read clocks
write pointer cross-over to read logic	~4 read clocks

continued...

[45] Do not use dcfifo or scfifo as the entity name for your FIFO2 Platform Designer system.

Operation	Estimated Pipe Stages (Latency)
write pointer gray-to-binary conversion	~2 read clocks
write pointer and read pointer comparison result	~2 read clocks
user read to data available	~6 read clocks

4.4.5. FIFO2 Intel FPGA IP Features

You can configure the FIFO2 Intel FPGA IP core as either a DCFIFO or a SCFIFO by using the parameter editor of the FIFO2 Intel FPGA IP core. When the FIFO2 Intel FPGA IP core is configured as a SCFIFO, the relevant clock domain crossing (CDC) structure will not be synthesized.

The following figures show the timing diagrams for read and write operations for FIFO2 Intel FPGA IP core.

Figure 46. Write to Full with Write Protection

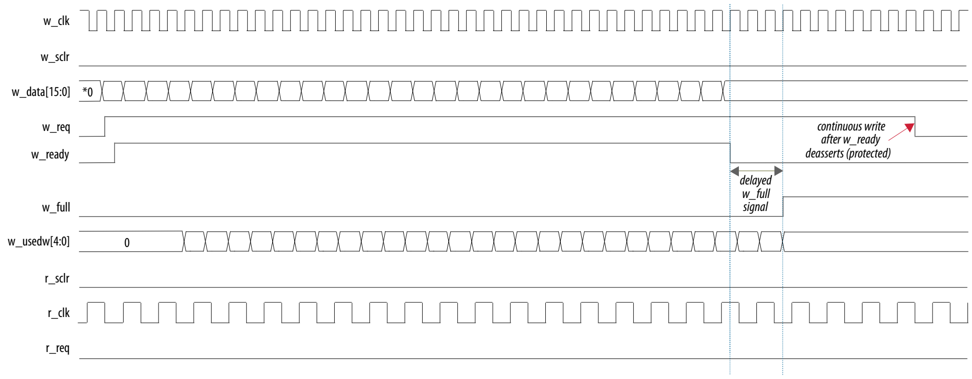
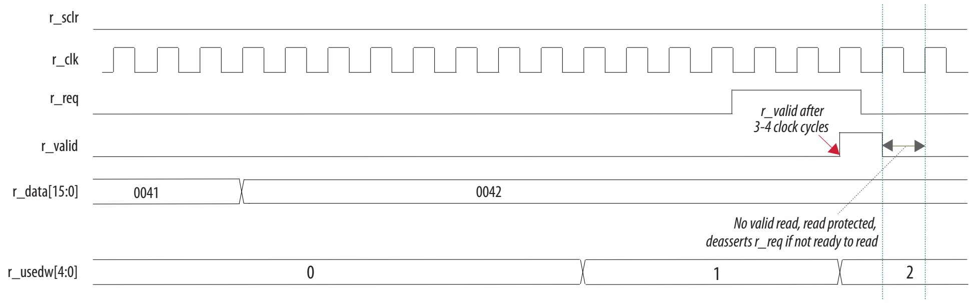


Figure 47. Single Read with Read Protection


r_sclr
r_clk
r_req
r_valid after 3-4 clock cycles
r_valid
r_data[15:0]
0041
0042
No valid read, read protected, deasserts r_req if not ready to read
r_usedw[4:0]
0
1
2

4.4.5.1. FIFO2 Specifications

The following table summarizes the specifications of the FIFO2 Intel FPGA IP core.

Table 59. FIFO2 Specifications

Feature	Storage Type	
	M20K	MLAB
Error Checking and Correcting (ECC)	Always [46]	No
Read-out Interface	Analogy to Avalon ST non-zero readLatency For each r_req asserts now, r_valid shall indicate whether there is valid data to be taken (and must be taken) L clock later. L = 6	Analogy to Avalon ST non-zero readLatency For each r_req asserts now, r_valid shall indicate whether there is valid data to be taken (and must be taken) L clock later. L = 5
Width (bits)	There is no hard limit on user data width but the internal RAM block is always in 32b x N; where N > 0. Maximum = 4096b Default to 1.	There is no hard limit on user data width but the internal RAM block is always in 20b x N; where N > 0. Maximum = 4080b Default to 1.
Depth	512	32
Depth Stitching	No, user can cascade multiple FIFOs	No, user can cascade multiple FIFOs
Targeted Performance	Intel Stratix 10, bin1 production device 32bx512: Up to 850 MHz 512bx512: Up to 700 MHz	Intel Stratix 10, bin1 production device 20bx32: Up to 850 MHz 512bx32: Up to 700 MHz
Almost Full	No, user can derive this from "Write Used"	No, user can derive this from "Write Used"
Almost Empty	No, user can derive this from "Read Used"	No, user can derive this from "Read Used"
Read Used	Yes, delayed RAM block words measurement excluding in-flight data	Yes, delayed RAM block words measurement excluding in-flight data
Write Used	Yes, delayed RAM block words measurement excluding in-flight data	Yes, delayed RAM block words measurement excluding in-flight data
RAM with registered read output	Always	Always
Write full prevention	Always, based on internal almost full	Always, based on internal almost full
Read empty prevention	Always	Always
Output data initial states	Unknown	Unknown
Reset Scheme	Contains non resettable flops, requires state flushing	Contains non resettable flops, requires state flushing
RTL	Encrypted	Encrypted

[46] In the FIFO2 Intel FPGA IP core, the ECC mode is embedded within the IP architecture and cannot be disabled. Unlike FIFO Intel FPGA IP core, there is no ECCSTATUS signal that can be exported for use in your design.

4.4.6. FIFO2 Intel FPGA IP Parameters

Table 60. FIFO2 Intel FPGA IP Core Parameters Description

This table lists the parameters for the FIFO2 Intel FPGA IP core.

Parameter	Legal Values	Description
What type of FIFO you prefer?	• Single-clock • Dual-clock	Specifies the type of FIFO.
How wide should the FIFO be?	—	Specifies the width of the data and q ports.
RAM block type	• MLAB • M20K	Specifies the type of RAM Block used for FIFO
Parameter Settings: Reset Option		
Enable Asynchronous Clear (ACLR)	On/Off	Specifies the write and read are reset asynchronously.
Parameter Settings: Performance Optimization		
Enable per RAM block preserve/duplication for: • RAM write address * • RAM read address * * Note: This will typically increase Fmax at the expense of resources.	On/Off	Enables per RAM block preserve/ duplication for: • RAM write address: Specifies whether RAM write address (and associated logic where appropriate) should be duplicated per RAM block. • RAM read address: Specifies whether RAM read address (and associated logic where appropriate) should be duplicated per RAM block. Note: This will typically increase $_{MAX}$ at the expense of resources.
When you select **Dual-clock**, the following options are available: • The synchronizer chain length for write gray-code pointer • The synchronizer chain length for read gray-code pointer	3, 4	Specify the multi-flop synchronizer chain length for write and read gray-code pointers.

4.4.6.1. FIFO2 Parameter Settings

Table 61. FIFO2 Parameters Description

Parameter	Description
DATAWIDTH	**FIFO Write and Read Data Width.** The user width granularity is as below, depending on the RAM block type: • M20K: 32n; where n = 1 to 128 • MLAB: 20n; where n = 1 to 205 This allows up to 4096 bit width which should be more than enough for different applications. All unused bits (for example, bits that do not carry any information) should be tied-off. For instance, if the user data width were 20-bit and M20K RAM block is used, there would be 12 unused bits to be tied-off. The default value for n is 1.
SCFIFO_MODE	**SCFIFO Mode.**

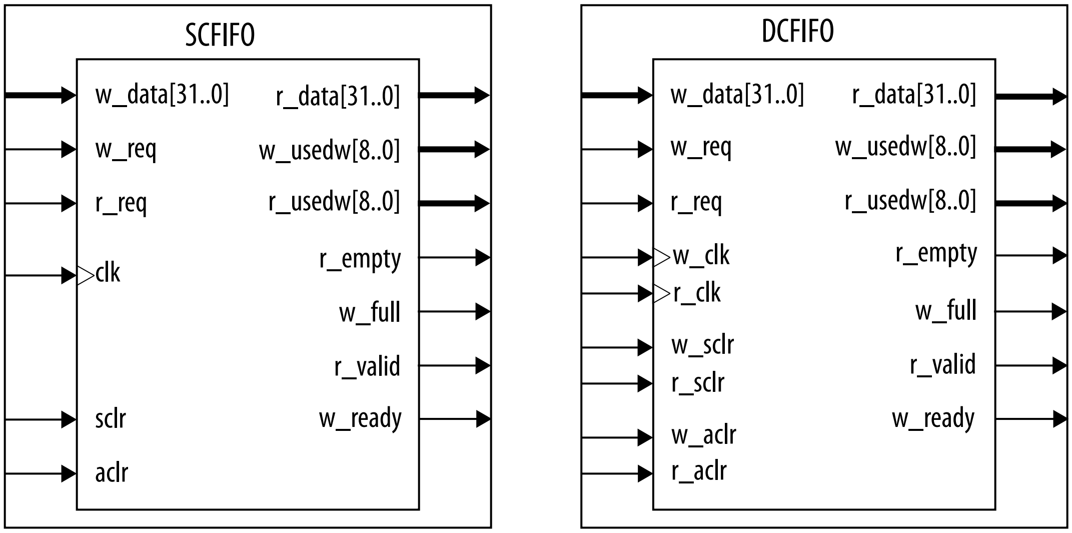
continued...

Parameter	Description
	Specify whether the FIFO should operate in SCFIFO mode, in which the clock crossing logic structure between Write and Read clock domains shall be removed. • 1— SCFIFO mode • 0 (default)—DCFIFO mode
RAM_BLK_TYPE	**RAM Block Type.** Specify the embedded RAM blocks to be used as the main FIFO storage. • "M20K" (default)—Use M20K • "MLAB"—Use MLAB
USE_ACLR_PORT	**Use Asynchronous Clear Port.** Specify whether the asynchronous reset ports (for example, w_aclr and r_aclr) of the IP should have effect. • 1—Ports are used to asynchronously reset the IP • 0 (default)—Ports are not used and have no effect
WRPTR_GRY_SYNC_CHAIN_LEN	**Write Gray-Code Pointer Synchronizer Chain Length.** Specify the number of flop stages used to synchronize Write Gray-Code Pointer to the r_clk domain. • 3 (default)—Use 3-stage synchronizer • 4—Use 4-stage synchronizer
RDPTR_GRY_SYNC_CHAIN_LEN	**Read Gray-Code Pointer Synchronizer Chain Length.** Specify the number of flop stages used to synchronize Read Gray-Code Pointer to the w_clk domain. • 3 (default)—Use 3-stage synchronizer • 4—Use 4-stage synchronizer
RAM_WRPTR_DUPLICATE	**RAM Write Address Duplication.** Specify whether RAM Write Address and associated logic (where appropriate) should be duplicated per RAM block. • 1—Enable per RAM block preserve/duplication. This may increase Fmax at the expense of resources • 0 (default)—Do not enable per RAM block preserve/duplication. You can determine which registers should be duplicated through assignment.
RAM_RDPTR_DUPLICATE	**RAM Read Address Duplication.** Specify whether RAM Read Address (and associated logic where appropriate should be duplicated per RAM block. U • 1—Enable per RAM block preserve/duplication. This may increase Fmax at the expense of resources • 0 (default)—Do not enable per RAM block preserve/duplication. You can determine which registers should be duplicated through assignment.

4.4.7. FIFO2 Intel FPGA IP Interface Signals

This section provides diagrams of the SCFIFO and DCFIFO blocks of the FIFO2 Intel FPGA IP core to help in visualizing their input and output ports. This section also describes each port in detail to help in understanding their usages, functionality, or any restrictions.

Figure 48. FIFO2 IP Core Input and Output Signals



4.4.7.1. SCFIFO Signals

Table 62. SCFIFO Input and Output Ports Description

Signal	Direction	Required	Description
clk	Input	Yes	FIFO Write Clock and Read Clock.
aclr	Input	No	Active-high reset signal that feeds the asynchronous clear pins of clk domain flip-flops. This reset is not synchronized within the IP, and hence, user logic should ensure it is de-asserted synchronously to clk whenever appropriate. This signal only takes effect if USE_ACLR_PORT is enabled. • 0 = reset inactive • 1 = reset active
sclr	Input	No	Active-high reset signal that feeds the synchronous clear pins of clk domain flip-flops. Reset sequence requirements must be followed. • 0 = request inactive • 1 = request active
w_req	Input	Yes	FIFO write request. This signal is expected to be inactive during reset. • 0 = request inactive • 1 = request active
w_data[FIFO_WIDTH-1:0]	Input	Yes	FIFO Write Data. This bus presents the data to be stored into the FIFO when there is a write request. The value is taken by the FIFO only when w_req is active, and the FIFO is not full (i.e. w_full = 1).
w_full	Output	No	FIFO Write Full. This signal indicates whether the space remained in the FIFO is about to run-out. When this signal asserts, further w_req is ignored. *Note:* Due to internal pipeline stages to improve Fmax, the actual usable space is a few entries less than that being configured to prevent data loss.

continued...

Signal	Direction	Required	Description
			• 0 = FIFO is not full • 1 = FIFO is (near) full
r_req	Input	Yes	FIFO Read Request / Read Ready. In order to achieve the highest possible Fmax, the use model of this signal is slightly different from the normal zero read to data ready latency FIFO. User application is expected to assert this signal at the appropriate time to indicate its readiness to take in data some clock cycles (L) from now. At L clock later, r_valid shall be asserted if there is data available, or de-asserted if there is no data available at r_data port. This is analogous to the Avalon ST non-zero read latency valid/ready interface semantics and implies enough buffer spaces are allocated in downstream user application to consume in-flight data. • L = 5 when RAM_BLK_TYPE is "MLAB" • L = 6 when RAM_BLK_TYPE is "M20K" • 0 = read request inactive • 1 = read request active
r_data[FIFO_WIDTH-1:0]	Output	Yes	FIFO Read Data. This bus presents the data corresponds to each read request which have taken place some clock cycles earlier on. The read data is only valid in the clock cycle when r_valid asserts.
r_empty	Output	No	FIFO Read Empty. Indicate whether there is still data word remained in the FIFO. This effectively is a pipelined version of r_usedw == 0. This signal may be used by user application for monitoring purpose, or to initiate a series of read requests. • 0 = RAM block is not empty • 1 = RAM block is empty
r_valid	Output	No	FIFO Read Data Valid. Indicate whether the data at r_data output port is valid. Each r_valid assertion correspond to a previous read request/ready. Due to read request to data ready latency introduced by internal pipeline stages, this signal can still be asserted for several clocks after r_empty asserts. When r_valid asserts, data must be taken by the user application; else, it will be lost. The r_valid and r_req interface is analogous to the Avalon ST valid and ready semantics with non-zero read latency. • 0 = data is not valid • 1 = data is valid
w_ready	Output	Yes	Active-low write-protect signal to gate data on the write port, before the delayed w_full asserts.

4.4.7.2. DCFIFO Signals

Table 63. DCFIFO Input and Output Ports Description

Signal	Direction	Required	Description
w_clk	Input	Yes	FIFO Write Clock.
w_aclr	Input	No	Active-high reset signal that feeds the asynchronous clear pins of w_clk domain flip-flops. This reset is not synchronized within the IP, and hence, user logic should ensure it is de-asserted synchronously to w_clk whenever appropriate.

continued...

Signal	Direction	Required	Description
			This signal only takes effect if USE_ACLR_PORT is enabled. • 0 = reset inactive • 1 = reset active
w_sclr	Input	No	Active-high reset signal that feeds the synchronous clear pins of w_clk domain flip-flops. Reset sequence requirements must be followed. • 0 = request inactive • 1 = request active
r_clk	Input	Yes	FIFO Read Clock.
r_aclr	Input	No	Active-high reset signal that feeds the asynchronous clear pins of r_clk domain flip-flops. This reset is not synchronized within the IP, and hence, user logic should ensure it is de-asserted synchronously to r_clk whenever appropriate. In addition, reset sequence requirements must be followed. This signal only takes effect if USE_ACLR_PORT is enabled. • 0 = reset inactive • 1 = reset active
r_sclr	Input	No	Active-high reset signal that feeds the synchronous clear pins of r_clk domain flip-flops. Reset sequence requirements must be followed. • 0 = reset inactive • 1 = reset active
w_req	Input	Yes	FIFO write request. This signal is expected to be inactive during reset. • 0 = request inactive • 1 = request active
w_data[FIFO_WIDTH-1:0]	Input	Yes	FIFO Write Data. This bus presents the data to be stored into the FIFO when there is a write request. The value is taken by the FIFO only when w_req is active, and the FIFO is not full (i.e. w_full = 1).
w_full	Output	No	FIFO Write Full. This signal indicates whether the space remained in the FIFO is about to run-out. When this signal asserts, further w_req is ignored. *Note:* Due to internal pipeline stages to improve Fmax, the actual usable space is a few entries less than that being configured to prevent data loss. • 0 = FIFO is not full • 1 = FIFO is (near) full
r_req	Input	Yes	FIFO Read Request / Read Ready. To achieve the highest possible Fmax, the use model of this signal is slightly different from the normal zero read to data ready latency FIFO. User application is expected to assert this signal at the appropriate time to indicate its readiness to take in data some clock cycles (L) from now. At L clock later, r_valid shall be asserted if there is data available, or de-asserted if there is no data available at r_data port. This is analogous to the Avalon ST non-zero read latency valid/ready interface semantics and implies enough buffer spaces are allocated in the downstream user application to consume in-flight data. • L = 5 when RAM_BLK_TYPE is "MLAB" • L = 6 when RAM_BLK_TYPE is "M20K"

continued...

Signal	Direction	Required	Description
			• 0 = read request inactive • 1 = read request active
r_data[FIFO_WIDTH-1:0]	Output	Yes	FIFO Read Data. This bus presents the data corresponds to each read request, which have taken place some clock cycles earlier. The read data is only valid in the clock cycle when r_valid asserts.
r_empty	Output	No	FIFO Read Empty. Indicate whether there is still data word remained in the FIFO. This effectively is a pipelined version of r_usedw == 0. This signal may be used by user application for monitoring purpose, or to initiate a series of read requests. • 0 = RAM block is not empty • 1 = RAM block is empty
r_valid	Output	No	FIFO Read Data Valid. Indicate whether the data at r_data output port is valid. Each r_valid assertion correspond to a previous read request/ready. Due to read request to data ready latency introduced by internal pipeline stages, this signal can still be asserted for several clocks after r_empty asserts. When r_valid asserts, data must be taken by the user application; else, it will be lost. The r_valid and r_req interface is analogous to the Avalon ST valid and ready semantics with non-zero read latency. • 0 = data is not valid • 1 = data is valid
w_ready	Output	Yes	Active-low write-protect signal to gate data on the write port, before the delayed w_full asserts.

4.4.8. Reset and Clock Schemes

4.4.8.1. Clock Domains

The logic of the FIFO2 Intel FPGA IP core is separated into 2 clock domains internally:

- w_clk
- r_clk

For example, in the default IP setting for a DCFIFO, the 2 clock domains are assumed to be asynchronous with proper clock-crossing structure in place.

You can configure the FIFO2 IP core to operate as a SCFIFO by setting the SCFIFO_MODE parameter to 1. In this mode:

- All revelant clock crossing structure logic are not synthesized.
- Both w_clk and r_clk signals are tied together to the same source and timed synchronously.

4.4.8.2. Reset

To maximize Fmax, there are non-resettable flops (or registers) with undefined initial power and reset states. Unless the reset state of a given interface signal is specified, you must not assume the reset the non-resettable flops to be of a specific value during power up or reset. As part of reset sequence, you must ensure that the FIFO internal stale states are flushed before normal operations are started or resumed.

The FIFO2 Intel FPGA IP core exposes both asynchronous and synchronous clear ports per clock domain so that the user application has full control on how reset sequences, such as entering and exiting reset, should work. The clear events for both `w_clk` and `r_clk` clock domains come from the same source so that the logic in both domains are brought into or out of reset together nominally. For example, you can choose to reset the logic in one clock domain such as `r_clk`, instead of `w_clk`. However, some signals such as the FIFO fill level status take time to settle down to the right states. In this case, the user application must ensure those signals do not cause any unintentional side effect.

By default, the FIFO2 Intel FPGA IP core only samples synchronous clear but ignores the asynchronous clear ports. You have the option to turn on **Enable Asynchronous Clear (ACLR)** to enable the synchronous clear feature in the parameter editor of the FIFO2 IP cores. You can also choose to implement only the asynchronous clear reset scheme by tying the synchronous clear ports to their inactive states.

Note: Non-resettable registers within the IP core require state flushing, even when asynchronous clear ports are used.

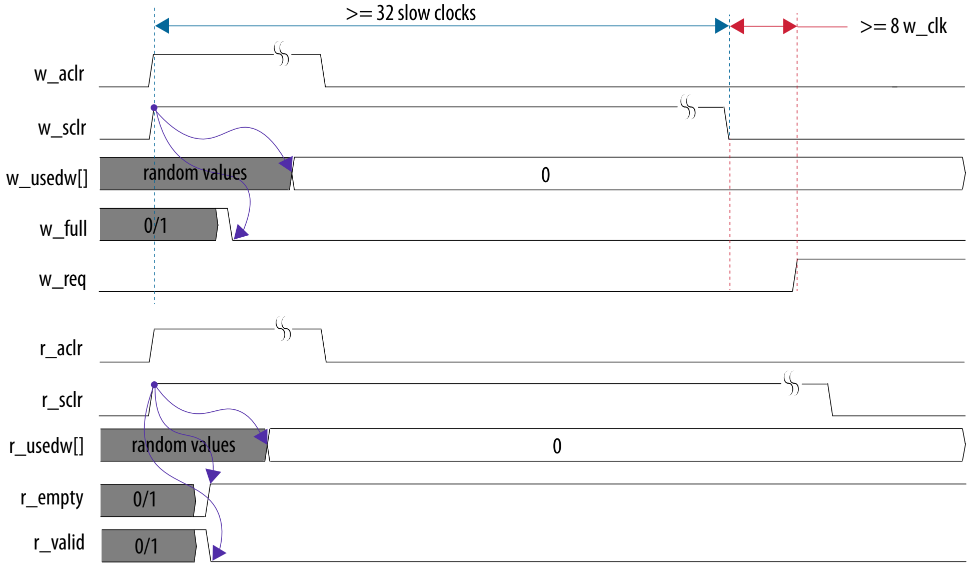
4.4.8.2.1. FIFO2 Intel FPGA IP Reset Guidelines

Use the following guidelines to provide a proper reset to the FIFO Intel FPGA IP core:

- Asynchronous clear is treated as a global IP reset event, and has the highest priority.
- If both asynchronous clear and synchronous clear are implemented:
  - When asynchronous clear asserts, the associated synchronous clear (for the clock domain) must also be asserted.
  - Asynchronous clear must be deasserted first before synchronous clear (for the clock domain) deasserts. Use synchronous clear to control when the IP should be out of reset.
  - The asynchronous clear duration may be as short as 1 clock, but the synchronous clear must last for at least 32* slow clock cycles (clock must be toggling) to ensure all IP internal stale states are flushed.
- If only asynchronous or synchronous clear is implemented, the clear assertion duration must last for at least 32* slow clock cycles (clock must be toggling) to ensure all IP internal stale states are flushed.
- All clocks must be toggling valid for some time before asynchronous or synchronous clear deassertion.
- As some reset signals are internally pipelined, write operations must not be started within 8* clocks after reset deassertion.

Figure 49. Reset Behavior

This figure shows the reset behavior of the FIFO2 Intel FPGA IP core.



Notes:

1. If used, assert w_aclr and r_aclr signals nominally at the same time.
2. Assert the associated *_sclr signal when when *_aclr signal is asserted.
3. Clocks must toggle the valid data for more than 32 slow clocks before *_sclr de-assertion to make sure that stale states are flushed.

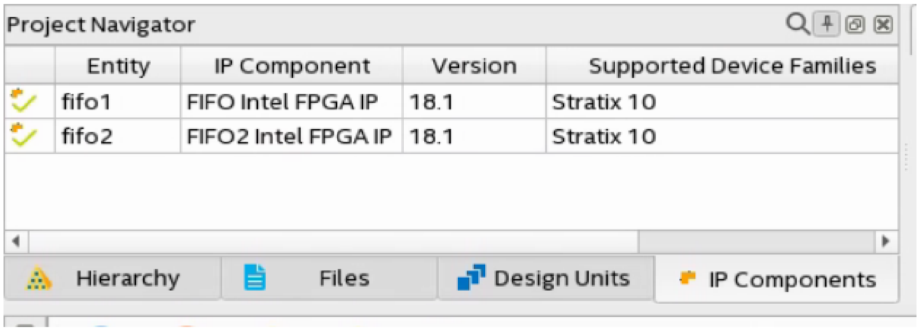
5. Intel Stratix 10 Embedded Memory Design Example

5.1. FIFO and FIFO2 Design Example

You can use this design example as a reference on how to instantiate the FIFO and FIFO2 Intel FPGA IP cores and what behavior to expect in a simulation.

5.1.1. Generating the Design Example

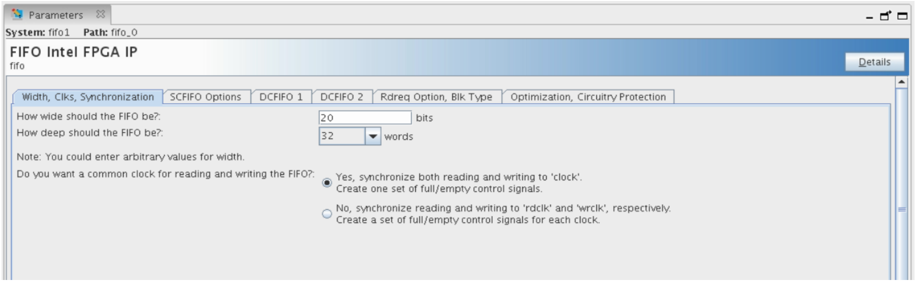
1. Download the design example from Design Store.
2. Using the Intel Quartus Prime Pro Edition software, restore the file by selecting **Open Project** and select the `.par` file. Click **OK** to load the project
3. Once the project is successfully loaded, go to **IP component** tab in **Project Navigator** pane. Double-click the FIFO Intel FPGA IP core (`fifo1`) to open the IP Parameter Editor to examine the IP configuration and regenerate the FIFO IP files.



	Entity	IP Component	Version	Supported Device Families
✓	fifo1	FIFO Intel FPGA IP	18.1	Stratix 10
✓	fifo2	FIFO2 Intel FPGA IP	18.1	Stratix 10

4. In the IP Parameter Editor window, ensure that following parameters are set correctly:

Parameter	Value
How wide should the FIFO be?	20 bits
How deep should the FIFO be?	32 words
Read and Write Clock	Single clock
Signals	full, empty, usedw[]
Use Asynchronous Clear	Yes
Use Synchronous Clear	Yes
Memory Block Type	MLAB
FIFO mode	Normal



System: fifo1 Path: fifo_0

FIFO Intel FPGA IP
fifo

How wide should the FIFO be?: 20 bits

How deep should the FIFO be?: 32 words

Note: You could enter arbitrary values for width.

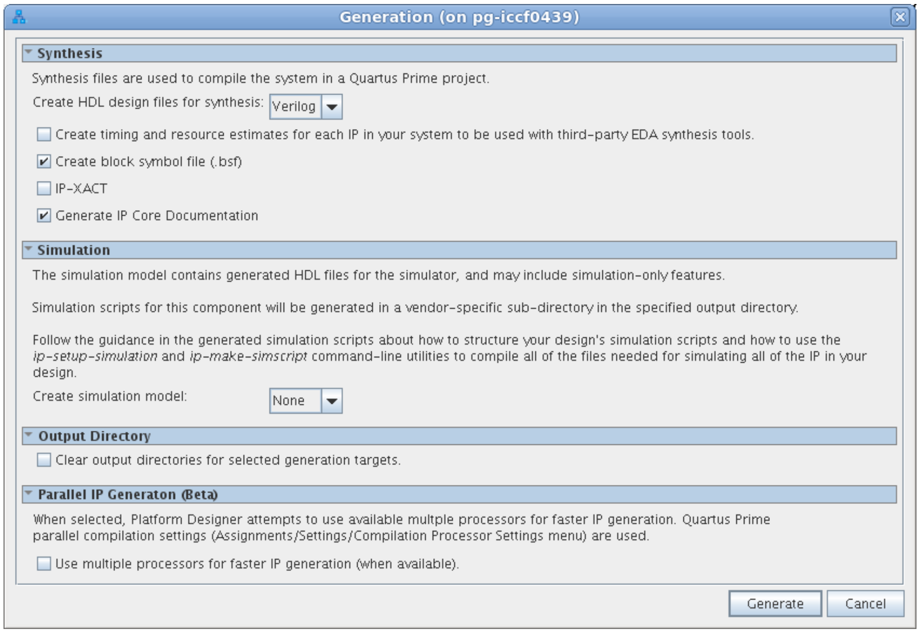
Do you want a common clock for reading and writing the FIFO?:

● Yes, synchronize both reading and writing to 'clock'.
Create one set of full/empty control signals.

○ No, synchronize reading and writing to 'rdclk' and 'wrclk', respectively.
Create a set of full/empty control signals for each clock.

5. To generate HDL files for this IP core, click **Generate HDL**. The **Generation** dialog box appears.

6. Specify output file generation options, and then click **Generate**. The synthesis and simulation files generate according to your specifications.



Synthesis

Synthesis files are used to compile the system in a Quartus Prime project.

Create HDL design files for synthesis: Verilog

□ Create timing and resource estimates for each IP in your system to be used with third-party EDA synthesis tools.

☑ Create block symbol file (.bsf)

□ IP-XACT

☑ Generate IP Core Documentation

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model: None

Output Directory

□ Clear output directories for selected generation targets.

Parallel IP Generaton (Beta)

When selected, Platform Designer attempts to use available multple processors for faster IP generation. Quartus Prime parallel compilation settings (Assignments/Settings/Compilation Processor Settings menu) are used.

□ Use multiple processors for faster IP generation (when available).

7. Repeat steps 3 to 6 for FIFO2 Intel FPGA IP core (`fifo2`) to examine the IP configuration and regenerate the FIFO2 IP files.

8. Once the HDL file generation for both FIFO and FIFO2 Intel FPGA IP is completed, click **Tools ➤ Generate Simulator Setup Script for IP** to generate a combined simulator setup script that automatically source all the required library files for the FIFO and FIFO2 IP simulation. Use the default directory and click **OK** to generate the file.

Related Information

Getting Started with the Design Store

5.1.2. Simulating the Design Example

1. Launch ModelSim to proceed with simulation. In ModelSim, change directory to the path where you saved this project by going to <file>-<change directory>.
2. In the Transcript window, input the following command to start simulation.

```
source sim_top.tcl
```

The Tcl script compiles all the source files and library files, and start the simulation.

5.1.2.1. Simulation Results

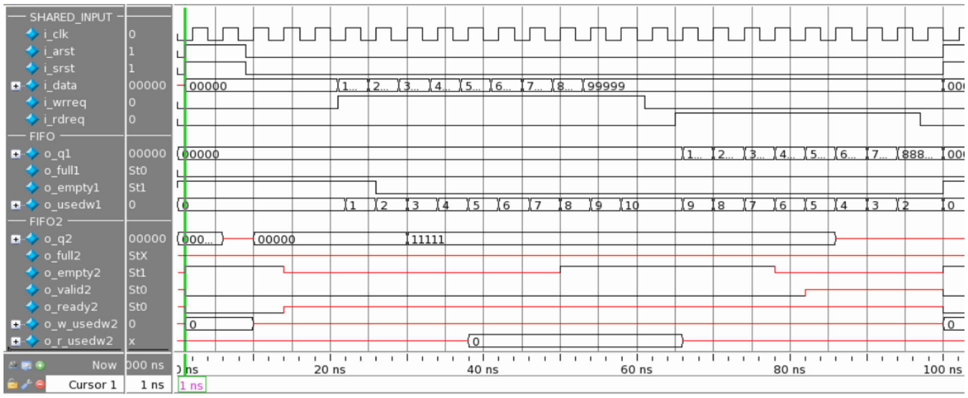
The simulation result of the design example shows the comparison of behavior between FIFO and FIFO2 Intel FPGA IP cores.

Note: The signal names used in this design example for both FIFO and FIFO2 Intel FPGA IP cores are defined with reference to the FIFO input and output ports.

Part 1 (0 - 100 ns)

This section of the simulation shows the read-after-write operation with an invalid reset condition for FIFO2. FIFO2 requires a reset period of 32 cycles. As a result, some signals cannot return to their known states with only two clock cycles of reset. This causes FIFO2 unable to operate as expected. In contrast, FIFO demonstrates a correct write-then-read operation because FIFO only requires a minimum of one clock cycle of reset.

Figure 50. Part 1—Read-after-write Operation of FIFO



Part 2 (100 - 480 ns)

This section of the simulation shows FIFO2 operation with a valid reset scheme, followed by a write-then-read operation. After 32 cycles of reset, all signals return to known states. An 8 cycles of wait is also required before any operation.

Figure 51. Part 2—Valid Reset Scheme for FIFO2

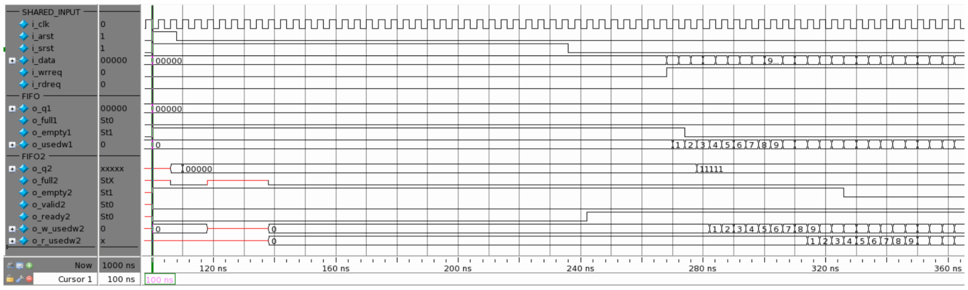
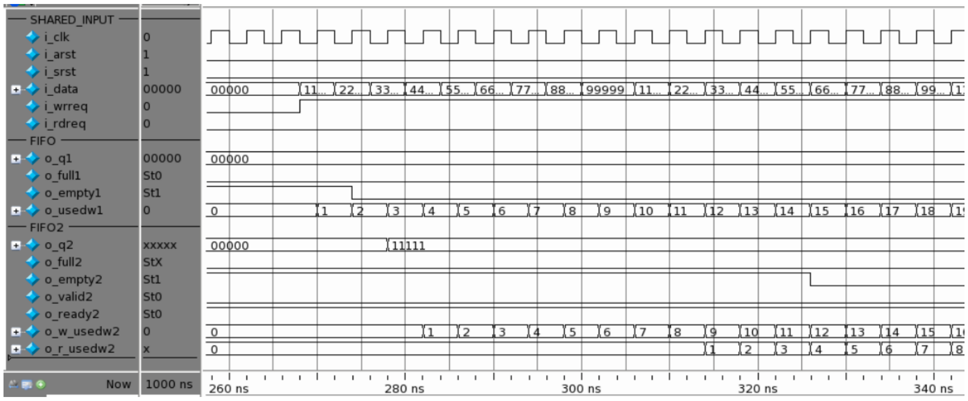


Figure 52. Write Operation

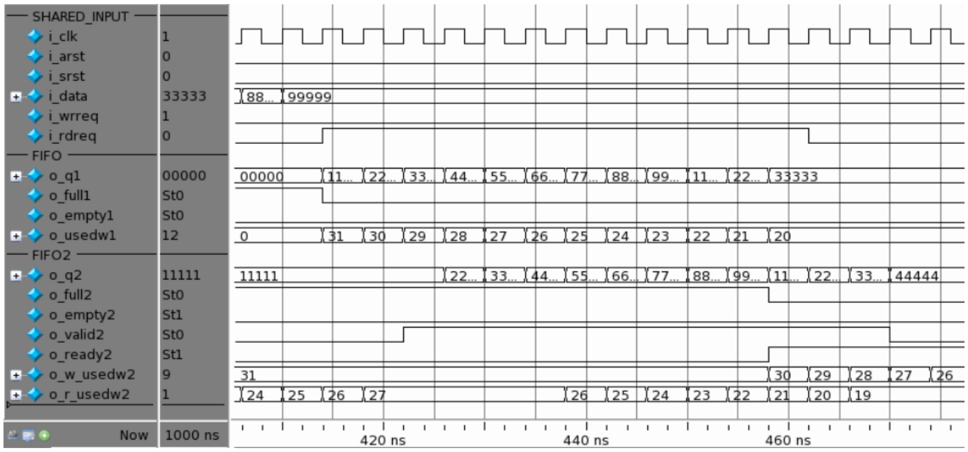


Notes to Figure 52 on page 116:

- **usedw signal**: After the write request signal (e.g., i_wrreq) is asserted, the usedw signal of FIFO (e.g., o_usedw1) starts counting as data is being stored into FIFO. For FIFO2, it has usedw signal on both write (e.g., o_w_usedw2) and read (o_r_usedw2) sides. The write usedw starts counting after 3 cycles of delay, and it takes another 8 cycles to update the read usedw signal.

- **Full and empty signals**: The FIFO full signal (e.g., o_full1) asserts high after the memory of FIFO is full. The FIFO2 full signal (e.g., o_full2) is asserted high on the same cycle as the o_w_usedw2 signal, which indicates that the memory is full. Note that when the three cycles before the full signal is asserted, the ready signal is already asserted low, which is an expected pipeline behavior of FIFO2. The FIFO empty signal (e.g., o_empty1), which has only one clock cycle of latency, asserts low after the first data is being written into memory. In contrast, FIFO2 takes three cycles of delay to indicate that its empty signal (e.g., o_empty2) has turned not-empty, after the o_r_usedw2 signal is risen to 1.

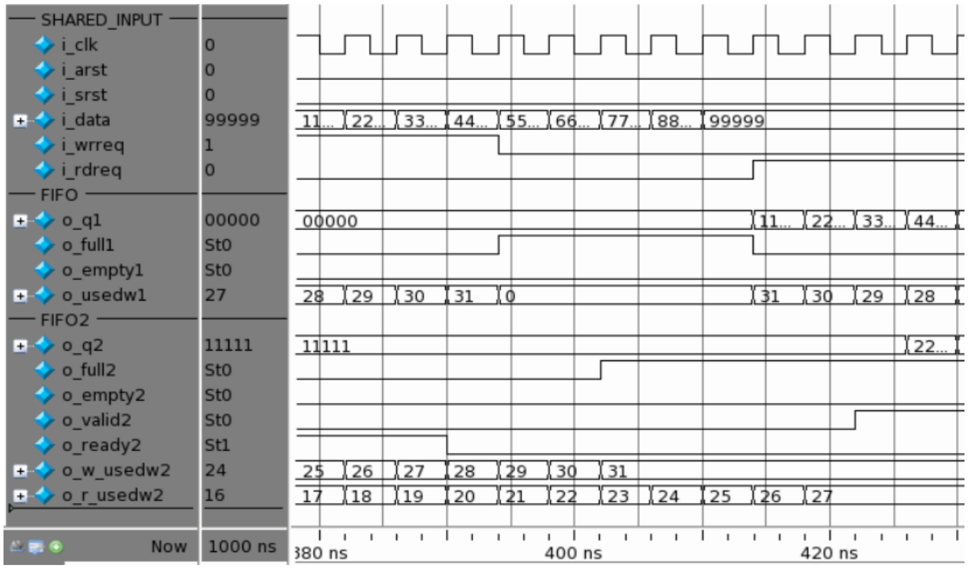
Figure 53. Read Operation



Notes to Figure 53 on page 117:

- On the read operation, the FIFO o_usedw1 signal is reflected on same positive clock edge with the read request signal. For FIFO2, two cycles of latency is observed before it is reflected on the o_r_usedw2 signal. It also takes another 9 cycles of propagation from o_r_usedw2 to o_w_usedw2.

Figure 54. FIFO Full Condition



Notes to Figure 54 on page 117:

- **Ready signal**: After a valid reset scheme, the FIFO2 ready signal (e.g., o_ready2) asserts high, which indicates that it is ready for the write operation.

  Note: FIFO does not have this signal.
- **Valid signal:** The FIFO2 valid signal (e.g., o_valid2) is to indicate that the data being read out is valid. In the case of a valid read operation, the o_valid2 signal will assert high after two cycles of delay, with the assertion of the read request (e.g., i_rdreq) signal.

  Note: FIFO does not have this signal.

Part 3 (480 - 700 ns)

This section of the simulation showcases a valid reset condition, followed by a read-during-write operation. After reset, the write request signal (e.g., i_wrreq) asserts high. After a few cycles later, the i_rdreq signal also asserts high. For FIFO, the number of used word maintains while the data is being read out, which follows an expected behavior. Note that in this case, FIFO2 has no data being read out because the o_empty2 signal is not asserted low. Therefore, it is not a valid read operation for FIFO2.

Part 4 (700 ns and onwards)

This section of the simulation observes the behaviour of FIFO and FIFO2 under different scenario. When the FIFO2 o_r_usedw2 signal counts down to 2, the o_empty2 signal asserts high and undergoes an internal check. After two cycles, the o_empty2 signal goes back to a not-empty state after the status is validated. This mechanism is specially designed for FIFO2 according to its multiple pipeline nature.

6. Intel Stratix 10 Embedded Memory User Guide Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
18.1	Intel Stratix 10 Embedded Memory User Guide
18.0	Intel Stratix 10 Embedded Memory User Guide
17.1	Intel Stratix 10 Embedded Memory User Guide
17.0	Intel Stratix 10 Embedded Memory User Guide

7. Document Revision History for the Intel Stratix 10 Embedded Memory User Guide

Document Version	Intel Quartus Prime Version	Changes
2020.08.03	20.1	• Updated Table: *Embedded Memory Capacity and Distribution in Intel Stratix 10 Devices* to add resource counts for device GX 1660, GX 2110, GX 10M, TX 850, DX 1100, DX 2100, and DX2800. • Added a new topic to the *Intel Stratix 10 Embedded Memory Design Considerations* section—*Including the Reset Release Intel FPGA IP in Your Design*. • Updated the *True Dual Port Dual Clock Emulator* section. • Updated Figure: *Mixed-Port Read-During-Write: New_a_old_b Mode*. • Updated the *On Chip Memory RAM and ROM Intel FPGA IPs* section. • Updated the descriptions for the following parameters in Table: *RAM: 1-PORT Intel FPGA IP Parameters*: — **Create an 'aclr' asynchronous clear for the registered ports** — **Create an 'sclr' synchronous clear for the registered ports** • Updated the description for `clock0` in Table: *Interface Signals of the Intel Stratix 10 RAM and ROM Intel FPGA IP Cores*.
2019.11.19	19.1	• Updated Table: *Mixed Port Read-During-Write Output Behaviors*: — Updated the Output Data when Read-During-Write value of the `constrained_dont_care` and `dont_care` parameters from "New data" to "Don't care". — Added a footnote to state that the output data is "don't care" because the IP does not guarantee metastability for the output data when read-during-write. • Updated the *FIFO Intel FPGA IP* section.
2019.10.17	19.1	• Added new Topic—*Avoid Providing Non-Deterministic Input*. • Added a note to the *RAM and ROM Interface Signals* topic. • Updated the description for `refclk` signal in Table: *eSRAM Intel FPGA IP Input and Output Signals*. • Updated Table: *Intel Stratix 10 Memory IP Cores*. • Updated Table: *RAM: 2-PORT Intel FPGA IP Parameter Settings*: — Updated the legal values and description for **What clocking method would you like to use?**. — Updated the description for **Enable Error Correction Check (ECC)**. • Updated Table: *Parameters for altera_syncram*.
2019.08.15	19.1	• Updated the legal values and description for the **What should the 'q' output be when reading from a memory location being written to?** parameter in Table: *RAM: 1-PORT Intel FPGA IP Parameters Description*. • Updated the description for the **Get x's for write masked bytes instead of old data when byte enable is used** parameter in Table: *RAM: 1-PORT Intel FPGA IP Parameters Description*.

continued...

Document Version	Intel Quartus Prime Version	Changes
2019.04.01	19.1	• Updated the *Byte Enable in Intel Stratix 10 Embedded Memory Blocks* topic. • Updated Table: *Byte Enable Controls in ×40 Data Width (M20K)*. • Updated the *Customize Read-During-Write Behavior* topic. • Updated the description for M20K blocks in the *Memory Blocks Error Correction Code Support* topic. • Updated Table: *RAM: 2-PORT Intel FPGA IP Parameter Settings* to add notes to the **Use clock enable for output registers** (Clock Enables) and **q_a port** (Sclr Options) options. • Updated Table: *RAM: 1-PORT Intel FPGA IP Parameters Settings*: — Added notes to the **Use clock enable for output registers** (Clock Enables) and **q_a port** (Sclr Options) options. — Updated the legal values and description for the **Set the maximum block depth to** parameter. • Updated Table: *RAM: 2-PORT Intel FPGA IP Parameters Settings* to update the legal values for the **Set the maximum block depth to** parameter. • Updated Table: *RAM: 4-PORT Intel FPGA IP Parameters Settings* to update the legal values for the **Set the maximum block depth to** parameter. • Made editorial and typographical updates throughout the document.
2018.12.24	18.1	• Added a FIFO and FIFO2 Simulation Design Example section. • Updated the note in the *True Dual Ports Dual Clock Emulator* topic. • Added a note to *Consider the Memory Block Selection* topic. • Updated the *Changing Parameter Settings Manually* topic. • Made minor topic restructuring to the *Intel Stratix 10 Embedded Memory Architecture and Features* and the *On Chip Memory RAM and ROM Intel FPGA IP Cores* sections.
2018.10.24	18.1	• Added a new topic: *Initial Value of Read and Write Address Registers*. • Updated *True Dual Ports Dual Clock Emulator* topic: — Updated the description for this topic. — Updated Table: *Differences between Intel Arria 10 TDP Dual Clock Mode and Intel Stratix 10 Emulated TDP Dual Clock Mode* to correct the device support for `sclr`. — Updated the following figures: • *Output Condition of Port A* • *Output Condition of Port B* • *Read-During-Write Condition of Port A* • *Read-During-Write Condition of Port B* • Renamed topic title *Hardware Behavior* to *Consider the Concurrent Read Behavior*. • Updated the following tables: — *Intel Stratix 10 Embedded Memory Features* — *Embedded Memory Capacity and Distribution in Intel Stratix 10 Devices* — *RAM: 2-PORT Intel FPGA IP Parameter Settings* — *RAM: 4-PORT Intel FPGA IP Parameter Settings* — *ROM: 1-PORT Intel FPGA IP Parameter Settings* — *ROM: 2-PORT Intel FPGA IP Parameter Settings* — *Interface Signals of the Intel Stratix 10 RAM and ROM Intel FPGA IP Cores* • Made minor editorial updates throughout the document.

Document Version	Intel Quartus Prime Version	Changes
2018.05.07	18.0	• Updated the following IP cores as per Intel rebranding: — "RAM: 1-PORT" IP core to "RAM: 1-PORT Intel FPGA IP" — "RAM: 2-PORT" IP core to "RAM: 2-PORT Intel FPGA IP" — "RAM: 4-PORT" IP core to "RAM: 4-PORT Intel FPGA IP" — "ROM: 1-PORT" IP core to "ROM: 1-PORT Intel FPGA IP" — "ROM: 2-PORT" IP core to "ROM: 2-PORT Intel FPGA IP" — "Intel Stratix 10 Native eSRAM" IP core to "eSRAM Intel FPGA IP" — "FIFO" IP core to "FIFO Intel FPGA IP" — "FIFO2" IP core to "FIFO2 Intel FPGA IP" • Added new topics: — *ECC Read-During-Write Behavior* — *Forwarding Logic* • Updated Table: *Intel Stratix 10 Embedded Memory Features*: — Added Force-to-Zero support information — Removed packed mode feature. • Updated Table: *Embedded Memory Capacity and Distribution in Intel Stratix 10 Devices* to remove redundant table content on Intel Stratix 10 MX1650 and MX2100. • Updated the *Memory Blocks Error Correction Code Support* topic: — Updated the description for the ECC feature. — Updated the ECC status flag signals for eSRAM blocks • Updated the *ECC Parity Flip* topic to correct the parity bit sequence for double-adjacent-error correction. • Updated the *Error Correction Code Truth Table* topic: — Updated Figure: *ECC Block Diagram for M20K Memory*. — Update Table: *ECC Status Flags Truth Table for eSRAM*. • Updated the *Force-to-Zero* topic. • Updated the *Coherent Read Memory* topic: — Renamed topic title from *Coherent Read* to *Coherent Read Memory*. — Added new Figures: *Coherent Read with Unregistered Output* and *Coherent Read with Registered Output*. — Removed Figures: *1-level Pipelining Waveform* and *2-level Pipelining Waveform*. • Updated Table: *Memory Blocks Clocking Modes Supported for Each Memory Mode* to add a footnote for the read/write clock mode mode of true-dual-port mode. • Updated the *Mixed-Width Port Configurations:* — Added Table: *Supported Mixed-width Ratios for Intel Stratix 10.* • Removed Topic: *Mixed-Width Ratio Configurations*. • Updated the *True Dual Ports Dual Clock Emulator* topic: — Updated topic description to include information on *valid* signal. — Added new Figures: • *Output Condition of Port A* • *Output Condition of Port B* • *RDW Condition of Port A* • *RDW Condition of Port B*
		• Updated the *Intel Stratix 10 Embedded Memory Configurations* topic: — Updated Table: *Supported Embedded Memory Block Configurations* to correct the depth and programmable width for eSRAM. — Removed the note about Intel Stratix 10 devices do not natively support 1/32, 1/16, and 1/8 mixed-width port ratios. • Updated the *Consider Power-Up State and Memory Initialization* topic. • Updated Table: *Output Modes for Embedded Memory Blocks in Same-Port Read-During-Write Mode* to add a note include a note for *Don't Care* mode.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• Added a Table: *Mixed Port Read-During-Write Output Behaviors*. • Updated the *RAM and ROM Intel FPGA IP Core* chapter: — Added *Changing Parameter Settings Manually* and *RAM and ROM Parameters* subtopics. — Updated Tables: • *RAM: 1-PORT Intel FPGA IP Core Parameter Settings* • *RAM: 2-PORT Intel FPGA IP Core Parameter Settings* • *RAM: 4-PORT Intel FPGA IP Core Parameter Settings* • *ROM: 2-PORT Intel FPGA IP Core Parameter Settings* • *ROM: 2-PORT Intel FPGA IP Core Parameter Settings* • *Interface Signals of the RAM and ROM Intel FPGA IP Cores* • Updated the *eSRAM Intel FPGA IP* — Updated Table: *eSRAM Specifications*: • Added a footnote to the clock frequency feature. • Corrected the clock frequency value of -2 speed grade from 200 MHz - 650 MHz to 200 MHz - 640 MHz. • Updated the write latency value from 0 to 0 + 1. • Added a footnote to the write latency feature. — Updated Table: *eSRAM Intel FPGA IP Core Parameter Editor: Channel Tab*. — Updated Table: *eSRAM Intel FPGA IP Core Input and Output Signals*: • Added a new interface signal—`iopll_lock2core`. • Updated the width of the `esram2f_clk` signal from 2 to 1. • Updated the description of the `esram2f_clk` signal. • Updated the width of `c<channel_number>_data_0` signal from '72 or 64' to '1-72'. — Updated the *eSRAM Intel FPGA IP Simulation Walkthrough* topic. • Updated *FIFO Intel FPGA IP* chapter: — Added *Reset Scheme* subtopic. • Updated the *FIFO2 Intel FPGA IP* chapter: — Updated Table: *Differences between FIFO and FIFO2 Intel FPGA IP Cores* to remove the reset scheme feature. — Updated Table: *FIFO2 Specifications*: • Added a footnote for M20K of the Error Checking and Correcting (ECC) feature. • Updated the description for MLAB of the Targeted Performance feature. — Renamed topic title *FIFO2 User-Configurable Parameters* to *FIFO2 Parameter Settings*. — Updated Figure: *FIFO2 IP Core Input and Output Signals*. — Updated Tables: *SCFIFO Input and Output Ports Description* and *DCFIFO Input and Output Ports Description* to include descriptions for `w_ready` signal. • Updated for latest Intel branding standards. • Made editorial updates throughout the document.

Date	Version	Changes
December 2017	2017.12.04	Updated the "Embedded Memory Capacity and Distribution in Intel Stratix 10 Devices" table: Corrected the total RAM Bit (Mbits) for Intel Stratix 10 GX, Intel Stratix 10 MX, and Intel Stratix 10 SX variants.
November 2017	2017.11.06	• Added a new feature—True Dual Ports Dual Clock Emulator. • Updated the *Intel Stratix 10 Embedded Memory Features* topic: Updated the number of banks for each channel in the eSRAM blocks from 40 banks to 42 banks. • Updated the "Intel Stratix 10 Embedded Memory Features" table: — Updated the description for eSRAM for the Mixed-port read-during-write and coherent read features. — Added freeze logic, hardware behavior, and TDP dual clock emulator features. • Updated the "Embedded Memory Capacity and Distribution in Intel Stratix 10 Devices" table: — Updated eSRAM block and RAM (Bit) values for Intel Stratix 10 GX and Intel Stratix 10 SX variants. — Added embedded memory capacity information for Intel Stratix 10 MX variant. — Updated the values of M20K and MLAB RAM Bits, and total RAM bits for TX1650 and TX2100 product lines for Intel Stratix 10 TX variant. • Updated the *Byte Enable in Intel Stratix 10 Embedded Memory Blocks* topic. • Updated the *Data Byte Output* subtopic. • Updated the *Asynchronous Clear and Synchronous Clear* topic: — Updated the topic description. — Updated Figures: "Behavior of Asynchronous Clear and Synchronous Clear In Registered Mode" and "Behavior for Asynchronous Clear and Synchronous Clear In Unregistered Mode". • Updated the *Memory Blocks Error Correction Code Support* topic: — Added a feature for memory blocks error correction code support—ECC Parity Flip. — Updated the description of the eSRAM Blocks. • Added "ECC Status Flags Truth Table for eSRAM" table in the *Error Correction Code Truth Table* subtopic. • Updated the *Embedded Memory Operating Modes* topic; — Renamed topic as *Intel Stratix 10 Embedded Memory Supported IP Cores*. — Updated "Intel Stratix 10 Memory IP Cores" table: Added IP Core column and information for ROM: 2 PORT. • Updated the "Memory Blocks Clocking Modes Supported for Each Memory Mode" table: — Added Dual-Port ROM memory mode. — Added input/output clock mode support for True Dual-Port. — Removed FIFO memory mode. • Updated the note for the simple dual-port mode in the *Mixed-Width Ratio Configuration* topic. • Updated the "RAM in Mixed-Port Read-During-Write Mode" table: — Added a note to the *Don't Care* description for the *Don't Care* mode. — Added *New_a_old_b* mode to the table. — Added new Figure—Mixed-Port Read-During-Write: *New_a_old_b* Mode. • Updated the RAM: 1-PORT and RAM: 2-PORT IP cores topics in the *On-Chip Memory RAM and ROM IP Cores* section. • Updated the "RAM: 2-Port Parameter Setting" table: Added the Emulate TDP dual clock mode option.

continued...

Date	Version	Changes
		• Updated the "Interface Signals of the Intel Stratix 10 On-Chip Memory RAM and ROM IP Cores" table: — Updated the direction values for `eccencbypass` and `eccencparity` signals. — Added three signals—`address2_a`, `address2_b`, and `sclr`. — Removed four signals: `clocken2`, `clocken3`, `aclr0`, and `aclr1`. — Updated the description for `aclr` signal. • Renamed the topic *Intel Stratix 10 eSRAM IP Core* to *Intel Stratix 10 Native eSRAM IP Core* to align with Intel Quartus Prime naming. • Added eSRAM IP core references to the *Intel Stratix 10 Native eSRAM IP Core* topic. • Added FIFO IP core references to the *FIFO IP Core* topic. • Added FIFO2 IP core references to the *FIFO2 IP Core* topic. • Updated for latest branding standards. • Made editorial updates throughout the document.
May 2017	2017.05.08	• Removed parity bit support for MLAB blocks under the Error Correction Code (ECC) support feature in the Intel Stratix 10 Embedded Memory Features table. • Updated the descriptions for M20K and MLAB blocks under Error Correction code (ECC) support feature in the Intel Stratix 10 Embedded Memory Features table. • Updated the Embedded Memory Capacity and Distribution in Intel Stratix 10 Devices table to remove TX4500 and TX5500, which are no longer part of Intel Stratix 10 TX variant. • Updated the Byte Enable Controls in ×10 Data Width (MLAB) table. • Removed parity bit support for MLAB blocks in the Parity Bit topic. • Added notes to the Supported Embedded Memory Block Configurations table in the Intel Stratix 10 Embedded Memory Configurations topic. • Added Mixed-Width Ratio Configurations topic. • Added Freeze Logic topic. • Added the Implement clock-enable circuitry for use in a partial reconfiguration region option for the RAM: 1-PORT, RAM: 2-PORT, and RAM: 4-PORT IP cores. • Removed the Use different data widths on different ports option from RAM: 4-Port Parameter Settings table because this option is not available in RAM: 4-Port. • Added Hardware Behavior topic. • Added figures for the Coherent Read topic. • Updated the feature description for ROM: 1-PORT and ROM: 2-PORT in the table of the On-Chip Memory RAM and ROM IP Cores topic. • Added `ecc_enc_bypass` and `ecc_enc_parity` signals in the Interface Signals of the Intel Stratix 10 On-Chip Memory RAM and ROM IP Cores table. • Added Intel Stratix 10 eSRAM IP core topic. • Minor typographical corrections.
October 2016	2016.10.31	Initial release