WISHBONE I2C IP CORE

USER MANUAL

Faculty of Electrical Engineering, Banja Luka

Table of Contents

1. Introduction

This document is meant to be a guide for future users of *Wishbone I2C IP CORE*. This document will explain how to access, configure and use I2C IP Core.

Wishbone I2C IP Core is designed to enable connection of Wishbone Transaction Generator on one side and I2C Test Device on the other side. Wishbone I2C IP Core is responsible for communication between them. Wishbone Transaction Generator sends data and comands that will let I2C IP Core know what to do with data (store into TX Fifo Buffer, read from RX Fifo Buffer, Start Transaction, ...). Also Wishbone Transaction Generator is used to configure I2C IP Core.

2. Specification

- Supported receive and transmit FIFO buffer with depth of 16 elements
- Desing is compatible with I2C Standard
- Supported Master and Slave mode
- Supported generation and detection of ACK bit
- Supported generation and detection of START Condition
- Supported generation and detection of STOP Condition
- Supported generation and detection of Repeated START Condition
- All registers are 32 bit width
- Supported logic for generating and controlling interrupt signals
- Supported only half-duplex communication
- Supported detection of arbitration lost and NACK bit
- Maximum system clock speed is 168 MHz
- Communication speeds according to the following standard are supported*: Fast-Mode Plus* 1MHz, *Fast-Mode* 400kHz i *Standard Mode* 100kHz
- Supported generation of GPO (General Purpose Output) signal which width is 1 – 8 bits
- Supported 7 bits and 10 bits slave address length
- Minimum system clock speed is 1 MHz
- Operating temperature 0 – 85 ℃

3. Detailed architecture

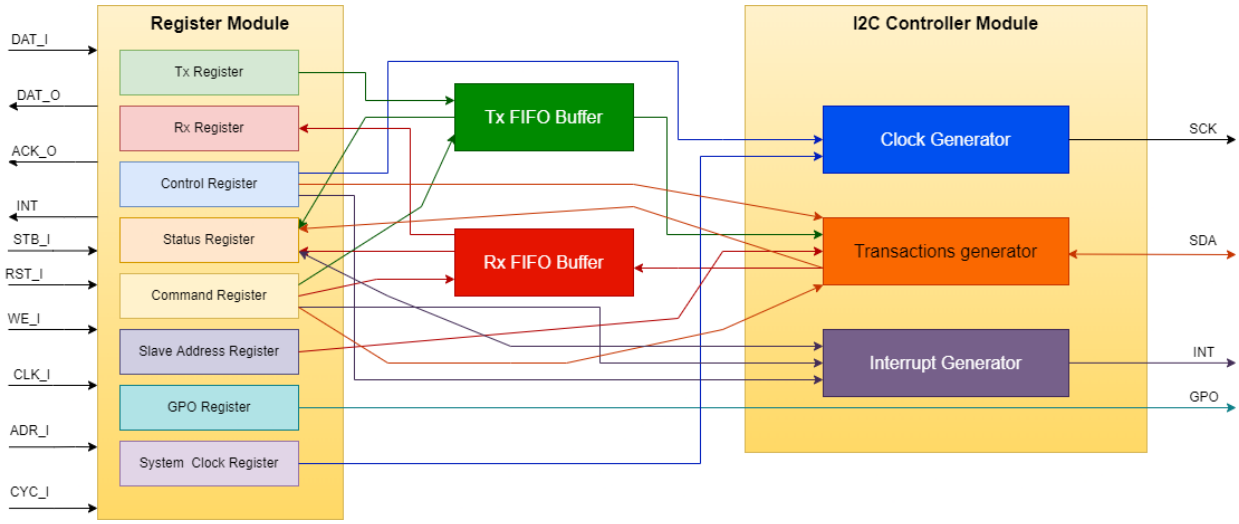


Image 1. Detailed architecture of the Wisbone I2C IP Core.

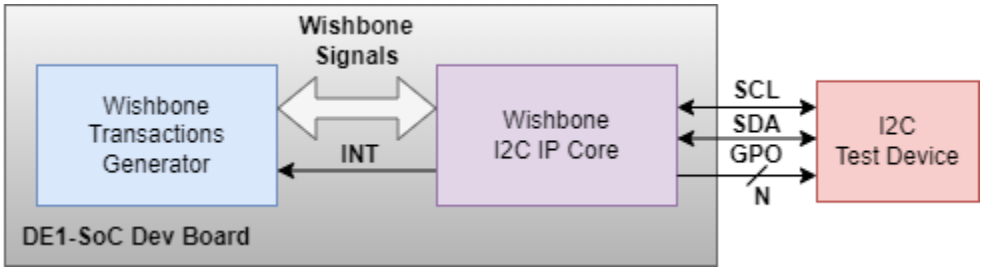


Image 2. System architecture verification.

4. Wishbone I2C IP Core – Signals

This section contains description of all the signals that are necessary for connecting Wishbone I2C IP Core to other devices.

4.1 Wisbone Signals

Signal name	Width	Type	Description
clk_i	1	Input	System clock
addr_i	8	Input	Adress of register
we_i	1	Input	Enable write to registers
rst_i	1	Input	Reset signal
dat_i	32	Input	Input Data
dat_o	32	Output	Output Data
int	1	Output	Interrrupt signal
cyc_i	1	Input	When asserted, indicates that valid bus cycle is in progress
stb_i	1	Input	When asserted, indicates that slave is selected
ack_o	1	Output	Acknowledgne signal

4.2 I2C Interface Signals

Signal name	Width	Type	Description
SCL	1	Inout	Serial Clock Line
SDA	1	Inout	Serial Data Line
GPO	1-8	Output	General Purpose Output

5. Registers

This section contains explanation about Wishbone I2C IP Core registers and their parameters that user can configure according to their needs.

Register name	Address	Width	Description
Tx Register	0x00	32	Transmit data register
Rx Register	0x01	32	Receive data register
Control Register	0x02	32	Control register
Status Register	0x03	32	Status register
Command Register	0x04	32	Command register
Slave Address	0x05	32	Slave address register
GPO Register	0x06	32	General Purpose Output register
System Clock Register	0x07	32	System Clock register

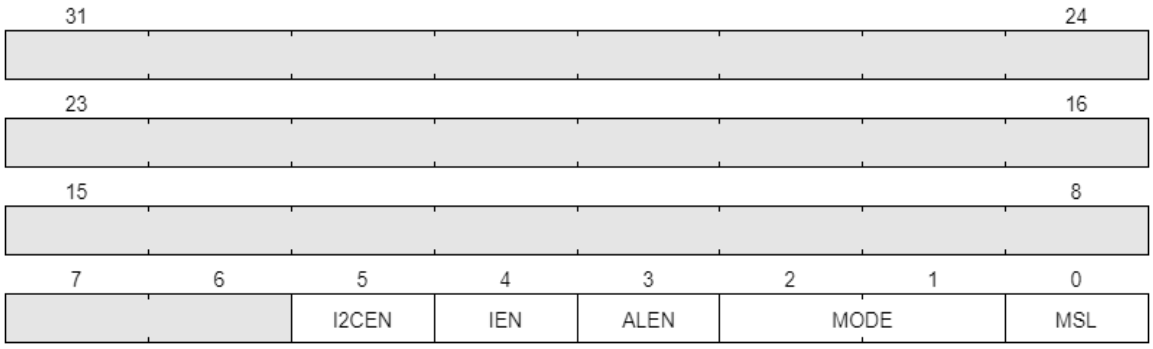
Tx Register contains data for transmit. In this register only 8 lower bits are valid.

Rx Register contains received data. In this register only 8 lower bits are valid.

GPO (General Purpose Output) register containt data which will be send on GPO. In this register only 1-8 lower bits are valid. Number of valid bits is configurable.

5.1 Control Register

Control register allows to enable I2C communication, enable interrupt, select speed mode, select master or slave mode and choice slave address length (7 or 10 bits).



bit 5 **I2CEN** : I2C Enable

1 = I2C Enabled

0 = I2C Disabled

bit 4 **IEN** : Interrupt Enable

1 = Interrupt Enabled

0 = Interrupt Disabled

bit 3 **ALEN**: Slave Address Length

1 = 10 bit adress

0 = 7 bit adress

bit 2:1 **MODE**: Mode Select

00 = Standard Mode 100 KHz

01 = Fast Mode 400 KHz

10 = Fast Plus Mode 1 MHz

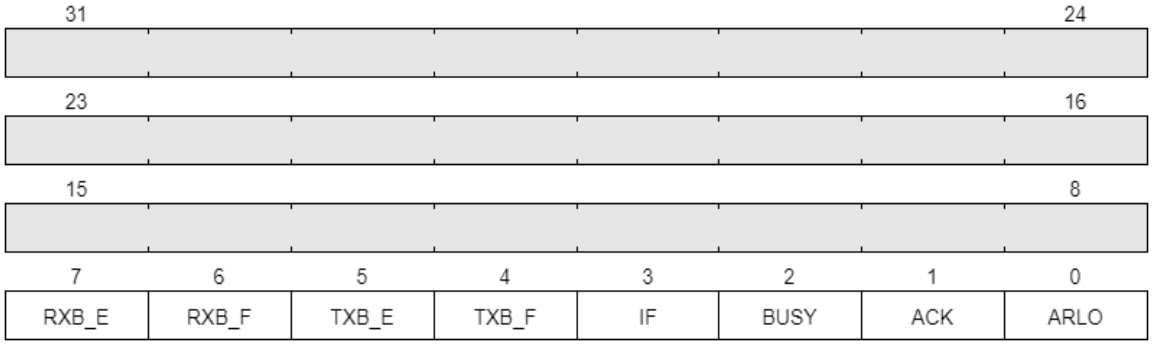
bit 0 **MSL** : Master/slave select

0 = Master

1 = Slave

5.2 Status Register

Status register is read only register. User can only read info data from this register.



bit 7 **RXB_E**: Rx Buffer Empty

1 = Rx Buffer Empty

0 = Rx Buffer Not Empty

bit 6 **RXB_F**: Rx Buffer Full

1 = Rx Buffer Full

0 = Rx Buffer Not Full

bit 5 **TXB_E**: Tx Buffer Empty

1 = Tx Buffer Empty

0 = Tx Buffer Not Empty

bit 4 **TXB_F**: Tx Buffer Full

1 = Tx Buffer Full

0 = Tx Bufffer Not Full

bit 3 **IF**: Interrupt Flag

1 = Interrupt detected

0 = Interrupt not detected

bit 2 **BUSY**: Indicates That I2C Bus is Busy

1 = Start Condition Detected

0 = Stop Condition Detected

bit 1 **ACK**: RxACK

1 = No ACK Received

0 = ACK Received

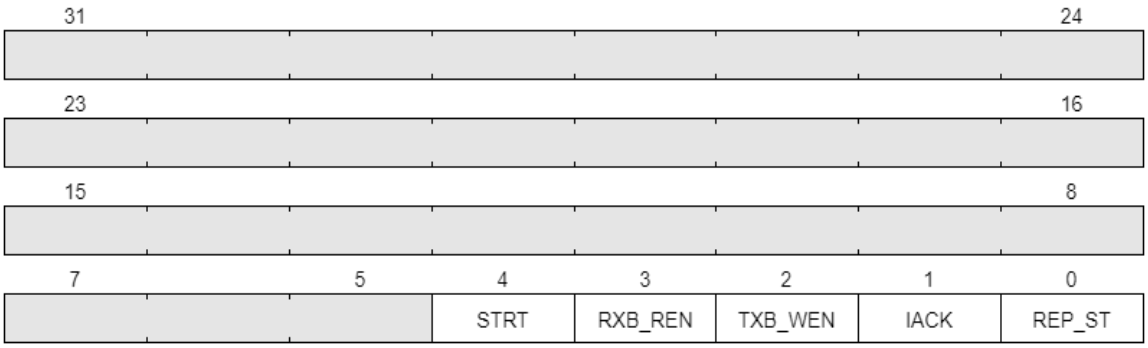
bit 0 **ARLO**: Arbitration Lost

1 = STOP is detected but not requested, Master drives SDA HIGH but SDA is LOW

0 = No Arbitration Lost

5.3 Command Register

Command register is read/write register.



bit 4 **STRT**: Start I2C Transaction

When asserted, indicates start of I2C Transaction

bit 3 **RXB_REN**: Rx Buffer Read Enable

When asserted, reading from Rx Buffer is enabled

bit 2 **TXB_WEN**: Tx Buffer Write Enable

When asserted, writing to Tx Buffer is enabled

bit 1 **IACK**: Clear pending Interrupt

When asserted, interrupt is cleared

bit 0 **REP_ST**: Repeated Start

When asserted, indicates Repeated Start

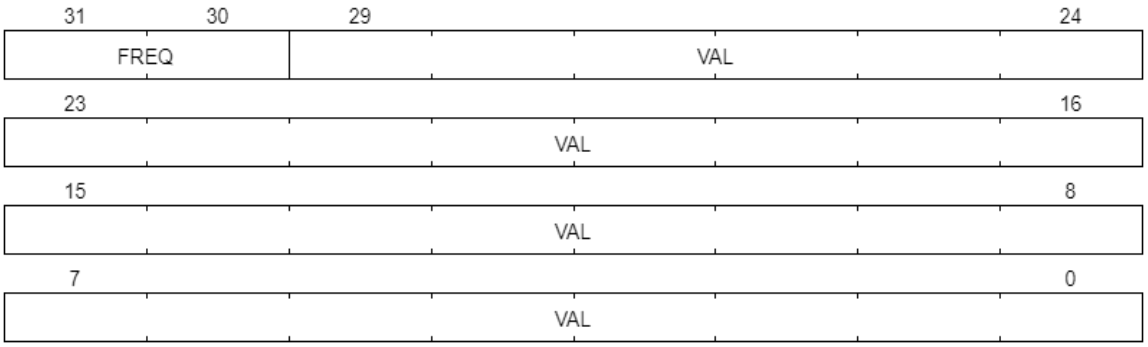
5.4 Slave Address Register

Slave Address Register contains device address when Wishbone I2C IP Core is in slave mode.

bit 9:0 **SAD**: Slave Address

5.5 System Clock Register

System Clock Register contains info about system clock value in Hz.



bit 31:30 **FREQ**: Signal Frequency

00 = Hz

01 = KHz

10 = MHz

11 = GHz

bit 29:0 **VAL:** Clock Signal Value in The Aforementioned Frequency

6. Examples

6.1 Master Mode - Write to Slave

```
wishbone_write(CONTROL_REG, I2C_MASTER_ENABLE);
wishbone_write(TX_REG, DATA_SIZE);
wishbone_write(COMMAND_REG, TX_ENABLE);
wishbone_write(COMMAND_REG, DISABLE_ALL);
wishbone_write(TX_REG, SLAVE_ADDRESS);
wishbone_write(COMMAND_REG, TX_ENABLE);
wishbone_write(COMMAND_REG, DISABLE_ALL);
wishbone_write(TX_REG, DATA);
wishbone_write(COMMAND_REG, TX_ENABLE);
wishbone_write(COMMAND_REG, DISABLE_ALL);
wishbone_write(COMMAND_REG, I2C_START);
wishbone_write(COMMAND_REG, DISABLE_ALL);
```

6.2 Master Mode – Read from Slave

```
wishbone_write(CONTROL_REG, I2C_MASTER_ENABLE);
wishbone_write(TX_REG, DATA_SIZE);
wishbone_write(COMMAND_REG, TX_ENABLE);
wishbone_write(COMMAND_REG, DISABLE_ALL);
wishbone_write(TX_REG, SLAVE_ADDRESS);
wishbone_write(COMMAND_REG, TX_ENABLE);
wishbone_write(COMMAND_REG, DISABLE_ALL);
wishbone_write(COMMAND_REG, I2C_START);
wishbone_write(COMMAND_REG, DISABLE_ALL);
// Wait for Data
wishbone_write(COMMAND_REG, RX_ENABLE);
wishbone_write(COMMAND_REG, DISABLE_ALL);
wishbone_read(RX_REG)
```

6.3 Slave Mode – Read

```
wishbone_write(SLAVE_ADDR_REG, SLAVE_ADDRESS);
wishbone_write(CONTROL_REG, I2C_SLAVE_ENABLE);
wishbone_write(TX_REG, DATA_SIZE);
wishbone_write(COMMAND_REG, TX_ENABLE);
wishbone_write(COMMAND_REG, DISABLE_ALL);
wishbone_write(TX_REG, DATA);
wishbone_write(COMMAND_REG, TX_ENABLE);
wishbone_write(COMMAND_REG, DISABLE_ALL);
wishbone_write(COMMAND_REG, I2C_START);
wishbone_write(COMMAND_REG, DISSABLE_ALL);
```

6.4 Slave Mode – Write

```
wishbone_write(SLAVE_ADDR_REG, SLAVE_ADDRESS);
wishbone_write(CONTROL_REG, I2C_SLAVE_ENABLE);
wishbone_write(TX_REG, DATA_SIZE);
wishbone_write(COMMAND_REG, TX_ENABLE);
wishbone_write(COMMAND_REG, DISABLE_ALL);
wishbone_write(COMMAND_REG, I2C_START);
// Wait until STOP Condition
wishbone_write(COMMAND_REG, RX_ENABLE);
wishbone_write(COMMAND_REG, DISABLE_ALL);
wishbone_read(RX_REG)
```