SGDMA Driver API Reference

Technical Note

FPGA-TN-02340-1.2

July 2024

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Contents....3
Acronyms in This Document....6
1. Introduction....7
1.1. Purpose....7
1.2. Audience....7
1.3. Driver Versioning....7
1.3.1. Driver Version....7
1.4. Driver and IP Compatibility....7
2. API Description....8
2.1. sgdma_init()....8
2.2. sgdma_reset()....8
2.3. sgdma_irq_mask()....8
2.4. get_sgdma_reg_status()....8
2.5. get_mm2s_bd_status()....9
2.6. get_s2mm_bd_status()....9
2.7. mm2s_get_desc_complete_bit()....9
2.8. mm2s_buf_desc_dma()....9
2.9. s2mm_get_desc_complete_bit()....10
2.10. s2mm_buf_desc_dma()....10
2.11. sgdma_register_read()....10
2.12. sgdma_register_write()....10
3. Function Call Flow Diagrams....11
3.1. sgdma_init()....11
3.2. sgdma_reset()....11
3.3. sgdma_irq_mask()....12
3.4. get_sgdma_reg_status()....12
3.5. get_mm2s_bd_status()....13
3.6. get_s2mm_bd_status()....13
3.7. mm2s_get_desc_complete_bit....14
3.8. mm2s_buf_desc_dma()....15
3.9. s2mm_get_desc_complete_bit....16
3.10. s2mm_buf_desc_dma()....17
3.11. sgdma_register_read()....18
3.12. sgdma_register_write()....19
4. API Data Structure and Enums....20
4.1. struct mm2s_ctrl_reg_t....20
4.2. struct mm2s_sts_reg_t....20
4.3. struct mm2s_desc_tx_t....20
4.4. struct mm2s_desc_tx_ext_t....21
4.5. union mm2s_desc_t....21
4.6. struct s2mm_ctrl_reg_t....21
4.7. struct s2mm_sts_reg_t....21
4.8. struct s2mm_desc_tx_t....22
4.9. struct s2mm_desc_tx_ext_t....22
4.10. union s2mm_desc_t....22
4.11. struct sgdma_instance_t....23
4.12. struct sgdma_ctrl_reg_t....23
4.13. struct sgdma_sts_reg_t....23
4.14. enum control_type_t....24
4.15. enum status_type_t....24
4.16. enum irq_mask_t....24
4.17. enum sgdma_reg_type_t....24

5. API Variables 25
6. API Macros 26
References 27
Technical Support Assistance 28
Revision History 29

Figures

Figure 3.1. sgdma_init() 11
Figure 3.2. sgdma_reset() 11
Figure 3.3. sgdma_irq_mask() 12
Figure 3.4. get_sgdma_reg_status() 12
Figure 3.5. get_mm2s_bd_status() 13
Figure 3.6. get_s2mm_bd_status() 13
Figure 3.7. mm2s_get_desc_complete_bit 14
Figure 3.8. mm2s_buf_desc_dma() 15
Figure 3.9. s2mm_get_desc_complete_bit 16
Figure 3.10. s2mm_buf_desc_dma() 17
Figure 3.11. sgdma_register_read() 18
Figure 3.12. sgdma_register_write() 19

Tables

Table 1.1. Driver and IP Compatibility 7
Table 1.2. Quick Facts of Driver Tested Environment 7
Table 2.1. sgdma_init() 8
Table 2.2. sgdma_reset() 8
Table 2.3. sgdma_irq_mask() 8
Table 2.4. get_sgdma_reg_status() 8
Table 2.5. get_mm2s_bd_status() 9
Table 2.6. get_s2mm_bd_status() 9
Table 2.7. mm2s_get_desc_complete_bit() 9
Table 2.8. mm2s_buf_desc_dma() 9
Table 2.9. s2mm_get_desc_complete_bit() 10
Table 2.10. s2mm_buf_desc_dma() 10
Table 2.11. sgdma_register_read() 10
Table 2.12. sgdma_register_read() 10
Table 4.1. struct mm2s_ctrl_reg_t 20
Table 4.2. struct mm2s_sts_reg_t 20
Table 4.3. struct mm2s_desc_tx_t 20
Table 4.4. struct mm2s_desc_tx_ext_t 21
Table 4.5. union mm2s_desc_t 21
Table 4.6. struct s2mm_ctrl_reg_t 21
Table 4.7. struct s2mm_sts_reg_t 21
Table 4.8. struct s2mm_desc_tx_t 22
Table 4.9. struct s2mm_desc_tx_ext_t 22
Table 4.10. union s2mm_desc_t 22
Table 4.11. struct sgdma_instance_t 23
Table 4.12. struct sgdma_ctrl_reg_t 23
Table 4.13. struct sgdma_sts_reg_t 23
Table 4.14. enum control_type_t 24
Table 4.15. enum status_type_t 24
Table 4.16. enum irq_mask_t 24
Table 4.17. enum sgdma_reg_type_t 24
Table 5.1. API Variables 25
Table 6.1. API Macros 26

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
API	Application Programming Interface
DMA	Direct Memory Access
MM2S	Memory Map to Stream
S2MM	Stream to Memory Map
SGDMA	Scatter-Gather Direct Memory Access

1. Introduction

Scatter gather direct memory access (SGDMA) refers to the ability to collect data from multiple non-contiguous memory locations and then "scatter" the data to different destinations or vice versa, where data from different sources are gathered and written to non-contiguous memory locations. SGDMA can be used to receive input or store outputs in the memory.

For more information on the SGDMA IP, refer to the SGDMA Controller IP Core - Lattice Radiant Software User Guide (FPGA-IPUG- 02131).

1.1. Purpose

This document is a reference guide for developers that provides details of the C language driver APIs, function calls, and flow charts.

1.2. Audience

The intended audience for this document includes embedded system designers and embedded software developers using CertusPro™-NX devices.

1.3. Driver Versioning

1.3.1. Driver Version

SGDMA version 24.01.00.

1.4. Driver and IP Compatibility

Table 1.1. Driver and IP Compatibility

Driver Version	IP Version
24.01.00	2.2.0

Table 1.2. Quick Facts of Driver Tested Environment

Driver tested on HW devices	Tool Version
CertusPro-NX Versa Board	Lattice Propel™ Builder 2024.1
	Lattice Propel SDK 2024.1
	Lattice Radiant™ Software 2024.1
	Radiant Programmer 2024.1

2. API Description

2.1. sgdma_init()

This API initializes the buffer descriptor base address and the sgdma IP base address.

```
void sgdma_init(sgdma_instance_t * this_sgdma, unsigned int base_addr)
```

Table 2.1. sgdma_init()

In/Out	Parameter	Description	Returns
In/out	*this_sgdma	This structure is initialized with the SGDMA IP address, num_of_desc , and the information of the s2mm and mm2s buffer descriptor addresses.	None
In	base_addr	Base address that specifies the SGDMA IP base address.	
In	num_of_desc	Specifies the buffer descriptor number.	

2.2. sgdma_reset()

This API resets the MM2S and S2MM control register.

```
void sgdma_reset(sgdma_instance_t * this_sgdma, control_type_t)
```

Table 2.2. sgdma_reset()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	None
In	sgdma_cntl_type	Describes the mm2s control offset and s2mm control offset.	

2.3. sgdma_irq_mask()

This API masks the mm2s and s2mm interrupt event upon transfer completion.

```
void sgdma_irq_mask(sgdma_instance_t * this_sgdma, control_type_t sgdma_cntl_type, irq_mask_t
mask_value)
```

Table 2.3. sgdma_irq_mask()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	None
In	sgdma_cntl_type	Describes the mm2s control offset and s2mm control offset.	
In	mask_value	Describes the mask and unmask values.	

2.4. get_sgdma_reg_status()

This API returns the mm2s and s2mm status register depending on the sgdma_sts_type offset.

```
sgdma_sts_reg_t *get_sgdma_reg_status(sgdma_instance_t * this_sgdma, status_type_t
sgdma_sts_type)
```

Table 2.4. get_sgdma_reg_status()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	mm2s and s2mm status register addresses.
In	sgdma_sts_type	Describes the mm2s status offset and s2mm status offset.	

2.5. get_mm2s_bd_status()

This API returns the mm2s single and multi buffer descriptor status field using the index which specifies the number of buffer descriptor.

```
unsigned int get_mm2s_bd_status(int idx)
```

Table 2.5. get_mm2s_bd_status()

In/Out	Parameter	Description	Returns
In	idx	Index that specifies the particular mm2s buffer descriptor number in multiple buffer descriptor.	The descriptor status field.

2.6. get_s2mm_bd_status()

This API returns the s2mm single and multi buffer descriptor status field using the index that specifies the number of buffer descriptor.

```
unsigned int get_s2mm_bd_status(int idx)
```

Table 2.6. get_s2mm_bd_status()

In/Out	Parameter	Description	Returns
In	idx	Index that specifies the s2mm buffer descriptor number in multiple buffer descriptor.	The descriptor status field.

2.7. mm2s_get_desc_complete_bit()

This API verifies the completion bit and error bit of the mm2s_desc_t descriptor.

```
unsigned int mm2s_get_desc_complete_bit(sgdma_instance_t * this_sgdma, unsigned int idx,
unsigned char * isComplete, unsigned char * isError)
```

Table 2.7. mm2s_get_desc_complete_bit()

In/Out	Parameter	Description	Returns
In	*this_sgdma	It has the info of base address of mm2s buffer, length, number of descriptor and blocking and non-blocking value.	1: success 0: failure
In	idx	Index number of descriptors.	
Out	*isComplete	Return specific descriptor's complete bit info.	
Out	*IsError	Return specific descriptor's error info.	

2.8. mm2s_buf_desc_dma()

This API configures the single or multiple buffer descriptors depending on the input parameters, triggers the DMA, and waits for completion depending on the blocking parameters.

```
unsigned int mm2s_buf_desc_dma(sgdma_instance_t * this_sgdma)
```

Table 2.8. mm2s_buf_desc_dma()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the base address of the mm2s buffer, length, number of descriptor, and blocking and non-blocking values.	1: success 0: failure

2.9. s2mm_get_desc_complete_bit()

This API verifies the completion bit and error bit of the s2mm_desc_t descriptor.

```
unsigned int s2mm_get_desc_complete_bit(sgdma_instance_t * this_sgdma, unsigned int idx,
unsigned char * isComplete, unsigned char * isError)
```

Table 2.9. s2mm_get_desc_complete_bit()

In/Out	Parameter	Description	Returns
In	*this_sgdma	It has the info of base address of s2mm buffer, length, number of descriptor and blocking and non-blocking value.	1: success 0: failure
In	idx	Index number of descriptors.	
Out	*isComplete	Return specific descriptor's complete bit info.	
Out	*IsError	Return specific descriptor's error info.	

2.10. s2mm_buf_desc_dma()

This API configures the single or multiple buffer descriptors depending on the input parameters, triggers the DMA, and waits for completion depending on the blocking parameters.

```
unsigned int s2mm_buf_desc_dma(sgdma_instance_t * this_sgdma)
```

Table 2.10. s2mm_buf_desc_dma()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the base address of the mm2s buffer, length, number of descriptor, and blocking and non-blocking values.	1: success 0: failure

2.11. sgdma_register_read()

This API reads data from the MM2S or S2MM register address based on the index value.

```
unsigned int sgdma_register_read(sgdma_instance_t * this_sgdma, sgdma_reg_type_t index,
unsigned int * reg_data)
```

Table 2.11. sgdma_register_read()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure
In	index	Holds the offset value of a particular register.	
Out	*reg_data	Pointer to the value where data is to be stored.	

2.12. sgdma_register_write()

This API writes data from the MM2S or S2MM register address based on the index value.

```
unsigned int sgdma_register_write(sgdma_instance_t * this_sgdma, sgdma_reg_type_t index,
unsigned int reg_data)
```

Table 2.12. sgdma_register_read()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure
In	index	Holds the offset value of a particular register.	
In	*reg_data	Holds that data that is written to the register address.	

3. Function Call Flow Diagrams

3.1. sgdma_init()

```
void sgdma_init(sgdma_instance_t * this_sgdma, unsigned int base_addr)
```

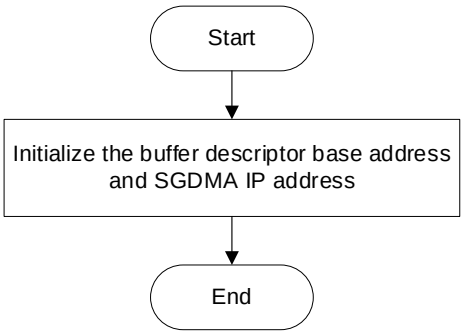

Figure 3.1. sgdma_init()

3.2. sgdma_reset()

```
void sgdma_reset(sgdma_instance_t * this_sgdma)
```

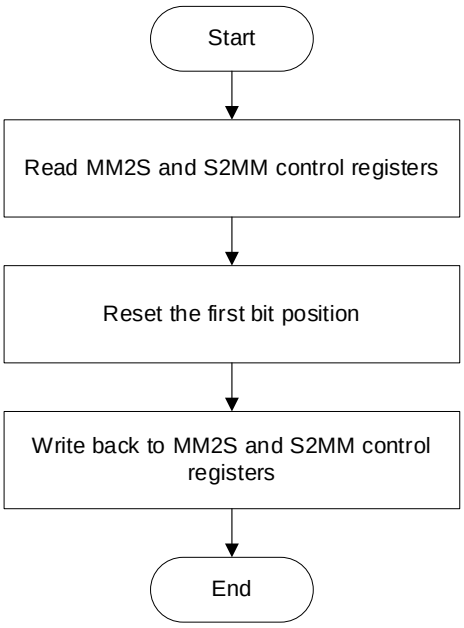

Figure 3.2. sgdma_reset()

3.3. sgdma_irq_mask()

```
void sgdma_irq_mask(sgdma_instance_t * this_sgdma, control_type_t sgdma_cntl_type,
irq_mask_t mask_value)
```

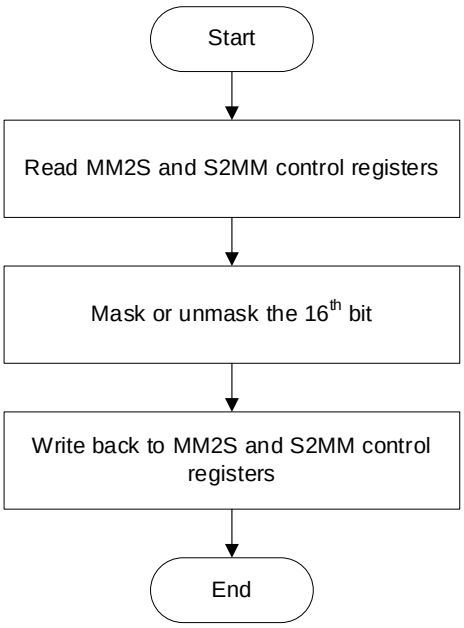

Figure 3.3. sgdma_irq_mask()

3.4. get_sgdma_reg_status()

```
sgdma_sts_reg_t * get_sgdma_reg_status(sgdma_instance_t * this_sgdma, status_type_t
sgdma_sts_type)
```

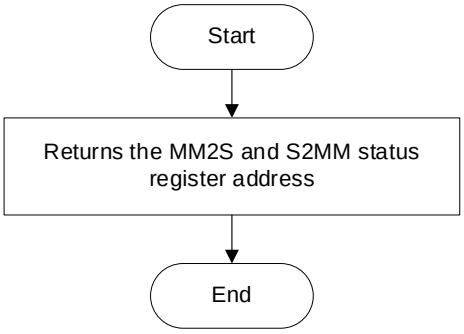

Figure 3.4. get_sgdma_reg_status()

3.5. get_mm2s_bd_status()

```
unsigned int get_mm2s_bd_status(int idx)
```

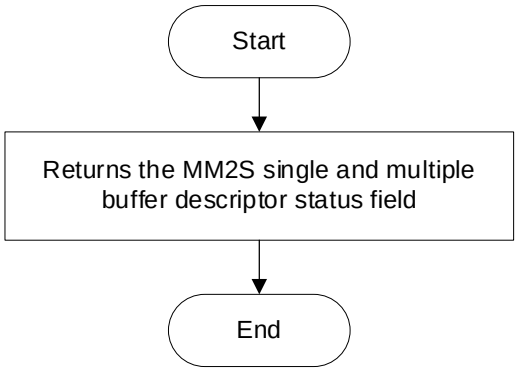

Figure 3.5. get_mm2s_bd_status()

3.6. get_s2mm_bd_status()

```
unsigned int get_s2mm_bd_status(int idx)
```

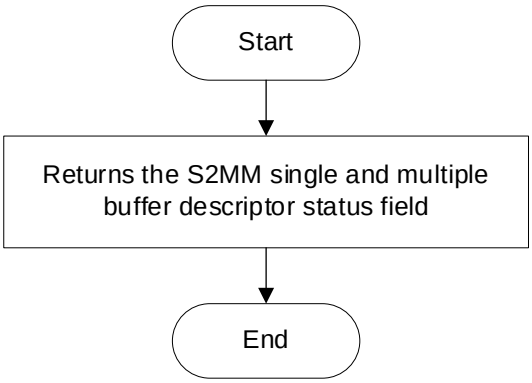

Figure 3.6. get_s2mm_bd_status()

3.7. mm2s_get_desc_complete_bit

```
unsigned int mm2s_get_desc_complete_bit(sgdma_instance_t * this_sgdma, unsigned int idx,
unsigned char * isComplete, unsigned char * isError)
```

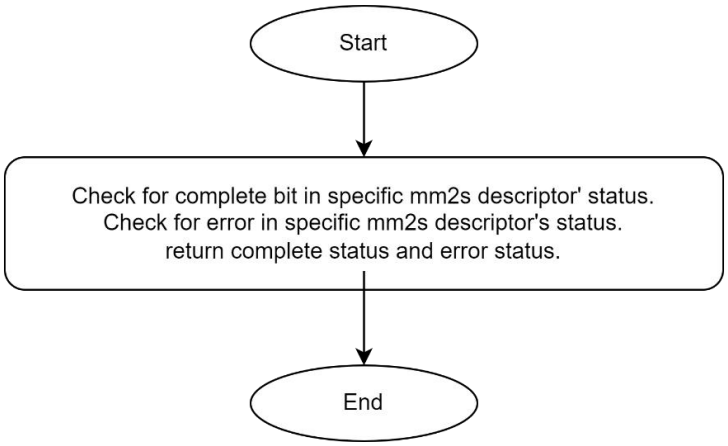

Figure 3.7. mm2s_get_desc_complete_bit

3.8. mm2s_buf_desc_dma()

```
unsigned int mm2s_buf_desc_dma(sgdma_instance_t * this_sgdma)
```

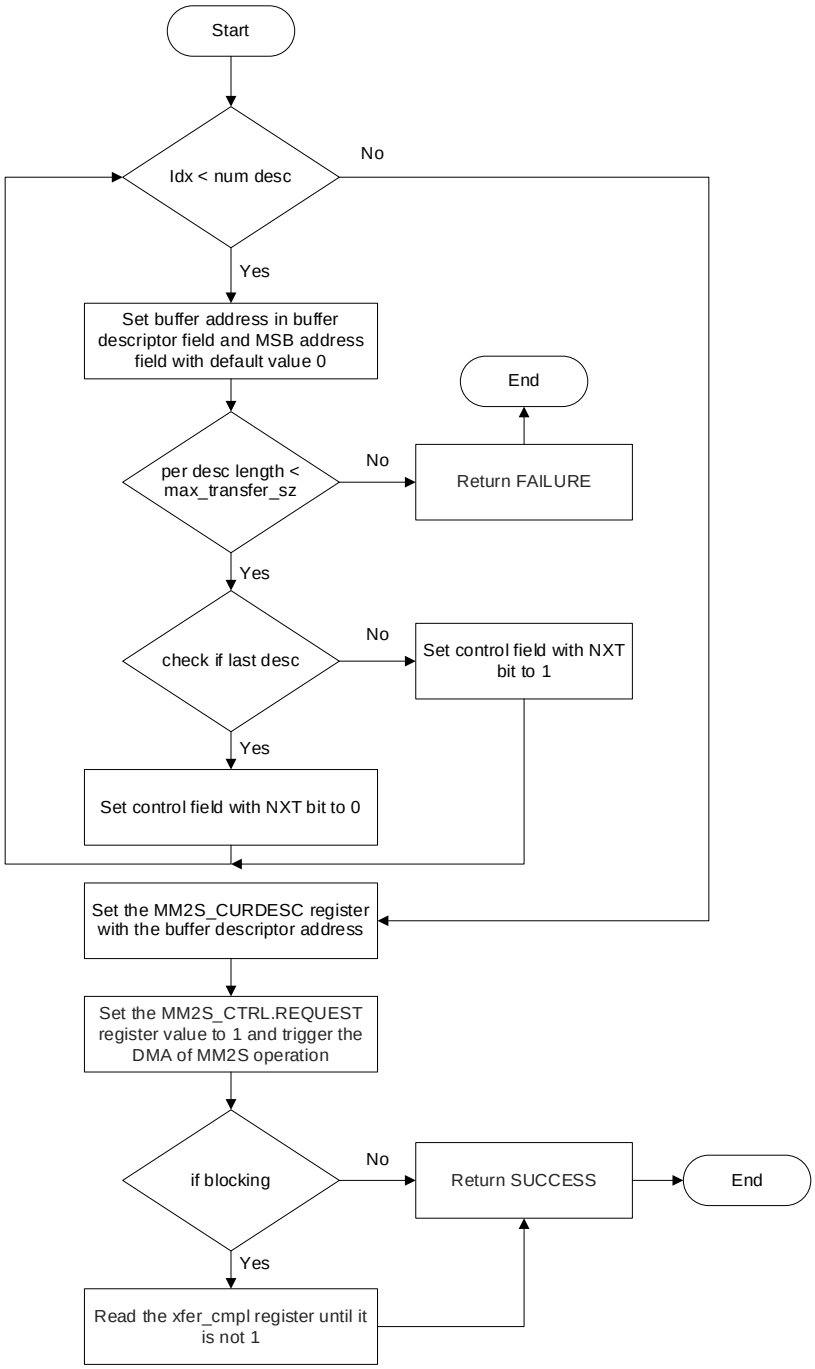

Figure 3.8. mm2s_buf_desc_dma()

3.9. s2mm_get_desc_complete_bit

```
unsigned int s2mm_get_desc_complete_bit(sgdma_instance_t * this_sgdma, unsigned int idx,
unsigned char * isComplete, unsigned char * isError)
```

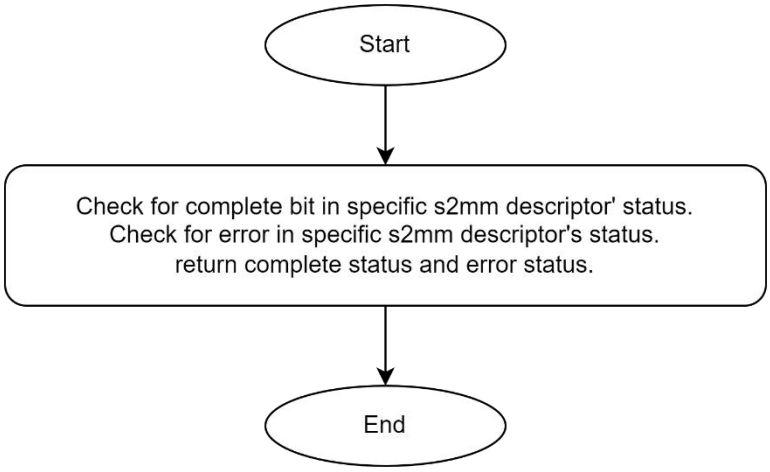

Figure 3.9. s2mm_get_desc_complete_bit

3.10. s2mm_buf_desc_dma()

```
unsigned int s2mm_buf_desc_dma(sgdma_instance_t * this_sgdma)
```

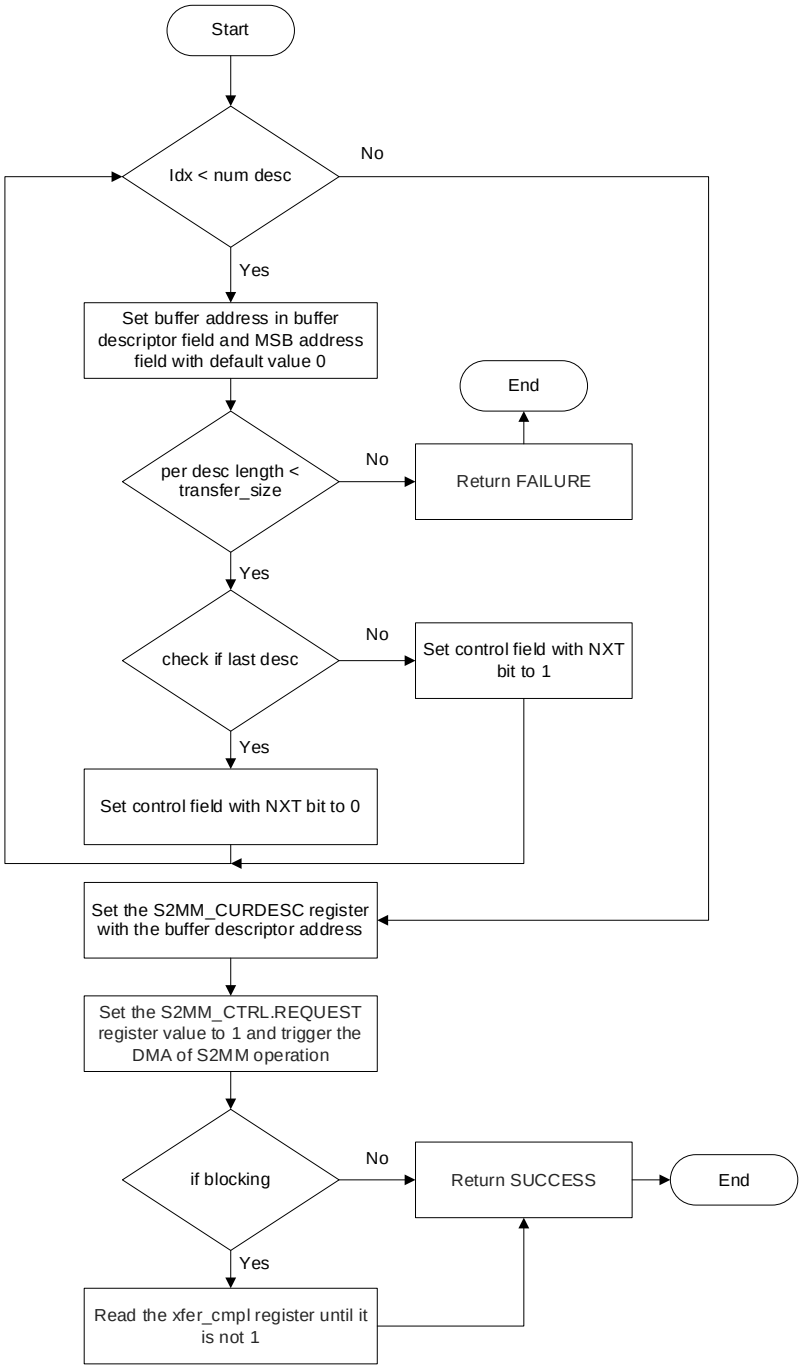

Figure 3.10. s2mm_buf_desc_dma()

3.11. sgdma_register_read()

```
unsigned int sgdma_register_read(sgdma_instance_t *this_sgdma, sgdma_reg_type_t index,
unsigned int *reg_data)
```

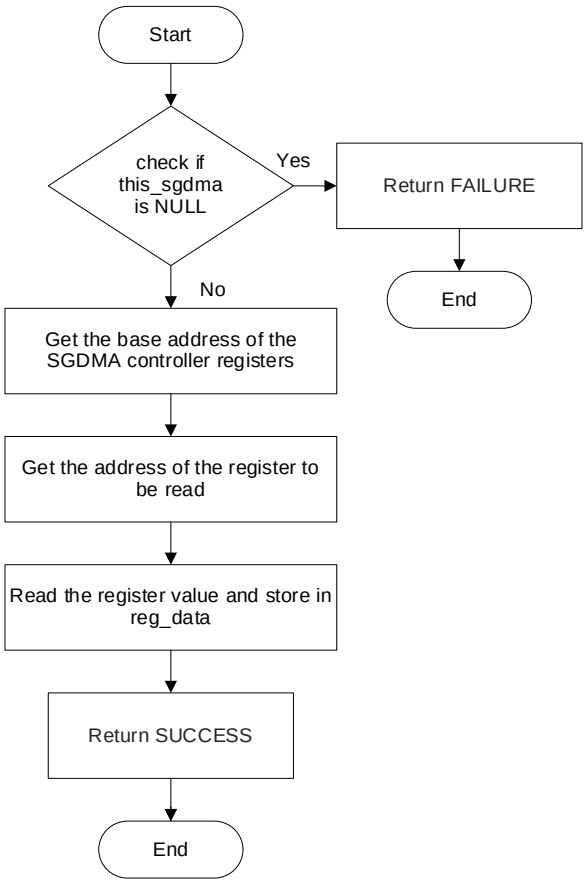

Figure 3.11. sgdma_register_read()

3.12. sgdma_register_write()

```
unsigned int sgdma_register_write(sgdma_instance_t *this_sgdma, sgdma_reg_type_t index,
unsigned int reg_data)
```

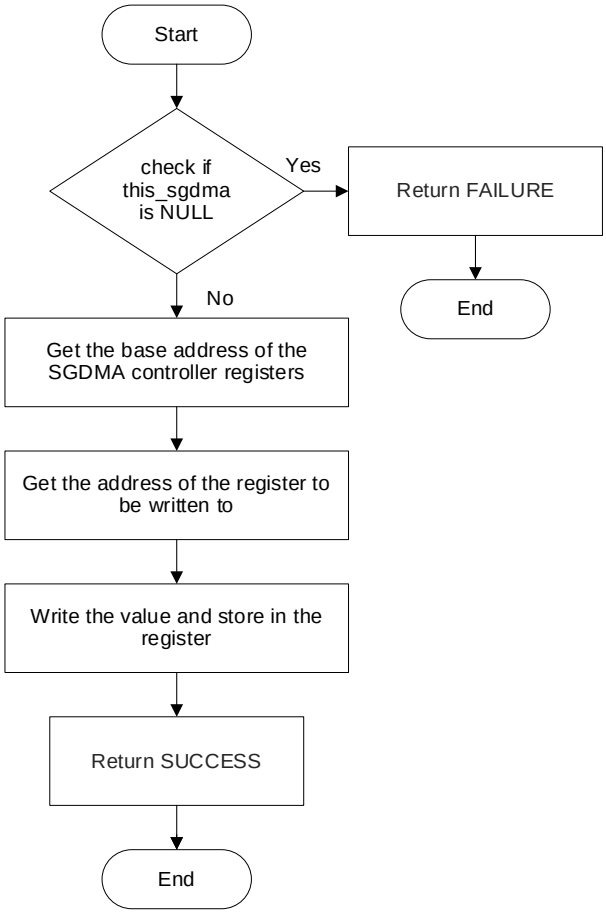

Figure 3.12. sgdma_register_write()

4. API Data Structure and Enums

4.1. struct mm2s_ctrl_reg_t

Table 4.1. struct mm2s_ctrl_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	mm_request	1
volatile unsigned int	mm_reset	1
volatile unsigned int	mm_rsvd_1	14
volatile unsigned int	mm_cmpl_irq_mask	1
volatile unsigned int	mm_rsvd_2	15

4.2. struct mm2s_sts_reg_t

Table 4.2. struct mm2s_sts_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	mm_status	1
volatile unsigned int	mm_rsvd_1	7
volatile unsigned int	mm_reg_bd_len_err	1
volatile unsigned int	mm_reg_axi_slave_err	1
volatile unsigned int	mm_reg_axi_desc_err	1
volatile unsigned int	mm_rsvd_2	5
volatile unsigned int	mm_xfer_cmpl	1
volatile unsigned int	mm_xfer_err	1
volatile unsigned int	mm_rsvd_3	14

4.3. struct mm2s_desc_tx_t

Table 4.3. struct mm2s_desc_tx_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	mm_buffer_addr	32
volatile unsigned int	mm_buffer_msb_addr	32
volatile unsigned int	mm_control_buffer_size	16
volatile unsigned int	mm_control_tdest	4
volatile unsigned int	mm_control_tid	4
volatile unsigned int	mm_control_arprot	3
volatile unsigned int	mm_control_rsvd	3
volatile unsigned int	mm_control_fp	1
volatile unsigned int	mm_control_nxt	1
volatile unsigned int	mm_status_transffered_size	16
volatile unsigned int	mm_status_rsvd	11
volatile unsigned int	mm_status_axi_desc_err	1
volatile unsigned int	mm_status_axi_slave_err	1
volatile unsigned int	mm_status_transffered_len_err	1
volatile unsigned int	mm_status_bd_len_err	1
volatile unsigned int	mm_status_cmpl	1

4.4. struct mm2s_desc_tx_ext_t

Table 4.4. struct mm2s_desc_tx_ext_t

Data Type	Structure member
volatile unsigned int	mm_buffer_addr
volatile unsigned int	mm_buffer_msb_addr
volatile unsigned int	mm_control
volatile unsigned int	mm_status

4.5. union mm2s_desc_t

Table 4.5. union mm2s_desc_t

Data Type	Union member
mm2s_desc_tx_ext_t	mm_bd_ext
mm2s_desc_tx_t	mm_bd

4.6. struct s2mm_ctrl_reg_t

Table 4.6. struct s2mm_ctrl_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	s_request	1
volatile unsigned int	s_reset	1
volatile unsigned int	s_rsvd_1	14
volatile unsigned int	s_cmpl_irq_mask	1
volatile unsigned int	s_rsvd_2	15

4.7. struct s2mm_sts_reg_t

Table 4.7. struct s2mm_sts_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	s_status	1
volatile unsigned int	s_rsvd_1	7
volatile unsigned int	s_reg_bd_len_err	1
volatile unsigned int	s_reg_axi_slave_err	1
volatile unsigned int	s_reg_axi_desc_err	1
volatile unsigned int	s_rsvd_2	5
volatile unsigned int	s_xfer_cmpl	1
volatile unsigned int	s_xfer_err	1
volatile unsigned int	s_rsvd_3	14

4.8. struct s2mm_desc_tx_t

Table 4.8. struct s2mm_desc_tx_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	s_buffer_addr	32
volatile unsigned int	s_buffer_msb_addr	32
volatile unsigned int	s_control_buffer_size	16
volatile unsigned int	s_control_rsvd_1	8
volatile unsigned int	s_control_awprot	3
volatile unsigned int	s_control_rsvd_2	4
volatile unsigned int	s_control_nxt	1
volatile unsigned int	s_status_transffered_size	16
volatile unsigned int	s_status_rsvd	11
volatile unsigned int	s_status_axi_desc_err	1
volatile unsigned int	s_status_axi_slave_err	1
volatile unsigned int	s_status_transffered_len_err	1
volatile unsigned int	s_status_bd_len_err	1
volatile unsigned int	s_status_cmpl	1

4.9. struct s2mm_desc_tx_ext_t

Table 4.9. struct s2mm_desc_tx_ext_t

Data Type	Structure member
volatile unsigned int	s_buffer_addr
volatile unsigned int	s_buffer_msb_addr
volatile unsigned int	s_control
volatile unsigned int	s_status

4.10. union s2mm_desc_t

Table 4.10. union s2mm_desc_t

Data Type	Union member
s2mm_desc_tx_ext_t	s_bd_ext
s2mm_desc_tx_t	s_bd

4.11. struct sgdma_instance_t

Table 4.11. struct sgdma_instance_t

Data Type	Structure member
unsigned int	base_addr
unsigned int	*mm2s_buffer_addr
unsigned int	*s2mm_buffer_addr
unsigned int	axi4_mm_data_width
unsigned int	num_of_mm2s_desc
unsigned int	num_of_s2mm_desc
unsigned int	blocking_S2MM
unsigned int	blocking_MM2S
mm2s_desc_t	*mm2s_bd_addr
s2mm_desc_t	*s2mm_bd_addr
unsigned int	*per_mm2s_desc_length
unsigned int	*per_s2mm_desc_length
unsigned int	fp

4.12. struct sgdma_ctrl_reg_t

Table 4.12. struct sgdma_ctrl_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	request	1
volatile unsigned int	reset	1
volatile unsigned int	rsvd_1	14
volatile unsigned int	cmpl_irq_mask	1
volatile unsigned int	rsvd_2	15

4.13. struct sgdma_sts_reg_t

Table 4.13. struct sgdma_sts_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	status	1
volatile unsigned int	rsvd_1	7
volatile unsigned int	reg_bd_len_err	1
volatile unsigned int	reg_axi_slave_err	1
volatile unsigned int	reg_axi_desc_err	1
volatile unsigned int	rsvd_2	5
volatile unsigned int	xfer_cmpl	1
volatile unsigned int	xfer_err	1
volatile unsigned int	rsvd_3	14

4.14. enum control_type_t

Table 4.14. enum control_type_t

Data Type	Value
MM2S_RESET	0
S2MM_RESET	0x20

4.15. enum status_type_t

Table 4.15. enum status_type_t

Data Type	Value
MM2S_STATUS	0x4
S2MM_STATUS	0x24

4.16. enum irq_mask_t

Table 4.16. enum irq_mask_t

Data Type	Value
UNMASK	0
MASK	1

4.17. enum sgdma_reg_type_t

Table 4.17. enum sgdma_reg_type_t

Data Type	Value
MM2S_CTRL_REG	0x0
MM2S_STATUS_REG	0x4
MM2S_CURDESC_REG	0x8
S2MM_CTRL_REG	0x20
S2MM_STATUS_REG	0x24
S2MM_CURDESC_REG	0x28

5. API Variables

Table 5.1. API Variables

Data Type	Variable Name
mm2s_desc_t	*mm_multi_bd_confg_ptr
s2mm_desc_t	*s_multi_bd_confg_ptr

6. API Macros

Table 6.1. API Macros

Macro name	Description
MM2S_CTRL	mm2s control offset
MM2S_STS	mm2s status offset
MM2S_CURDESC	mm2s current descriptor offset
S2MM_CTRL	s2mm control offset
S2MM_STS	s2mm status offset
S2MM_CURDESC	s2mm current descriptor offset
MAX_TRANSFER_SIZE	max total data bytes data transferred
MULTI_BUF_SIZE	multiple buffer descriptor number
FP_BIT_POSITION	full packet bit position
NXT_BIT_POSITION	next bit position
TID_BIT_POSITION	tid bit position
TDEST_BIT_POSITION	tdest bit position
XFER_CMPL_BIT_POSITION	xfer_cmpl bit position
DESC_SIZE	size of a buffer descriptor
BUFFER_SIZE_MASK	use masking for extract buffer size
SGDMA_RESET	reset control register of s2mm and mm2s
DMA_TRIGGER	dma trigger value
MSB_ADDR	use for msb address
NXT_1	1 for multiple descriptor
NXT_0	0 for last descriptor
FP_1	full packet pre-fetch required
FP_0	full packet pre-fetch not required
XFER_CMPL_1	use for bit operation
IDX_0	use for index zero
TRUE	1 for conditional operation
FALSE	0 for conditional operation
RET_FAILURE	1
RET_SUCCESS	0
IS_NULL	0

References

- SGDMA Controller IP Core - Lattice Radiant Software User Guide (FPGA-IPUG- 02131)
- CertusPro-NX web page
- Lattice Solutions IP Cores web page
- Lattice Radiant Software FPGA web page
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.2, July 2024

Section	Change Summary
Introduction	• Updated SGDMA version from 1.0.0 to *24.01.00*. • Updated Driver and IP versions in Table 1.1. Driver and IP Compatibility. • Added Table 1.2. Quick Facts of Driver Tested Environment.
API Description	• Updated sgdma_init() to remove *unsigned int num_of_desc* in the programming code. • Updated sgdma_reset() to remove *sgdma_cntl_type* in the programming code. • Updated API description of get_sgdma_reg_status() section. • Added mm2s_get_desc_complete_bit() and s2mm_get_desc_complete_bit(). • Updated tables in mm2s_buf_desc_dma() and s2mm_buf_desc_dma() to change Returns column value to *1 – success* and *0 – failure*. • Updated API descriptions and tables of sgdma_register_read() and sgdma_register_write() to change Returns column value to *1 – success* and *0 – failure*.
Function Call Flow Diagrams	• Added programming codes in sgdma_init() to sgdma_register_write(). • Added mm2s_get_desc_complete_bit and s2mm_get_desc_complete_bit sections.

API Data Structure and Enums	• Updated the following in the struct sgdma_instance_t table:
	  • Updated values in Structure Member column.
	  • Added axi4_mm_data_width, num_of_s2mm_desc, *per_s2mm_desc_length, and fp rows.
	  • Removed **buffer* row.

API Variables	Updated Variable Name column to add *_ptr* to the variable name.

Revision 1.1, January 2024

Section	Change Summary
Inclusive Language	Added this section.
Introduction	Updated the IP Version to 2.0.1 in 1.4. Driver and IP Compatibility section.
API Description	• Updated header name for 2.1. sgdma_init() – 2.10. sgdma_register_write() sections. • Added the full API descriptions into respective paragraph for 2.1. sgdma_init() – 2.10. sgdma_register_write() sections. • Updated the descriptions for *this_sgdma, base_addr, and num_of_desc parameters in 2.1. sgdma_init() section. • Updated the paragraph to: This API configures the single or multiple buffer descriptors depending on the input parameters, triggers the DMA, and waits for completion depending on the blocking parameters in 2.7. mm2s_buf_desc_dma() and 2.8. s2mm_buf_desc_dma() sections. • Added 2.9. sgdma_register_read() and 2.10. sgdma_register_write() sections.
Function Call Flow Diagrams	• Updated header name for 3.1. sgdma_init() – 3.10. sgdma_register_write() sections. • Updated Figure 3.1. sgdma_init() – Figure 3.8. s2mm_buf_desc_dma(). • Added 3.9. sgdma_register_read() and 3.10. sgdma_register_write() sections.
API Data Structure and Enums	• Added and Enums to header name of 4. API Data Structure and Enums section. • Added 4.12. struct sgdma_ctrl_reg_t – 4.17. enum sgdma_reg_type_t sections.
API Macros	• Updated macro names from *FAILURE* to *RET_FAILURE* and *SUCCESS* to *RET_SUCCESS*. • Removed *DWORD_4* macro. • Added *IS_NULL* macro.

Revision 1.0, December 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com