Hello-FPGA BISS-C Master IP User Manual

Hello-FPGA
INFO@HELLO-FPGA.COM

目录

Hello-FPGA BISS-C Master User Manual

1 BISS-C INTRODUCTION

BISS (Bi-Synchronous Serial interface) is a type of serial communication protocol used in sensors and actuators for high-precision measurement and control applications. It is a synchronous protocol that uses a clock signal to synchronize the data transmission between the master and slave devices.

The BISS protocol has two variants: BISS-C and BISS-B. BISS-C is a unidirectional protocol used for transmitting measurement data from the slave to the master device, while BISS-B is a bidirectional protocol used for both transmitting and receiving data.

In BISS, the slave device sends a serial data stream to the master device, which includes position, velocity, or other measurement data. The master device sends a clock signal to the slave device, which uses it to synchronize the data transmission. BISS-B also includes an acknowledgment signal from the master device to the slave device to confirm that data has been received correctly.

BISS has several advantages over other serial communication protocols, including **high accuracy, low latency, and high noise immunity**. However, it is a complex protocol and requires specialized hardware to implement. Our IP is a FPGA logic which can communicate well with the BISS-C sensors or actuators.

1.1 DATA FORMAT AND TIMING

BiSS C-mode (unidirectional) is a fast synchronous serial interface for acquiring position data from an encoder. It is a master-slave interface. The master controls the timing of position acquisition and the data transmission speed, and the encoder is the slave.

The interface consists of two unidirectional differential pairs of lines:

- “MA” transmits position acquisition requests and timing information (clock) from master to encoder
- “SLO” transfers position data from encoder to master, synchronized to MA.

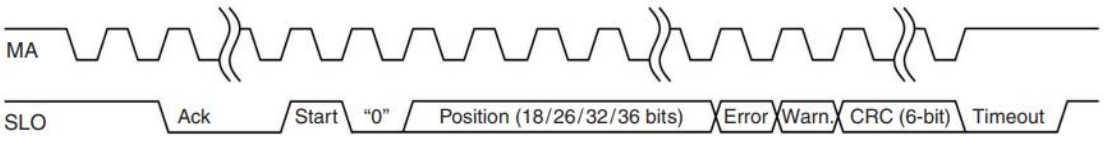

图 1-1 BISS-C data format

The master-slave signal communication format is RS485/RS422 differential line-driven.

A typical request cycle proceeds as follows:

1. When idle, the master holds MA high. The encoder indicates it is ready by holding SLO high.
2. The master requests position acquisition by starting to transmit clock pulses on MA.

3. The encoder responds by setting SLO low on the second rising edge on MA.
4. After the “Ack” period is complete, the encoder transmits data to the master synchronized with the clock as shown in the diagrams above.
5. When all data has been transferred, the master stops the clock and sets MA high.
6. If the encoder is not yet ready for the next request cycle, it sets SLO low (the Timeout period).
7. When the encoder is ready for the next request cycle, it indicates this to the master by setting SLO high.

Description of data

- Ack This is the period during which the valid information transmitted back.
- Start and “0” (1 bit each) The encoder transmits the start bit to signal to the master that it is starting to transmit data. The start bit is always high and the “0” bit is always low.
- Position (18, 26, 32 or 36 bits) The absolute position data is in binary format and sent MSB first. For rotary encoders, there are exactly 2n counts per revolution, after which the count “wraps around” to zero. [Lower resolutions may be achieved by ignoring the least significant bit(s) of the position data.]
- Error (1 bit) The error bit is active low.
- Warning (1 bit) The warning bit is active low.
- CRC for position data (6 bit) The CRC polynomial for position, error and warning data is: $x^6 + x^1 + x^0$. It is transmitted MSB first and inverted. The start bit and “0” bit are omitted from the CRC calculation.
- Timeout RESOLUTE encoders are capable of acquiring a new position reading every 40 µs (a maximum request cycle rate of 25 kHz). Therefore 40 µs must elapse between the start of one request cycle and the start of the next.

1.2 IP FEATURES

- 1-N parallel BISS-C sensors, the cable length difference can be compensated by the IP itself.
- Valid position data resolution is 1-32.
- AXI4-Lite Interface for register access.
- AXI-Stream valid data output.
- Reconfigurable valid data resolution.
- Internal CRC-6 can be enabled or disabled.
- Single point, finite, continuous sample mode, internal or external sample clock.

2 PRODUCT SPECIFICATION

2.1 OVERVIEW

The following figure shows the top level block diagram of the IP exposing all the relevant

interfaces. The SW interface to the IP is via AXI-Lite. The valid position data is via AXI Stream Interface. Error and Warn status can used to monitor the sensor's state. External sample clock can used to synchronize with different devices.

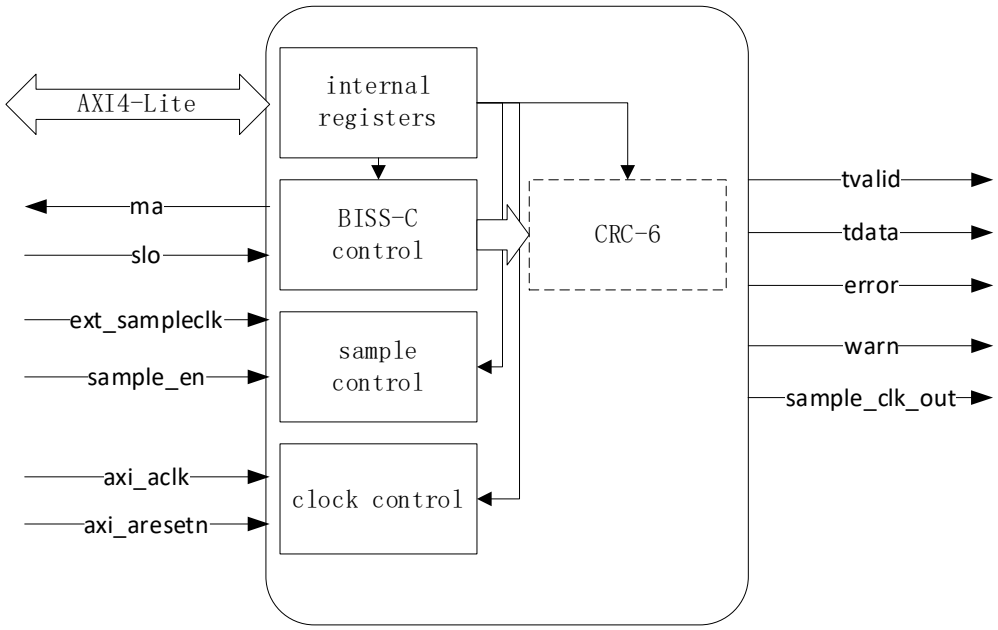

图 2-1 BISS-C master block diagram

2.2 PARAMETERS

Table 1 User configurable parameters

Name	Value	Description
BISS_CHANNEL_WIDTH	1-N	The channel number of BISS-C

2.3 PORT DESCRIPTIONS

2.3.1 AXI Lite Interface Ports

Table 2 SW AXI-Lite Interface Ports

Port Name	I/O	Width	Description
axi_aclk	I	1	AXI Lite clock, the main clock of the IP
axi_aresetn	I	1	Active low synchronous AXI Lite reset
S_AXI_*	-	-	See the Vivado Design Suite: AXI Reference Guide (UG1037) for the description of AXI4 signals.

2.3.2 AXI Stream Interface Ports

Table 3 HW AXI Stream Interface Ports

Port Name	I/O	Width	Description
s_axis_tdata	I	32*BISS_CHANNEL_WIDTH	Decoded valid position data, [31:0] is the channel 0 position data, if the encoder's resolution is not 32bit, the high bits will be zero. [63:32] is channel 1 position data.
s_axis_tvalid	I	1	Active high, indicate the s_axis_tdata is valid data, active one clock cycle.
s_axis_tlast	-	1	Active high, indicate the s_axis_tdata is the last data(actually it will be always active).

2.3.3 BISS-C

Table 4 BISS-C Interface Ports

Port Name	I/O	Width	Description
ma	O	BISS_CHANNEL_WIDTH	master clock "MA" transmits position acquisition requests and timing information (clock) from master to encoder
slo	I	BISS_CHANNEL_WIDTH	slave output "SLO" transfers position data from encoder to master, synchronized to MA.

This is a standard BISS-C serial interface.

2.3.4 Sample clock

Table 5 BISS-C sample clock Ports

Port Name	I/O	Width	Description
ext_sample_clk	O	1	External sample clock for the IP, sample means the period which one position data is read.
sample_clk_output	O	1	Sample clock output
sample_en	I	1	Active high, sample start

Sample clock means the IP will read one complete position data from the sensors at it's rising edge.

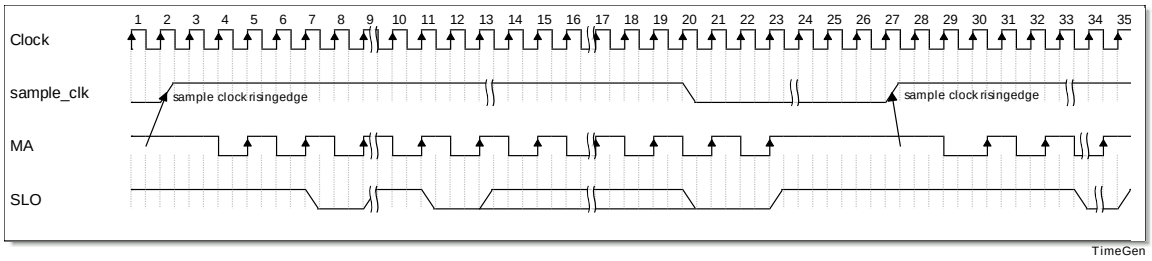

Figure 2-1 sample_clk timing

The sample clock can route from internal divider or external sample clock pins.

2.3.5 Others

Table 6 others Ports

Port Name	I/O	Width	Description
error	O	1	error mask, active 0
warn	O	1	/warn mask, active 0

2.4 REGISTER DESCRIPTION

2.4.1 Registers

Address(hex)	R/W	Width	default	range	Register
0x00	R/W	1	0	0/1	Soft reset, active high
0x04	R/W	1	0	0/1	External sample clock enable, 1: external sample clock; 0: internal sample clock(divided by axi_aclk)
0x08	R/W	1	0	0/1	Single point mode enable, 1: single point mode, only read the sensor's position data when SW request. 0: sample controlled by the sample clock, read the sensor's position data at the sample clock rising edge.
0x0c	R/W	1	0	0/1	Single point position data read enable, active rising edge. It will lead

					the IP send ma clock to the sensor, so user can get a new and complete position data.
0x10	R/W	28	0	1-268,435,455	Internal sample clock divider. 1000 means the internal sample clock is divided 2000 from the axi_aclk.
0x14	R/W	8	0	1-255	MA clock divider. 4 means the MA is divided 8 from the axi_aclk.
0x18	R/W	8	0	1-32	Resolution bits. The valid bit in the sensor. 1-32
0x1c	R/W	1	0	0/1	Ignore crc check. 0: do not check the crc value; 1: check the crc value.
0x20	R/W	1	0	0x00-0xffff	channel enable. Bit enable. 0x3 means channel 0 and channel 1 enabled.
0x24	R/W	1	0	0x00-0xff	channel index. 0x2 means current channel is channel 2.
0x28	R/W	BISS_CHANNEL_WIDTH	0	0/1	Shift MA clock enable or disable. Bit enable.
0x2c	R	6(CRC6 only)	0	-	shifted_crc_data_debug, CRC data from the sensors.
0x30	R	6(CRC6 only)	0	-	crc_check_value_debug, the internal CRC data calculated by the internal CRC module.
0x34	R	BISS_CHANNEL_WIDTH	0	0x0-0xff	single_point_data_valid, if the single point data is valid or invalid, this signal will keep until hardware or software reset happened.
0x38	R	32	0	-	single_point_data
0x3c	R	10	0	0x0-0x3ff	ma_slo_delay_ticks_debug, the output ma clock and the echo ma clock delay ticks(1 tick is 1 period of the aclk)

2.5 SOFTWARE DRIVER

Next below are the software driver API, user can use these API to write their own software.

```
// ********************************************************************************
/// <summary>
/// initial configure the biss ip
/// </summary>
/// <param name="hdev">register configuration device handle</param>
/// <returns>0: sucess;others: faild</returns>
// ********************************************************************************
int BISS_C_Init(Int hdev);

// ********************************************************************************
/// <summary>
/// ignore crc check for the data
/// </summary>
/// <param name="hdev">register configuration device handle</param>
/// <param name="ignoreCRC">1: ignore the internal crc check</param>
/// <returns>0: sucess;others: faild</returns>
// ********************************************************************************
int BISS_C_IgnoreCRC(Int hdev, int ignoreCRC);

// ********************************************************************************
/// <summary>
/// this is for debug, read back the MA delay ticks
/// </summary>
/// <param name="hdev">register configuration device handle</param>
/// <param name="channelIndex">chanel index</param>
/// <param name="maSloDelayTicks">delayed ticks of ma clock</param>
/// <returns>0: sucess;others: faild</returns>
// ********************************************************************************
int BISS_C_CheckAjustDelay(Int hdev, int channelIndex, int *maSloDelayTicks );

// ********************************************************************************
/// <summary>
/// config the sample clock of the biss-c ip
/// </summary>
/// <param name="hdev">register configuration device handle</param>
/// <param name="highlowTicksNum">the divider parameters of the divider</param>
/// <param name="isExtSampleClk">1: external sample clock</param>
/// <returns>0: sucess;others: faild</returns>
// ********************************************************************************
int BISS_C_ConfigSampleClk(Int hdev, int highlowTicksNum, bool isExtSampleClk);

// ********************************************************************************
/// <summary>
/// MA signal divider, user can change the ma clock frequency
```

```
/// </summary>
/// <param name="hdev">register configuration device handle</param>
/// <param name="highlowTicksNum">the divider parameters of the divider</param>
/// <returns>0: sucess;others: faild</returns>
// ********************************************************************************
int BISS_C_ConfigMAClk(Int hdev, int highlowTicksNum);

// ********************************************************************************
/// <summary>
///
/// </summary>
/// <param name="hdev"></param>
/// <param name="bitsNum"></param>
/// <returns>0: sucess;others: faild</returns>
// ********************************************************************************
int BISS_C_ConfigPositionDatabits(Int hdev, int bitsNum);

// ********************************************************************************
/// <summary>
/// enable the channels
/// </summary>
/// <param name="hdev">register configuration device handle</param>
/// <param name="channelEnable">active low, bit active, 0x03 means channel 0 and channel 1
are enabled</param>
/// <returns>0: sucess;others: faild</returns>
// ********************************************************************************
int BISS_C_ConfigChannels(Int hdev, int channelEnable);

// ********************************************************************************
/// <summary>
/// read back single point data of the sensors
/// </summary>
/// <param name="hdev">register configuration device handle</param>
/// <param name="channelIndex">0:means the chnanel index</param>
/// <param name="data">sigle point data of the specified channel dara</param>
/// <param name="isValid">check if the dara is valid or not</param>
/// <returns>0: sucess;others: faild</returns>
// ********************************************************************************
int BISS_C_ReadSinglePoint(Int hdev, int channelIndex, unsigned int* data, bool* isValid);
```

Annex A：参考资料

1、CoaXPess JIIA CXP-001-2021
2、https://www.cnblogs.com/xingce/category/2165251.html