LatticeMico VID

LatticeMico Voltage ID (VID) is a soft IP that allows users to provide target voltage set points of the Close Loop Trim (CLT) circuitry on a Lattice Analog Sense and Control (ASC) device. Although the voltage set points of the CLT can be defined through voltage profiles during the design stage, the VID IP provides a mean to update the target set points on the fly while the system is in operation.

The LatticeMico VID IP provides the voltage set points as starting points for individual CLT channels. It does not monitor the voltage trimming / margining process thus cannot be considered as part of the close-loop process. For more information details of the CLT functionality and process, refer to the Platform Designer documentation in Diamond online help.

Version

This document describes the 1.1 version of the LatticeMico VID.

Features

The LatticeMico VID IP must be used together with the EFB (Embedded Functional Block) of the Platform Manager 2 device. The voltage set points of the CLT channels are updated by the VID IP through the primary I^2C port of the EFB.

- WISHBONE Master or WISHBONE Slave mode support.
- WISHBONE interface with 8-bit data bus.
- Individual control for each CLT channel.

- Toggling of VID enable signal activates the VID request for the specific channel.
- Support voltage look-up table from 8 to 64 entries per CLT channel.
- VID requests are processed in round-robin algorithm without priority.
- Maximum of 16 CLT channels can be supported.
- The voltage set point for the CLT channels is 13-bit binary value.
- Handles internal I^2C resources sharing through MUTEX.
- External I^2C bus collision and arbitration lost detection.
- Enable/disable the I^2C Write Protect pin of the ASC devices.

Functional Description

This VID IP can be used as a WISHBONE Master or a WISHBONE Slave. When in the WISHBONE Master mode, the IP is self-contained and does not require a microcontroller to function. When in the WISHBONE Slave mode, it requires LatticeMico soft microcontroller to be used together to complete the function. Figure 1 and Figure 2 give an overview of the IP applications based on the mode it is selected.

Figure 1: VID WISHBONE Master Mode Block Diagram

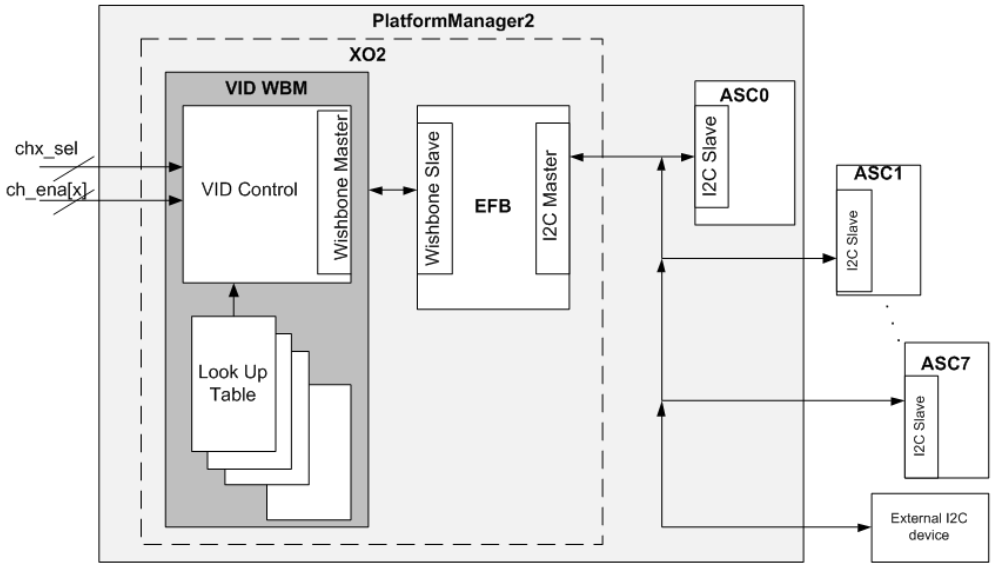
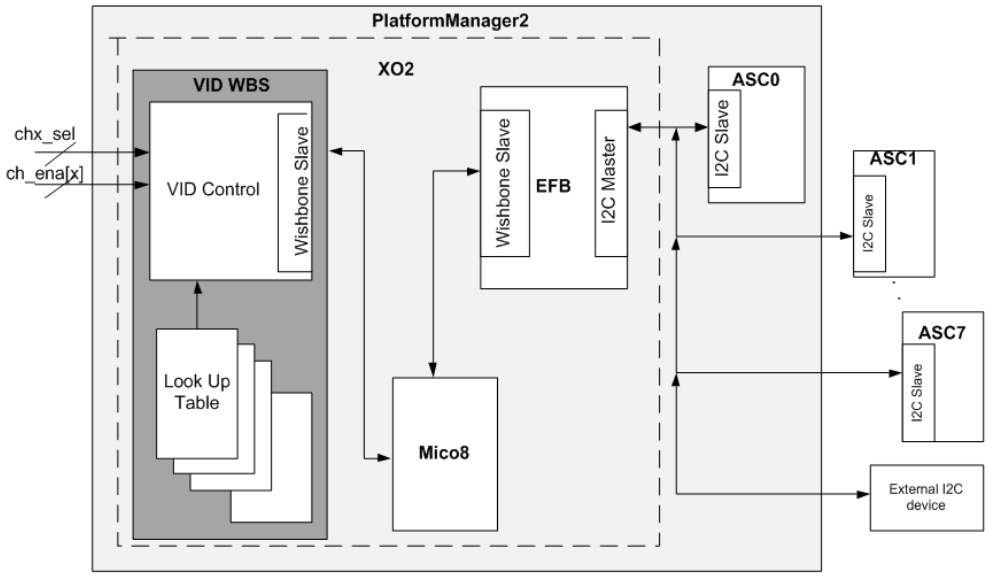


Figure 2: VID WISHBONE Slave Mode Block Diagram



VID Request Triggering Events

A VID event is triggered by the toggling of the channel enable signals (ch_ena[x]). The polarity of the channel enable signals is set by the users during the IP configuration. Depending on the polarity of the channel enable signals, either the rising edge or the falling edge signals a VID request. This edge at the same time registers the channel select signals (chx_sel). This tells the VID IP that voltage set point register of channel x needs to be updated with the voltage value pointed by the chx_sel signals. The identification of channel x is the ASC slave address and the CLT channel number. Both are pre-defined during the design stage by the user.

Once a VID request is captured, it is not cleared unless successfully processed. When a particular channel's VID request is not being processed, users can over-write the current request with a new request.

Voltage Look Up Tables

Each channel has a table that stores a set of voltage set point values. The table size is between eight entries to 64 entries. A common table can be shared among multiple channels. The IP can have one table to be shared by all 16 channels, or maximum of 16 tables, one for each channel. The voltage set point values and the tables are defined during the design stage through the Platform Designer GUI in the Diamond software.

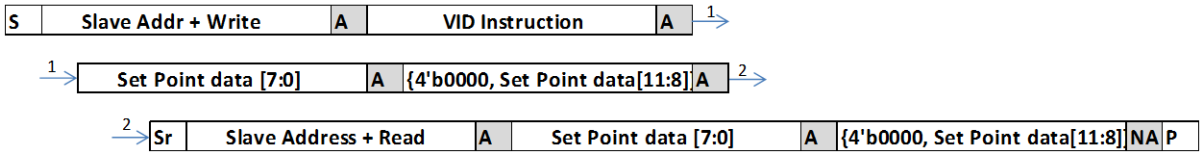
WISHBONE Master Mode

When in WISHBONE Master mode, the sequence of processing a VID request is controlled by the WISHBONE Master state machine. The state machine will not start working unless there is a VID request in place. Before initiating a transfer on the I^2C bus, the state machine checks for the MUTEX possession and the I^2C bus status. The MUTEX is a soft IP that provides the arbitration control among resources on XO2 side that need to use the I^2C bus. The WISHBONE master has to own the MUTEX before it can process the VID request. Then the checking of I^2C status register allows the detection of busy on the external I^2C bus. Only when both conditions are cleared, the state machine will start the transfer on the I^2C bus and disable the I^2C Write Protect pin.

The state machine also checks the arbitration lost during the I^2C WRITE operation. This is done through the checking of the ARBL bit of the EFB's I^2C status register. Once the arbitration lost is detected, the master enables the I^2C Write Protect pin, and waits for the I^2C bus to be free again. Then it enables the I^2C Write Protect pin and re-transmits the data again. When the VID request is successfully processed, meaning the read back data matching sending data, the request bit is cleared. Otherwise, the request is kept and is processed again in a round-robin fashion.

A normal I^2C bus sequence for a VID request is shown in Figure 3. The required number of I^2C clock to complete a VID request is at least 66. The sequence could be much longer if arbitration lost is detected during the I^2C Write operations.

Figure 3: I^2C Bus Sequence for VID Request



WISHBONE Slave Mode

When in WISHBONE Slave mode, the VID IP issues an interrupt request to inform LatticeMico that a VID request is in place. LatticeMico has to enable the interrupt mask register in order to see the status change on the interrupt signal. When the microcontroller decides to process the VID request, it gets the necessary information first from the VID IP by accessing the register set through WISHBONE bus. A typical sequence of the processing the VID request in Slave mode is listed for reference. Refers to the VID Slave Register Map section for the definition of each register.

1. Enable interrupt mask bit (IRQENR register) to allow the interrupt status be seen on the interrupt pin.

2. Check if Dual Boot is busy by checking the STATUS register bit 4. Wait until Dual Boot is not busy before proceeding to the following steps.
3. Find out which channels have the VID requests by read the REQ_LOW and REQ_HIGH registers
4. Decide to process a particular channel's request first by writing to the IN_PROC_LOW or IN_PROC_HIGH registers. Setting a "1" in the register bit will prevent the new request on the same channel to over-write the current request. Ideally there should be one active IN_PROC bit at a time.
5. Gather the identification of the CLT channel by reading the CHx_INFO register, where x corresponds to the position of register bit in the IN_PROC register. This provides the channel ID and the ASC ID information to LatticeMico.
6. Get the ASC slave address by reading the SLAVE_ADDR register, which provides the ASC slave address of the channel that LatticeMico is going to process
7. Get the data that needs to be sent to the ASC CLT voltage set point register. This is stored in the DATA_LOW and DATA_HIGH registers.
8. When all the necessary data is available for the VID request, LatticeMico writes to the STATUS register to disable the I^2C Write Protect so that the ASC devices can accept data.
9. LatticeMico can enable I^2C Write Protect (write to the STATUS register) at any time to stop the ASC devices from receiving data when arbitration lost is detected, or when I^2C Write is completed.
10. LatticeMico writes to STATUS register (bits 1 to 3) when VID request is completed. STATUS bits are updated depending on whether the request is processed successfully or not.
11. Write to the IN_PROC register to clear the bit, indicating the request is served.

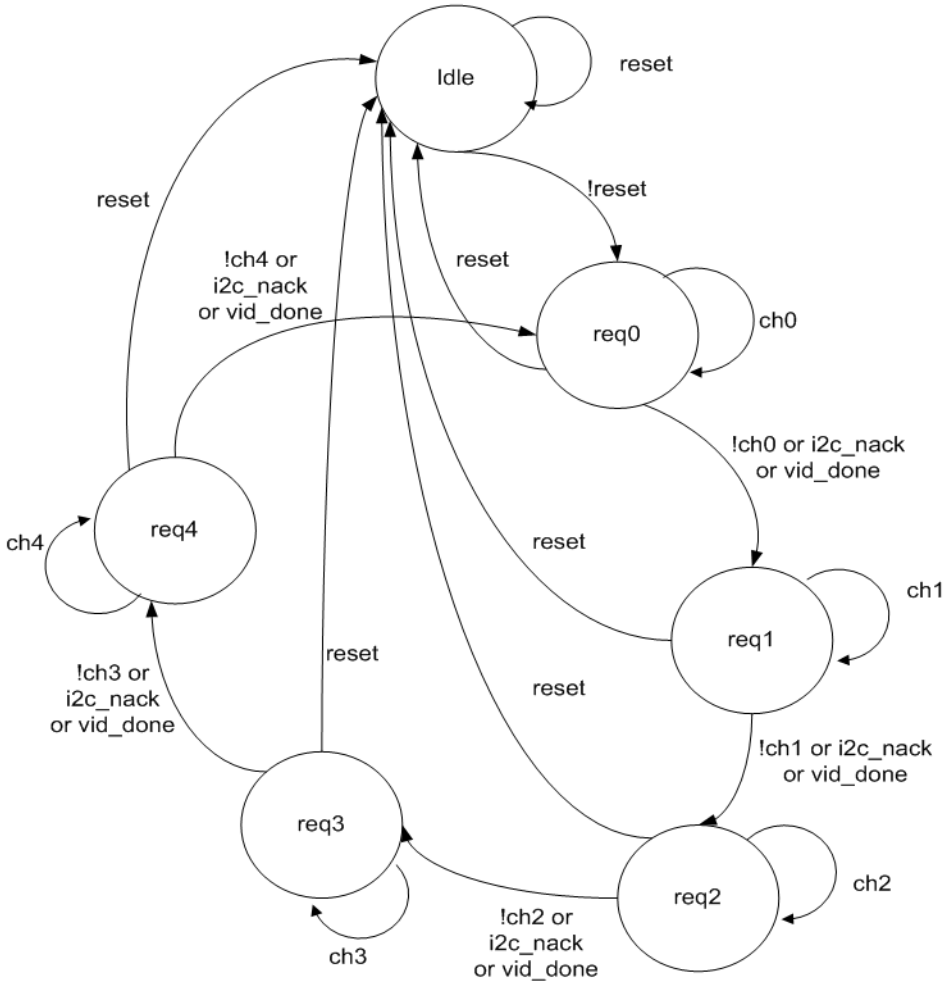
Figure 4 is a state diagram of a 5 VID channel request.

LatticeMico EFB and LatticeMico Mutex Support

The LatticeMico EFB will be used for the interface between VID control and the ASC devices. The LatticeMico EFB I^2C always acts as I^2C Master, and ASC I^2C always acts as slave and supports 7-bit addressing only. Users need to avoid setting the ASC I^2C slave address to be the same as the LatticeMico EFB I^2C slave address.

Both VID WBM and LatticeMico 8 will poll the bits of the EFB primary I^2C Status Register to check the status on the I^2C bus. WBM will issue a START when the Busy bit is not asserted. When the START is issued, WBM monitors the ARBL bit to see if it wins the arbitration. If an arbitration loss is encountered, WBM will monitor the BUSY status bit again to wait for the I^2C bus to free up.

Figure 4: State Diagram of 5 VID channel Request



In the case of competing I^2C resources with background programming, WBM will write and read the MUTEX component to determine if the I^2C resources is occupied. Detail of arbitrating between VID and internal JTAG-I^2C route will base on the LatticeMico Mutex documentation.

Configuration

The following sections describe the graphical user interface (UI) parameters, the hardware description language (HDL) parameters, and the I/O ports that user can use to configure and operate the LatticeMico VID.

UI Parameters

Table 1 shows the UI parameters available for configuring the LatticeMico VID through the Mico System Builder (MSB) interface. For more information refer to the Platform Designer documentation in Diamond online help.

Table 1: VID UI Parameters

Dialog Box Options	Description	Allowable Values	Default Value
Instance Name	Specifies the name of the VID instance.	Alphanumeric and underscores	vid
VID Mode Selection			
Master Mode	Specifies the VID as master mode.	selected | not selected	not selected
Slave Mode	Specifies the VID as slave mode.	selected | not selected	selected
Number of VID channel	Specifies the number of VID channel.	1 - 16	1
Base Address	Specifies the base address for configuring the VID. The minimum boundary alignment is 0x80.	0X80000000–0XFFFFFFFF	0X80000000
EFB Slave Address	Specifies the base address for configuring the EFB. The minimum boundary alignment is 0x80.	0X80000000–0XFFFFFFFF	0X80000000
Mutex Enable	When selected, Mutex is enabled.	selected | not selected	not selected
Mutex Slave Address	Specifies the base address for configuring the Mutex. The minimum boundary alignment is 0x80.	0X80000000–0XFFFFFFFF	0X80000000
Channel Setting			
VID Table Setting			
Table ID	Specifies the Table ID.	0-15	0
Initialization File	Specifies the Initialization table.	<user_defined>.mem	none
VID Channel Polarity			
Positive Polarity	When selected, chx_ena is positive edge triggered.	selected | not selected	selected
Negative Polarity	When selected, chx_ena is negative edge triggered.	selected | not selected	not selected
VID Channel Setting			
Select Channel Width	Channel select width.	3-6	3

Table 1: VID UI Parameters (Continued)

Dialog Box Options	Description	Allowable Values	Default Value
ASC ID	Specifies the ASD ID.	0-7	0
CLT ID	Specifies the CLT ID.	0-7	0

HDL Parameters

Table 2 lists the parameters that appear in the HDL.

Table 2: VID HDL Parameter

Parameter Name	Description	Allowable Values
LATTICE_FAMILY	Define the device family for the IP	MACHXO2 | LPTM2
VID CHANNELS	Define the total number of VID channels	1 to 16
WBM_VID_EN	A value of 1 defines the LatticeMico VID as Master Mode	0 | 1
WBS_VID_EN	A value of 1 defines the LatticeMico VID as Slave Mode	0 | 1
WBM_VID_MUTEX_EN	A value of 1 defines the Mutex is enabled	0 | 1
WBM_MUTEX_BASE_ADDR	WISHBONE base address of MUTEX	32 bits
WBM_EFB_BASE_ADDR	WISHBONE base address of EFB	32 bits
VID Table (per CLT channel that are using the VID function)		
VID TABLEx	Define the VID table name	ASCI file name
VID_TABLEx_ID	Define the VID table IDs of the VID channel	0 - 15
VID_CHy_SEL_BITS	Define the select bits range of the VID channel	3 to 6
VID CHy_POL	A value of 1 defines the Polarity of Chx_ena signal is positive edge triggered	0 | 1
VID_CHy_ASC	Define the ASC ID of the VID channel	0 to 7
VID_CHy_CLT	Define the CLT ID of the VID channel within an ASC	0 to 7
WAIT_COUNT	Number of wait state added between each VID process	0 to 15
I^2C slave address		
ASCy_I2C_S_ADDR	I^2C slave addresses for ASC devices.	32 bits

Note 1: x = 0 – 15, y = 0 - 7

Note 2: Each VID table has a unique ID. This is to identify which channels are sharing the same VID table. The VID Table ID always starts from 0. That means if there is one VID table then the VID Table ID=0. The channels that share a VID table should always use the VID Table ID of the lowest channel that shares the table. Table 3 shows an example of five VID channels (channels 0 to 4) that share three VID tables. Their IDs are shown in “VID Table ID” column.

Table 3: Example of Five VID Channels That Share Three VID Tables

VID Table names	Channel	VID Table ID
vid_3v3	0	0
vid_3v3	1	0
abc_123	2	1
thrd_2v5	3	2
abc_123	4	1

If not using the Platform Designer GUI to configure the IP, users must do design rule checking manually. Here is a list of design rules for reference.

1. WISHBONE Master and WISHBONE Slave functions are mutually exclusive. Their corresponding enable parameter cannot be turned on (assigned 1'b1) or turned off (assigned 1'b0) at the same time.
2. Number of VID Channel cannot exceed 16.
3. Each VID table has a unique ID. This is to identify which channels are sharing the same VID table. The VID Table ID always starts from 0. That means if there is one VID table then the VID Table ID=0. The channels that share a VID table should always use the VID Table ID of the lowest channel that shares the table.
4. The number of select bits must correspond to the size of the VID table. For example a channel with 4-bit select bits can support up to 16-entry table.
5. When more than one channels share a table, make sure the VID Table ID are the same, the number of select inputs are the same, and the table name are the same.
6. VID Table ID must not skip number. For example if there are 3 channels, then ID can be 0, 1, 2, or 0, 0, 1, etc. as long as the numbers starts from 0 and moves up sequentially without skipping, It cannot be, for example, 0, 1, 4.

I/O Ports

Table 4 describes the input and output ports of the LatticeMico VID.

Table 4: VID I/O Ports

I/O Port	Direction	Active	Description
System Clock and Reset			
CLOCK	I	—	System Clock

Table 4: VID I/O Ports (Continued)

I/O Port	Direction	Active	Description
RESETN	I	Low	System Reset
WISHBONE Master Signal			
WBM_DAT_I	I	—	8-bit data used to read a byte of data from a specific register in the register
WBM_ACK_I	I	High	Transfer acknowledge signal asserted, indicates the requested transfer is acknowledged.
WBM_ERR_I	I	—	Not used, always tied to 0
WBM_RTY_I	I	—	Not used, always tied to 0
WBM_CYC_O	O	High	Indicates a valid bus cycle is present on the WISHBONE bus.
WBM_STB_O	O	High	Indicates the WISHBONE slave is the target for the current transaction on the bus.
WBM_WE_O	O	—	Level sensitive Write/Read control signal. Low - Read operation, High - Write operation
WBM_ADR_O	O	—	32-bit wide address used to select a specific register
WBM_DAT_O	O	—	8-bit data used to read a byte of data from a specific register
WBM_CTI_O	O	—	Not used, always tied to 0
WBM_BTE_O	O	—	Not used, always tied to 0
WBM_LOCK_O	O	—	Not used, always tied to 0
WBM_SEL_O	O	—	Not used, always tied to 0
WISHBONE Slave Signal			
WBS_CYC_I	I	High	Indicates a valid bus cycle is present on the bus.
WBS_STB_I	I	High	Asserts an acknowledgment in response to the assertion of the WISHBONE Master strobe.
WBS_WE_I	I	—	Level sensitive Write/Read control signal. Low - Read operation, High - Write operation
WBS_ADR_I	I	—	32-bit wide address used to select a specific register
WBS_DAT_I	I	—	8-bit data used to read a byte of data from a specific register
WBS_CTI_I	I	—	Not used, always tied to 0
WBS_BTE_I	I	—	Not used, always tied to 0
WBS_LOCK_I	I	—	Not used, always tied to 0
WBS_SEL_I	I	—	Not used, always tied to 0

Table 4: VID I/O Ports (Continued)

I/O Port	Direction	Active	Description
WBS_DAT_O	O	—	8-bit data used to read a byte of data from a specific register
WBS_ACK_O	O	High	Indicates the requested transfer is acknowledged.
WBS_ERR_O	O	—	Indicates the address is incorrect
WBS_RTY_O	O	—	Not used, always tied to 0
Other signals			
CHx_SEL	I	High	VID select signal for VID channel x (x = 0-15)
CH_ENA	I	—	Each bit is a VID enable signal for VID channel
DB_BUSY	I	High	Dual-boot busy signal (VID WBM only)
GPIO1	O	High	Drive the ASC write protect pin
VID_IRQ_O	O	High	Interrupt request signal (VID WBS only)
SLOW_CLOCK	I	—	Clock for counting the wait state between each VID process.

Register Descriptions

The LatticeMico VID WISHBONE Slave module has a register map to allow the service of the hardened functions through the WISHBONE bus interface read/write operations. Table 5 through Table 9 describe the register map of the VID WBS module.

Table 5: WISHBONE Addressable Registers for VID Module

Register Name	Register Function	Address	Access
CHx_INFO	Holds the info of ASC ID and CLT ID of channel x	0x0 - 0xF	Read
REQ_LOW	VID request holding register [7:0]	0x10	Read
REQ_HIGH	VID request holding register [15:8]	0x11	Read
IN_PROC_LOW	In Process status register [7:0]	0x12	Write
IN_PROC_HIGH	In Process status register [15:8]	0x13	Write
SLAVE_ADDR	Current ASC I^2C Slave Address	0x14	Read
DATA_LOW	VID data to be transmitted to ASC [7:0]	0x15	Read
DATA_HIGH	VID data to be transmitted to ASC [15:8]	0x16	Read
STATUS	Status of I^2C transfer	0x17	Write
IRQENR	Interrupt Request Enable	0x18	Read/Write

Table 6 through Table 9 provide details about each register in the LatticeMico VID.

Channel Information Data Register Definition - CHx_INFO

The WISHBONE host has Read-Only access to these registers.

Table 6: CHx_INFO Register Bit Definition

Bit	Field	Description
2:0	ASC ID	Identify the ASC0 to ASC7 Device
5:3	CLT ID	Identify the one of the 8 CLT Channel in each ASC
7:6	RSVD	Reserved Bit

Request Register Definition – REQ_LOW / REQ_HIGH

REQ_LOW / REG_HIGH are 8-bit registers, which combined to hold the VID channel request value. Each bit corresponding to a VID Channel, i.e. Bit 0 goes high means Channel 0 has triggered the request event. The WISHBONE host has Read-Only access to these registers.

REQ_LOW register holds the VID Request of the lower 8-bit value [7:0].

REG_HIGH register holds the VID Request of the upper 8-bit value [15:8].

In Process Register Definition – IN_PROC_LOW / IN_PROC_HIGH

IN_PROC_LOW / IN_PROC_HIGH are 8-bit registers, which combined to hold the In Process status. Each bit corresponding to a VID Channel, i.e. Bit 0 goes high means Channel 0 is in process. The WISHBONE host has Write-Only access to these registers.

IN_PROC_LOW register holds the In Process Register Status of the lower 8-bit value [7:0].

IN_PROC_HIGH register holds the In Process Register Status of the upper 8-bit value [15:8].

I2C Slave Address Register Definition – SLAVE_ADDR

The WISHBONE host has Read-Only access to these registers.

Table 7: SLAVE_ADDR Register Bit Definition

Bit	Field	Description
5:0	ASC ID	Holding the 7-bits I^2C Address
7:6	RSVD	Reserved Bit

VID Data Register Definition – DATA_LOW / DATA_HIGH

DATA_LOW / DATA_HIGH are 8-bit registers, which combined to hold the current data for transmitting to ASC device. The WISHBONE host has Read-Only access to these registers.

DATA _LOW register holds the current VID DATA of the lower 8-bit value [7:0].

DATA _HIGH register holds the current VID DATA of the upper 4-bit value [11:8].

The upper 4 bits value [15:12] of the register is default to 0

I2C Status Register Definition – STATUS

The WISHBONE host has Read and Write access to these registers expect bit 4 - DB_BUSY

Table 8: STATUS Register Bit Definition

Bit	Field	Description	Access
0	GPIO1	To activate or deactivate the GPIO1 bit 1 – Activated, 0 - Deactivated	Read/Write
1	DONE	Completion of a successful VID request transfer 1 – Completed, 0 - Incomplete	Read/Write
2	FAIL	Incomplete VID request transfer 1 – Incomplete, 0 - Completed	Read/Write
3	NACK	I^2C NACK is received 1 – NACK received, 0 – ACK received	Read/Write

Table 8: STATUS Register Bit Definition (Continued)

Bit	Field	Description	Access
4	DB_BUSY	LatticeMico Dual Boot is busy 1 – Dual Boot is Busy, 0 – Dual Boot is Free	Read
7:4	RSVD	Reserved Bit	N/A

Interrupt Enable Register Definition – IRQENR

The WISHBONE host has Read and Write access to these registers.

Table 9: IRQENR Register Bit Definition

Bit	Field	Description
0	IRQEN	VID Interrupt Enable
7:1	RSVD	Reserved Bit

LatticeMico8 Microprocessor Software Support

This section describes the LatticeMico8 microcontroller software support provided for the LatticeMico VID component.

Device Driver

The VID device driver interacts directly with the VID instance. This section describes the limitations, type definitions, structure, and functions of the VID device driver.

Type Definitions

This section describes the type definitions for the VID device context structure. This structure, shown in Figure 5, contains the VID component instance-specific information and is dynamically generated in the DDStructs.h header file. This information is largely filled in by the managed build process by extracting the VID component-specific information from the platform specification file. As part of the managed build process, designers can choose to control the size of the generated structure, and hence the software executable, by selectively enabling some of the elements in this structure via C preprocessor macro definitions. These C preprocessor macro definitions

are explained later in this document. You should not manipulate the members directly, because this structure is for exclusive use by the device driver. Table 10 describes the parameters of the VID device context structure shown in Figure 5.

Device Context Structure

Figure 5 shows the VID device context structure.

Figure 5: VID Device Context Structure

```
struct st_MicoVIDCtx_t {
    const char *   name;
    size_t   base;
    unsigned char   intrLevel;
    unsigned char   master_mode;
    unsigned char   curr_channel;
    unsigned char   max_channel;
    void *   p_efb;
    void *   p_mutex;
    unsigned char   i2c_mutex;
} MicoVIDCtx_t;
```

Table 10 describes the VID device context parameters.

Table 10: VID Device Context Parameters

Parameter	Data Type	Description
name	const char*	VID instance name (entered in MSB)
base	size_t	MSB-assigned base address for this instance
intrLevel	unsigned char	Processor interrupt line to which this instance is connected
master_mode	unsigned char	This value is 1 if the VID is configured as Master mode. Otherwise, it is Slave Mode
curr_channel	unsigned char	This value specific the current channel
max_channel	unsigned char	This value specific maximum number of channel is used
p_efb	void *	This value points to the EFB instance used by VID
p_mutex	void *	This value points to the Mutex instance used by VID
i2c_mutex	unsigned char	This value specific the Mutex owner ID for I^2C communication protocol

C Preprocessor Macro Definitions

This section describes the C preprocessor macro definitions that are available to the software developer. There are two types of macro definitions: 'object-like' and 'function-like'.

The 'object-like' macro definitions do not take any arguments and are used to control the size of the generated application executable. There are three ways an 'object-like' macro definition can be used by the software developer.

1. Manually adding the -D<macro name> option to the compiler's command line in the application's 'Build Properties'. Refer to the LatticeMico8 Developer User Guide for more information on how to manually add the macro definition in the application's 'Build Properties' GUI.
2. Automatically adding the -D<macro name> option to the compiler's command-line in the application's 'Build Properties' by enabling the 'check-box' associated with the macro definition. Refer to the LatticeMico8 Developer User Guide for more information on how to set up the check/ uncheck the macro definitions in the application's 'Build Properties' GUI.
3. Manually adding the macro definition to the C code using the following syntax:

   ```
   #define <macro name>
   ```

It is recommended that the developer use option 1 or 2.

- __MICOVID_ENABLE_MUTEX__

  This preprocessor macro definition enables code and data structures for LatticeMico8 Mutex within LatticeMico VID device driver and application. It is not defined by default.

- __MICOVID_USER_IRQ_HANDLER__

  This preprocessor macro definition disables code and data structures within the device driver that allow the user to define the custom interrupt routine, the default routine will be disabled. It is not defined by default.

Table 11: C Preprocessor Function-like Macros For VID

Macro Name	Second Argument to Macro / Third Argument to Macro (if exist.	Description
MICO_VID_READ_CHX_INFO	The 8-bit value read from the channel info / channel number	This macro reads a character from the channel info register with a specific channel number
MICO_VID_READ_REQ_LOW	The 8-bit value read from the lower byte of channel request register.	This macro reads a character from the lower byte of channel request register
MICO_VID_READ_REQ_HIGH	The 8-bit value read from the upper byte of channel request register.	This macro reads a character from the upper byte of channel request register
MICO_VID_WRITE_IN_PROC_LOW	The 8-bit value writes to the lower byte of In Process register.	This macro writes a character to the lower byte of In Process register
MICO_VID_WRITE_IN_PROC_HIGH	The 8-bit value writes to the upper byte of In Process register.	This macro writes a character to the upper byte of In Process register
MICO_VID_READ_SLAVE_ADDR	The 8-bit value read from the Slave address register	This macro reads a character from the I^2C slave address register

Table 11: C Preprocessor Function-like Macros For VID (Continued)

Macro Name	Second Argument to Macro / Third Argument to Macro (if exist.	Description
MICO_VID_READ_DATA_LOW	The 8-bit value read from the lower byte of current VID data register.	This macro reads a character from the lower byte of VID data register
MICO_VID_READ_DATA_HIGH	The 8-bit value read from the upper byte of current VID data register.	This macro reads a character from the upper byte of VID data register
MICO_VID_READ_STATUS	The 8-bit value read to the status register.	This macro reads a character to the I^2C status register
MICO_VID_WRITE_STATUS	The 8-bit value writes to the status register.	This macro writes a character to the I^2C status register
MICO_VID_READ_IRQENR	The 8-bit value read from the Interrupt Enable register.	This macro reads a character from the Interrupt Enable register
MICO_VID_READ_CHX_INFO	The 8-bit value read from the channel info / channel number	This macro reads a character from the channel info register with a specific channel number
MICO_VID_READ_REQ_LOW	The 8-bit value read from the lower byte of channel request register.	This macro reads a character from the lower byte of channel request register
MICO_VID_READ_REQ_HIGH	The 8-bit value read from the upper byte of channel request register.	This macro reads a character from the upper byte of channel request register
MICO_VID_WRITE_IN_PROC_LOW	The 8-bit value writes to the lower byte of In Process register.	This macro writes a character to the lower byte of In Process register
MICO_VID_WRITE_IN_PROC_HIGH	The 8-bit value writes to the upper byte of In Process register.	This macro writes a character to the upper byte of In Process register
MICO_VID_WRITE_IRQENR	The 8-bit value writes to the Interrupt Enable register	This macro writes a character to the Interrupt Enable register

Note: The first argument to the macro is the VID address.

Functions

This section describes the implemented device-driver-specific functions.

MicoVIDInit Function

```
void MicoVIDInit (MicoVIDCtx_t *ctx);
```

This is the VID initialization function. Table 12 describes the parameter in the MicoVIDInit function syntax.

Table 12: MicoVIDInit Function Parameter

Parameter	Description
MicoVIDCtx_t	Pointer to a valid MicoVIDCtx _t structure representing a valid VID instance.

MicoVIDRegisterEFB Function

```
void MicoVIDRegisterEFB (MicoVIDCtx_t *ctx,
                         MicoEFBCtx_t *p_efb);
```

This function registers an EFB instance into the VID instance. This EFB will be used for the interface between VID control and the ASC device.

Table 13 describes the parameters in the MicoVIDRegisterEFB function syntax.

Table 13: MicoVIDRegisterEFB Function Parameter

Parameter	Description
MicoVIDCtx_t	Pointer to a valid MicoVIDCtx_t structure representing a valid VID instance.
MicoEFBCtx_t	Pointer to a valid MicoEFBCtx_t structure representing a valid EFB instance.

MicoVIDRegisterEFBnMutex Function

```
void MicoVIDRegisterEFBnMutex (MicoVIDCtx_t *ctx,
                         MicoEFBCtx_t *p_efb,
                         MicoMutexCtx_t *p_mutex,
                         unsigned char i2c_mutex_id);
```

This function registers an EFB instance and a Mutex instance into the VID instance. This EFB will be used for the interface between VID control and the ASC device, and Mutex will be used for controlling the EFB I^2C resources to avoid collision.

Table 14 describes the parameters in the MicoVIDRegisterEFBnMutex function syntax.

Table 14: MicoVIDRegisterEFBnMutex Function Parameter

Parameter	Description
MicoVIDCtx_t	Pointer to a valid MicoVIDCtx_t structure representing a valid VID instance.
MicoEFBCtx_t	Pointer to a valid MicoEFBCtx_t structure representing a valid EFB instance.

Table 14: MicoVIDRegisterEFBnMutex Function Parameter (Continued)

Parameter	Description
MicoMutexCtx	Pointer to a valid MicoMutexCtx_t structure representing a valid Mutex instance.
unsigned char	Mutex owner ID for I^2C communication protocol

MicoVIDProcessRequest Function

```
char MicoVIDProcessRequest (MicoVIDCtx_t *ctx,
                            unsigned char channel);
```

This function process the VID Request with an user specific channel. User has to check which VID channel has triggered the event by reading the VID_REQ register. Error code will return when the request process is incomplete.

Table 15 describes the parameter in the MicoVIDProcessRequest function syntax.

Table 15: MicoVIDProcessRequest Function Parameter

Parameter	Description
MicoVIDCtx_t	Pointer to a valid MicoVIDCtx_t structure representing a valid VID instance.
unsigned char	Channel request to be processed.

Table 16 describes the values returned by the MicoVIDProcessRequest Function.

Table 16: Values Returned by the MicoVIDProcessRequest Function

Parameter	Description
0	Process successful
-1	No request for this channel
-2	Failed to receive ACK during I^2C
-3	Failed to verify the ASC device data

MicoVIDISR Function

```
void MicoVIDISR (MicoVIDCtx_t *ctx);
```

This is the VID Interrupt handler. The interrupt routine is implemented in the software driver. User has an option to use the default interrupt handler or implement a custom interrupt routine.

Table 17 describes the parameter in the MicoVIDISR function syntax.

Table 17: MicoVIDISR Function Parameter

Parameter	Description
MicoVIDCtx_t	Pointer to a valid MicoVIDCtx_t structure representing a valid VID instance.

Software Usage Example

This section provides an example of using the VID. The example is shown in Figure 6 and assumes the presence of a VID component named “vid”, a Mutex component named “mutex” and an EFB component named “efb.

Figure 6: VID Software Example

```
#include "MicoUtils.h"
#include "DDStructs.h"
#include "MicoEFB.h"
#include "MicoVID.h"

int main(void){
MicoVIDCtx_t *vid = &vid_vid;
MicoEFBCtx_t *efb = &efb_efb;

#ifdef __MICOVID_ENABLE_MUTEX__
MicoMutexCtx_t *mutex = &mutex_mutex;
if(mutex == 0){
   return(-1);
}
// Register the EFB and Mutex instance into VID
MicoVIDRegisterEFBnMutex(vid, efb, mutex, 0);
# else
// Register the EFB instance into VID
MicoVIDRegisterEFB(vid, efb);
#endif

if(vid == 0 || efb == 0){
   return(-1);
}

#ifndef __MICO_NO_INTERRUPTS__
// Enable the interrupt
MICO_VID_WRITE_IRQENR( vid->base , MICO_VID_ENABLE_IRQ);
#else
unsigned char curr_channel = 0x0;
do{
if(curr_channel < vid->max_channel){
MicoVIDProcessRequest(vid, curr_channel);
curr_channel++;
}else{
curr_channel = 0;
}
MicoSleepMicroSecs (50);
}while (1);
#endif

return(0);
}
```

Revision History

Component Version	Description
1.0	Initial Release.
1.1	▶ Added WAIT_COUNT parameter to add a wait state between each successive VID transactions. ▶ Improved the hardware protect capability in the VID Master Mode. ▶ Fixed VID Write Status Register issue.

Trademarks