# I2S Controller



## Introduction

The I2S bus (Inter-IC Sound bus) is a 3-wire, half-duplex serial link for connecting digital audio devices in an electronic system. The bus handles audio data and clocks separately to minimize jitter that may cause data distortion in the digital analog system. Invented by Philips Semiconductor, the I2S bus is widely used by equipment and IC manufacturers.

This document describes the implementation of I2S Controller.

The design is implemented in Verilog language. The Lattice iCEcube2™ Place and Route tool integrated with the Synopsys Synplify Pro® synthesis tool is used for the implementation of the design. The design can be targeted to other iCE40™ FPGA product family devices.

## Features

- Configurable as an I2S transmit master or I2S receive master
- Parameterized data width from eight to 32 bits

## System Block Diagram

The SD Host Controller is used to setup communication between a processor and a SD card. It supports the translation protocol from a serial peripheral interface (SPI) bus to SD task.

Figure 1 shows the System Interface of this reference design.

***Figure 1. System Block Diagram for I2S Tx***

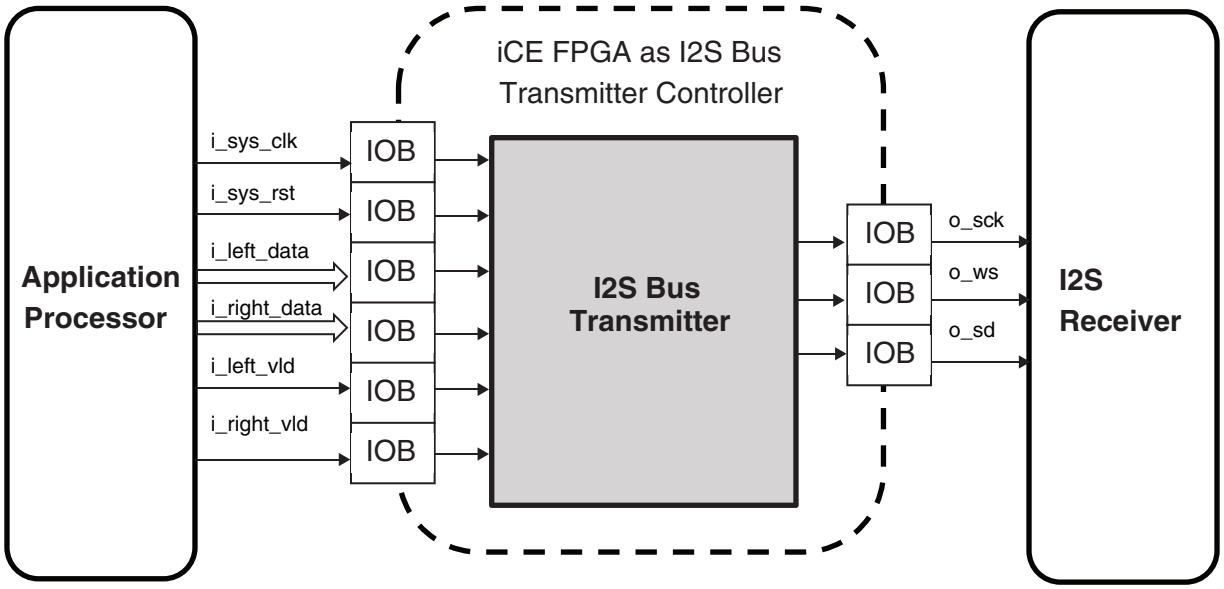
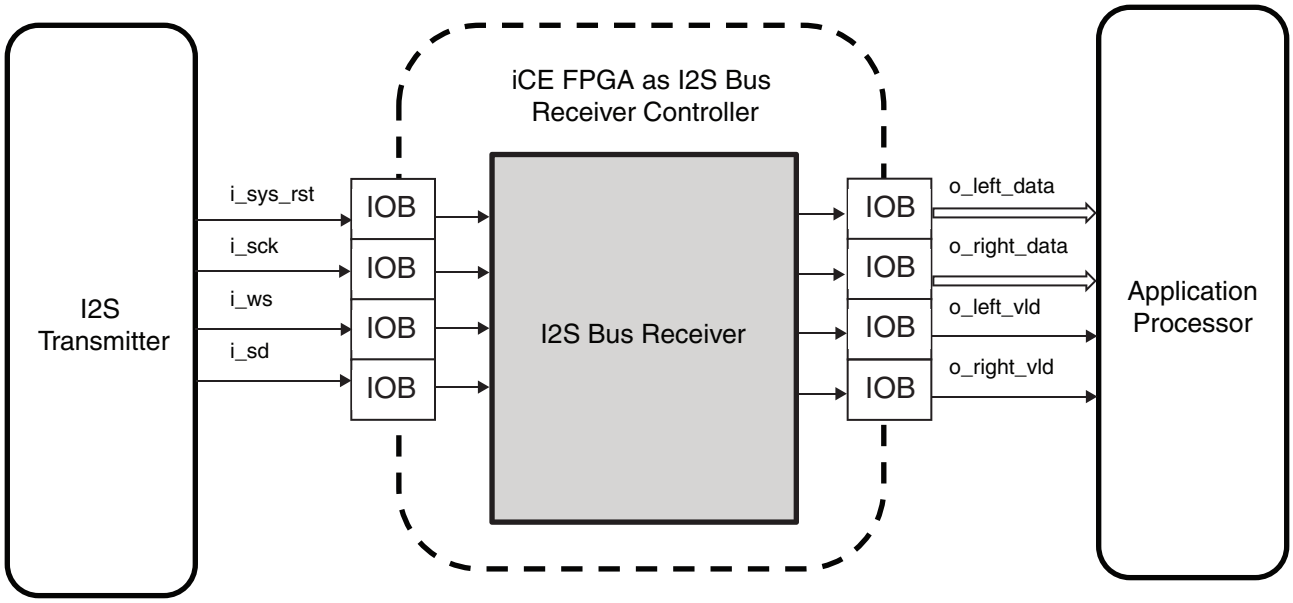


*Figure 2. System Block Diagram for I2S Rx*




## Signal Description

*Table 1. Signal Description for I2S Tx*

| Signal | Width | Type | Description |
|---|---|---|---|
| i_sck | 1 | Input | Input System Clock |
| i_sys_rst | 1 | Input | Input System Reset |
| i_left_data | [Data_Width-1:0] | Input | Input data bus-carries data to be sent via I2S bus-Left Channel Data |
| i_right_data | [Data_Width-1:0] | Input | Input data bus-carries data to be sent via I2S bus-Right Channel Data |
| i_left_vld | 1 | Input | A high indicates existence of left channel data on input bus |
| i_right_vld | 1 | Input | A high indicates existence of right channel data on input bus |
| o_sck | 1 | Output | Serial Clock Line |
| o_ws | 1 | Output | Write Select, a high indicates right channel, low indicates low channel |
| o_sd | 1 | Output | Serial Data Line |

# Functional Description

***Figure 3. Functional Block Diagram for I2S Tx***

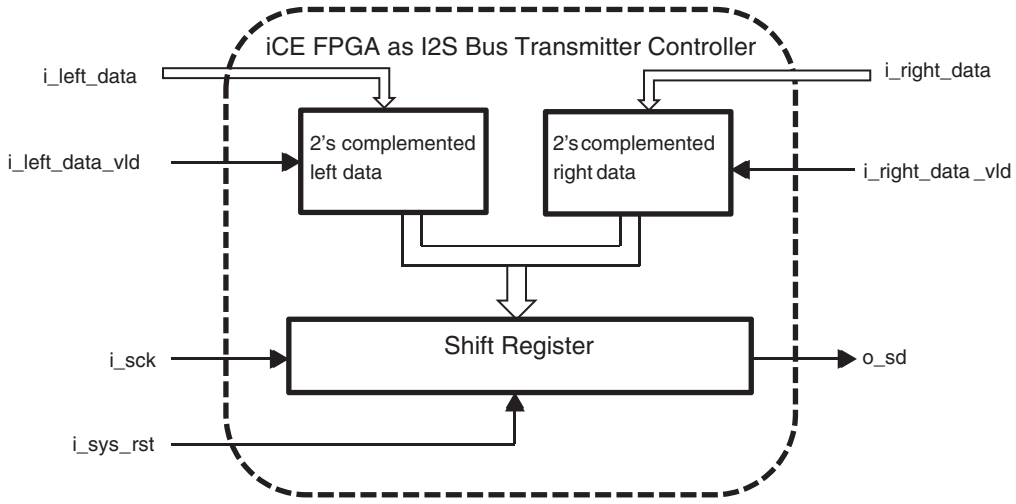
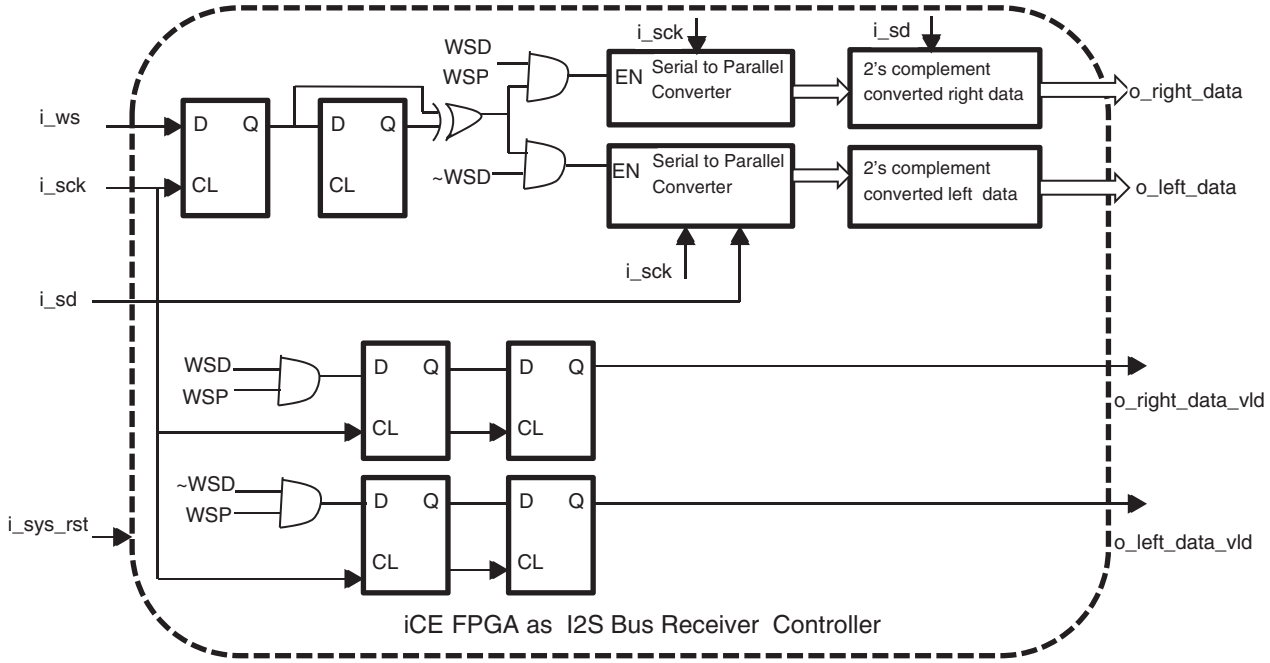



***Figure 4. Functional Block Diagram for I2S Rx***




## Basic Serial Bus Requirements

The bus has only to handle audio data, while the other signals, such as sub-coding and control, are transferred separately. To minimize the number of pins required and to keep wiring simple, a 3-line serial bus is used consisting of a line for two time-multiplexed data channels, a word select line and a clock line. Since the transmitter and receiver have the same clock signal for data transmission, the transmitter as the master, has to generate the bit clock, word-select signal and data. In complex systems however, there may be several transmitters and receivers, which makes it difficult to define the master. In such systems, there is usually a system master controlling digital

audio data-flow between the various ICs. Transmitters then, have to generate data under the control of an external clock, and so act as a slave.

## The I2S Bus

As shown in Figure 1, the bus has three lines as listed below and the device generating SCK and WS is the master.

- continuous serial clock (SCK)
- word select (WS)
- serial data (SD)

## Serial Data

Serial data is transmitted in two's complement with the MSB first. The MSB is transmitted first because the transmitter and receiver may have different word lengths. It isn't necessary for the transmitter to know how many bits the receiver can handle, nor does the receiver need to know how many bits are being transmitted.

When the system word length is greater than the transmitter word length, the word is truncated (least significant data bits are set to '0') for data transmission. If the receiver is sent more bits than its word length, the bits after the LSB are ignored. On the other hand, if the receiver is sent fewer bits than its word length, the missing bits are set to zero internally. And so, the MSB has a fixed position, whereas the position of the LSB depends on the word length. The transmitter always sends the MSB of the next word one clock period after the WS changes.

Serial data sent by the transmitter may be synchronized with either the trailing (HIGH-to-LOW) or the leading (LOW-to-HIGH) edge of the clock signal. However, the serial data must be latched into the receiver on the leading edge of the serial clock signal, and so there are some restrictions when transmitting data that is synchronized with the leading edge.

## Word Select

The word select line indicates the channel being transmitted:

- WS = 0; channel 1 (left)
- WS = 1; channel 2 (right)

WS may change either on a trailing or leading edge of the serial clock, but it doesn't need to be symmetrical. In the slave, this signal is latched on the leading edge of the clock signal. The WS line changes one clock period before the MSB is transmitted. This allows the slave transmitter to derive synchronous timing of the serial data that will be set up for transmission. Furthermore, it enables the receiver to store the previous word and clear the input for the next word.

## Timing

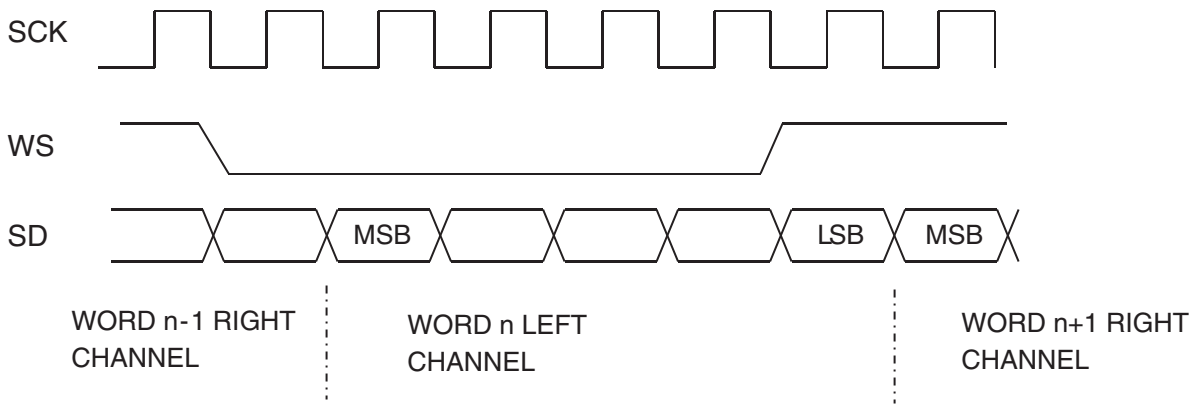
In the I2S format, any device can act as the system master by providing the necessary clock signals. A slave will usually derive its internal clock signal from an external clock input. This means, taking into account the propagation delays between master clock and the data and/or word-select signals, that the total delay is simply the sum of:

- the delay between the external (master) clock and the slave's internal clock; and
- the delay between the internal clock and the data and/or word-select signals.

For data and word-select inputs, the external to internal clock delay is of no consequence because it only lengthens the effective set-up Time. The major part of the time margin is to accommodate the difference between the propagation delay of the transmitter, and the time required to set up the receiver.

All timing requirements are specified relative to the clock period or to the minimum allowed clock period of a device. This means that higher data rates can be used in the future.

*Figure 5. Timing Diagram*




## Initialization Condition

No user specific initialization conditions, except that the i_sys_rst must be held low initially to bring-up the design in a correct operating state.

## Configurable Parameters

- I2S_Tx_Rx – This parameter configures the I2S Controller for Tx or Rx.
  - '1' configures Controller for I2S Transmitter and
  - '0' configures Controller for I2S Receiver
- DATA_WIDTH – This parameter configures the data width of the I2S transaction. By default, it is set to 16

## Operation Sequence

### I2S Transmitter

1. Set the parameter I2S_Tx_Rx to 1
2. Testbench has tasks to generate control signals like left valid and right valid signals
3. Testbench also has tasks to send left and right data to the controller.
4. Sent data and received data are compared to state SET_PASS

### I2S Receiver

1. Set the parameter I2S_Tx_Rx to 0
2. Testbench has tasks to set or initialize the memory
3. Testbench also has tasks for generating WS, Serial data
4. Written and read data are compared to state SET_PASS

## Simulation Waveforms

***Figure 6. Simulation Waveform for I2S Tx***

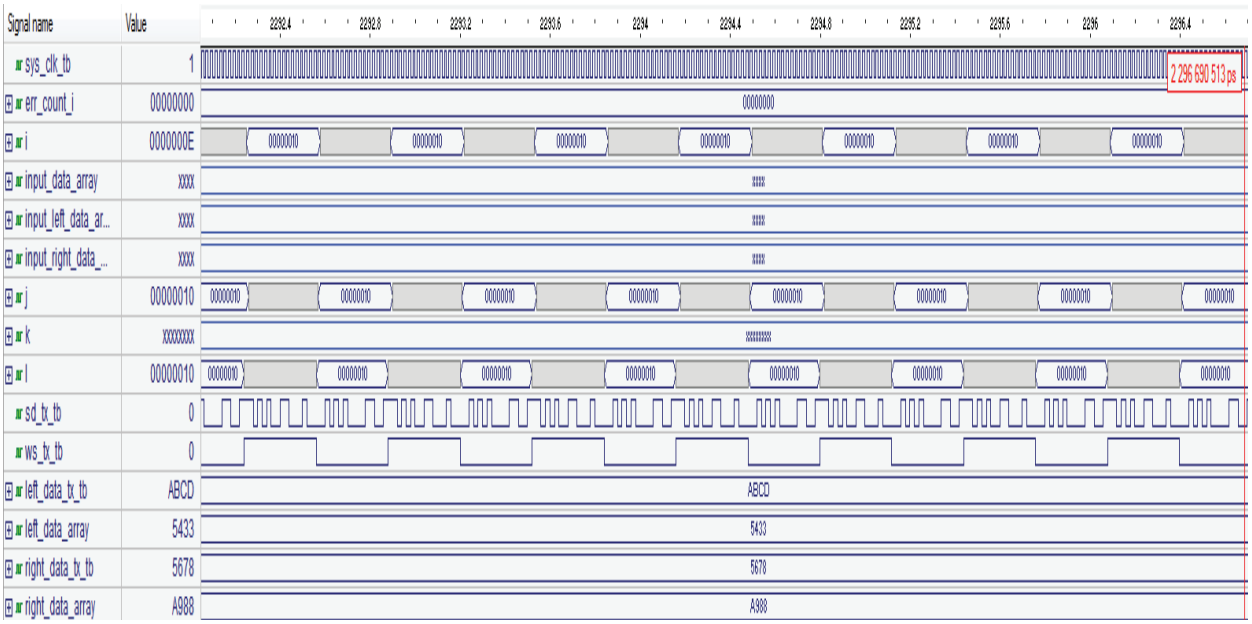
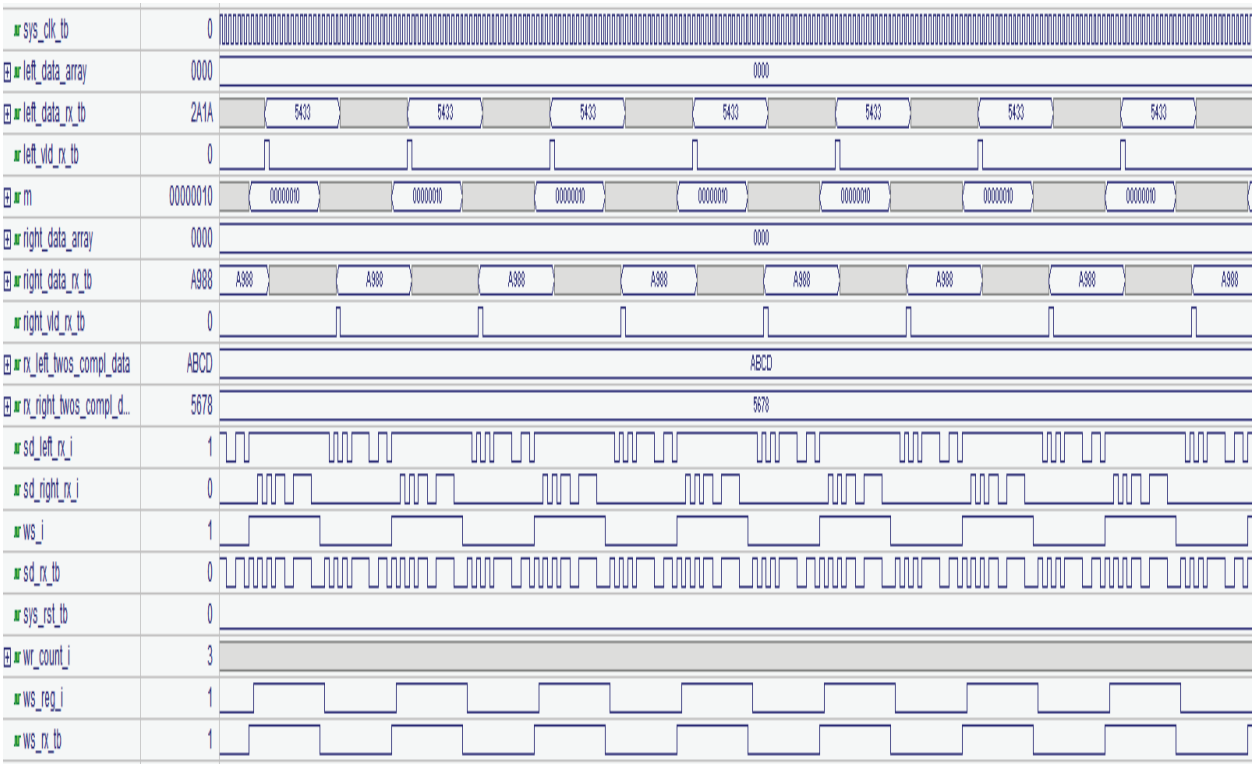



***Figure 7. Simulation Waveform for I2S Rx***



## Implementation

This design is implemented in Verilog. When using this design in a different device, density, speed or grade, performance and utilization may vary.

***Table 2. Performance and Resource Utilization***

| Family | Language | Utilization (LUTs) | $f_{MAX}$ (MHz) | I/Os | Architectural Resources |
|---|---|---|---|---|---|
| iCE40[1] | Verilog | 111 | >50 | 41 | (24/160)PLBs |

1. Performance and utilization characteristics are generated using iCE40LP1K-CM121 with iCEcube2 design software.

## References

- iCE40 Family Handbook

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

## Revision History

| Date | Version | Change Summary |
|---|---|---|
| April 2013 | 01.0 | Initial release. |