Single-wire Signal Aggregation

Reference Design

FPGA-RD-02039 Version 1.1

September 2018

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
ACK	Acknowledge bit sent from RX side to TX side to indicate the received data parity check is OK
GPIO	General Purpose Inputs and Outputs
PID	Payload ID
PLL	Phase Locked Loop
PT	Payload Type
RX	Receiver
TDM	Time Domain Multiplexing
TX	Transmitter
I^2S	Inter-IC Sound
MDP	Mobile Development Platform

1. Introduction

Single-wire is a single-ended single-wire connection between two FPGAs to provide a TDM-based bidirectional communication aggregating multiple data such as I²C, I²S, GPIO to add more flexibility on customer's system and board design. The design is targeted to iCE40™ UltraPlus and compiled by Radiant 1.0 to generate a bit file. Hardware behavior is verified aggregating multiple I²C and GPIOs on iCE40 UltraPlus Breakout Board. In addition, hardware behavior on I²S is verified on iCE40 Ultraplus MDP (Mobile Development Platform) Board. Consult technical support for the software patch if you encounter compilation issue on Radiant 1.0.

1.1. Features List

- Up to 7 channels can be aggregated in total
- Variable packet length for efficient use of the bandwidth
- Retransmit feature is offered when parity error is detected on the Receiver (RX) side. This feature is applicable for I²C and GPIO data
- Supports I²C fast-mode 400 kbps and fast-mode plus 1 Mbps
- Support up to two channels of I²S data
- Support I²S Controller option to generate SCK and WS for the I²S transmitter device
- I²C interrupt can be realized by GPIO with event-based transmission
- Raw data rate on Single-wire is ~7.5 Mbps or higher

1.2. Block Diagram

Figure 1.1 shows a block level diagram of Single-wire reference design (RD) in typical configuration with I²C, I²Sand GPIO aggregation, which has a Master FPGA and Slave FPGA. There are two essential design differences between the Master FPGA and Slave FPGA:

- Upon the power-up, Master FPGA generates low pulse on the Single-wire link and wait for another low pulse generated by Slave FPGA as an acknowledgement.
- In case I²S data was enable, I²S sck pulse will be sent by Master FPGA to the link for Slave FPGA clock training. This is after the link acknowledgement by the Slave FPGA.
- Master FPGA always wins in the very first transmitter (TX) request contention after power-up.

Regarding I²C interface, both I²C master bridge and I²C slave bridge can reside in both Master and Slave FPGAs. For I²S interface, both I²S master and slave can reside in both Master and Slave FPGA. Each I²S channel may a different Sample rate, Sample Depth and Buffer Depth per I²S channel. GPIO data width can be different between gpio_in and gpio_out, but gpio_in/gpio_out data width on Master FPGA must match the corresponding gpio_out/gpio_in data width on Slave FPGA. The channel relationships between two FPGAs must be properly maintained. Channel configuration is flexible.

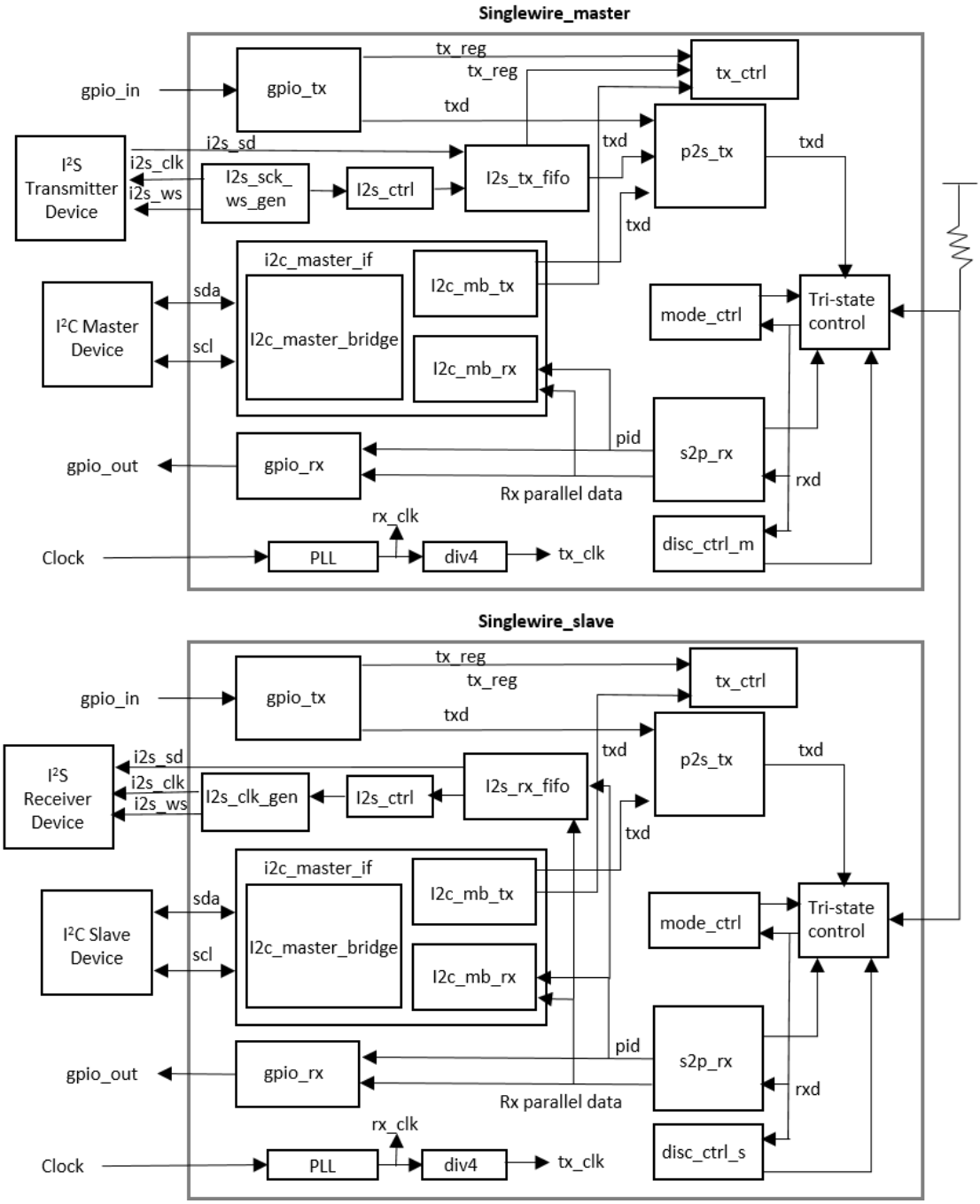


Figure 1.1. Single-wire Block Diagram Example

2. Parameters and Port List

Compiler directives and parameters are included in the top-level Verilog files of singlewire_master.v and singlewire_slave.v. You can modify these directives and parameters upon your own configurations.

2.1. Compiler Directives

Table 2.1 shows the compiler directives affect Single-wire RD. Compiler directives can be defined by creating a module with the corresponding define directives or by defining Verilog compiler directives in command line during compilation. Only one directive is set in each category. As shown in Table 2.1 and Table 2.2, some parameter selections are restricted by other parameter settings.

Table 2.1. Top-level Compiler Directives

Category	Compiler Directive	Remarks
I²C channel count[1]	NUM_OF_I2C_CH_0	Number of I²C channels aggregated. Only one of these 8 directives must be defined.
	NUM_OF_I2C_CH_1	
	NUM_OF_I2C_CH_2	
	NUM_OF_I2C_CH_3	
	NUM_OF_I2C_CH_4	
	NUM_OF_I2C_CH_5	
	NUM_OF_I2C_CH_6	
	NUM_OF_I2C_CH_7	
I²C Type[2]	I2C1_TYPE_MB	Connection of I²C channel #1. _MB for I²C Master and _SB for I²C Slave connection. _NA is used if that I²C channel #1 is not used. Only one of these 3 directives must be defined.
	I2C1_TYPE_SB	
	I2C1_TYPE_NA	
	I2C2_TYPE_MB	Connection of I²C channel #2. _MB for I²C Master and _SB for I²C Slave connection. _NA is used if that I²C channel #2 is not used. Only one of these 3 directives must be defined.
	I2C2_TYPE_SB	
	I2C2_TYPE_NA	
	I2C3_TYPE_MB	Connection of I²C channel #3. _MB for I²C Master and _SB for I²C Slave connection. _NA is used if that I²C channel #3 is not used. Only one of these 3 directives must be defined.
	I2C3_TYPE_SB	
	I2C3_TYPE_NA	
	I2C4_TYPE_MB	Connection of I²C channel #4. _MB for I²C Master and _SB for I²C Slave connection. _NA is used if that I²C channel #4 is not used. Only one of these 3 directives must be defined.
	I2C4_TYPE_SB	
	I2C4_TYPE_NA	
	I2C5_TYPE_MB	Connection of I²C channel #5. _MB for I²C Master and _SB for I²C Slave connection. _NA is used if that I²C channel #5 is not used. Only one of these 3 directives must be defined.
	I2C5_TYPE_SB	
	I2C5_TYPE_NA	
	I2C6_TYPE_MB	Connection of I²C channel #6. _MB for I²C Master and _SB for I²C Slave connection. _NA is used if that I²C channel #6 is not used. Only one of these 3 directives must be defined.
	I2C6_TYPE_SB	
	I2C6_TYPE_NA	
	I2C7_TYPE_MB	Connection of I²C channel #7. _MB for I²C Master and _SB for I²C Slave connection. _NA is used if that I²C channel #7 is not used. Only one of these 3 directives must be defined.
	I2C7_TYPE_SB	
	I2C7_TYPE_NA	
GPIO Tx channel count[1]	NUM_OF_GPIO_TX_CH_0	Number of GPIO TX channels aggregated. Only one of these 8 directives must be defined.
	NUM_OF_GPIO_TX_CH_1	
	NUM_OF_GPIO_TX_CH_2	
	NUM_OF_GPIO_TX_CH_3	

Category	Compiler Directive	Remarks
	NUM_OF_GPIO_TX_CH_4	
	NUM_OF_GPIO_TX_CH_5	
	NUM_OF_GPIO_TX_CH_6	
	NUM_OF_GPIO_TX_CH_7	
GPIO Rx channel count[1]	NUM_OF_GPIO_RX_CH_0	Number of GPIO TX channels aggregated. Only one of these 8 directives must be defined.
	NUM_OF_GPIO_RX_CH_1	
	NUM_OF_GPIO_RX_CH_2	
	NUM_OF_GPIO_RX_CH_3	
	NUM_OF_GPIO_RX_CH_4	
	NUM_OF_GPIO_RX_CH_5	
	NUM_OF_GPIO_RX_CH_6	
	NUM_OF_GPIO_TX_CH_7	
I²S channel count[1]	NUM_OF_I2S_CH_0	Number of I²S channels aggregated. Only one of these 3 directives must be defined.
	NUM_OF_I2S_CH_1	
	NUM_OF_I2S_CH_2	
I²S TX count	NUM_OF_I2S_TX_CH_0	Number of I²S channels that will act as TX for Transmitter connection. Only one of these 3 directives must be defined.
	NUM_OF_I2S_TX_CH_1	
	NUM_OF_I2S_TX_CH_2	
I²S RX count	NUM_OF_I2S_RX_CH_0	Number of I²S channels that will act as a RX for the Receiver Connection. Only one of these 3 directives must be defined.
	NUM_OF_I2S_RX_CH_1	
	NUM_OF_I2S_RX_CH_2	
I²S Clock count	NUM_OF_I2S_CLK_0	Number of I²S clocks to be used. Only one of these 3 directives must be defined
	NUM_OF_I2S_CLK_1	
	NUM_OF_I2S_CLK_2	
I²S RX CLK Types	I2S1_RX_MASTER	Directive to let the FPGA outputs I2S CLK/WS on the RX side.
CLK_SEND	I2S1_CLK_SEND	This enables the FPGA to send I²S CLK to the other FPGA for training or clock learning
Enable I²S Controller	I2S1_CONTROLLER	Master device will generate the SCK and WS for the Transmitter Device. Do not define if the Transmitter device will provide its SCK and WS

Notes:

1. Maximum total number of channels is 7. The sum of channel numbers specified by NUM_OF_I2C_CH_*; NUM_OF_I2S_CH_* and max (NUM_OF_GPIO_TX_CH_*, NUM_OF_GPIO_RX_CH_*) must not exceed 7.
2. I²C Type of unused I²C channel must be set to TYPE_NA.

2.2. Parameters

Table 2.2 shows the global parameters of Single-wire RD. Some parameters are applicable only to associated I/Os.

Table 2.2. Top-level Parameters

Single-wire RD Parameters	Values	Remarks
FPGA_ID	0 or 1	Set to 0 for Master FPGA. Set to 1 for Slave FPGA.
NUM_OF_CH	1 to 7	Total number of channels aggregated
MAX_PAYLOAD_LENGTH	1 to 32	Maximum Payload data length. If I^2S channel is set, set the value to the highest I^2S Sample Depth assigned. If I^2S channel is not set, the value must be 9 if I^2C is used and GPIO data width is 9 or less. Otherwise set the GPIO data width.
MAX_TX_LENGTH	4 to 35	Maximum of TX data length. If I^2S channel is set, set the value of the highest I^2S sample depth assigned, If I^2S channel is not set, the value must be 14 if I^2C is used and GPIO data width is 11 or less. Otherwise set the GPIO data width + 3.
CH1_TYPE [6*8:1]	GPTXRX, GPIOTX, GPIORX, I2C_MB, or I2C_SB	CH #1 data type
CH2_TYPE [6*8:1]	GPTXRX, GPIOTX, GPIORX, I2C_MB, or I2C_SB, or UNUSED	CH #2 data type
CH3_TYPE [6*8:1]	GPTXRX, GPIOTX, GPIORX, I2C_MB, or I2C_SB, or UNUSED	CH #3 data type
CH4_TYPE [6*8:1]	GPTXRX, GPIOTX, GPIORX, I2C_MB, or I2C_SB, or UNUSED	CH #4 data type
CH5_TYPE [6*8:1]	GPTXRX, GPIOTX, GPIORX, I2C_MB, or I2C_SB, or UNUSED	CH #5 data type
CH6_TYPE [6*8:1]	GPTXRX, GPIOTX, GPIORX, I2C_MB, or I2C_SB, or UNUSED	CH #6 data type
CH7_TYPE [6*8:1]	GPTXRX, GPIOTX, GPIORX, I2C_MB, or I2C_SB, or UNUSED	CH #7 data type
I2C1_FREQ_K [9:0]	400 to 1000	I^2C #1 clock rate in kHz
I2C2_FREQ_K [9:0]	400 to 1000	I^2C #2 clock rate in kHz
I2C3_FREQ_K [9:0]	400 to 1000	I^2C #3 clock rate in kHz
I2C4_FREQ_K [9:0]	400 to 1000	I^2C #4 clock rate in kHz
I2C5_FREQ_K [9:0]	400 to 1000	I^2C #5 clock rate in kHz
I2C6_FREQ_K [9:0]	400 to 1000	I^2C #6 clock rate in kHz
I2C7_FREQ_K [9:0]	400 to 1000	I^2C #7 clock rate in kHz
GPIO1_TX_TYPE [6*8:1]	SAMPLE or EVENTS	GPIO #1 Tx trigger type
GPIO2_TX_TYPE [6*8:1]	SAMPLE or EVENTS	GPIO #2 Tx trigger type
GPIO3_TX_TYPE [6*8:1]	SAMPLE or EVENTS	GPIO #3 Tx trigger type
GPIO4_TX_TYPE [6*8:1]	SAMPLE or EVENTS	GPIO #4 Tx trigger type
GPIO5_TX_TYPE [6*8:1]	SAMPLE or EVENTS	GPIO #5 Tx trigger type
GPIO6_TX_TYPE [6*8:1]	SAMPLE or EVENTS	GPIO #6 Tx trigger type
GPIO7_TX_TYPE [6*8:1]	SAMPLE or EVENTS	GPIO #7 Tx trigger type
GPIO1_TX_RATE_K [10:0]	1 to 2000	GPIO #1 Tx data rate in kbps
GPIO2_TX_RATE_K [10:0]	1 to 2000	GPIO #2 Tx data rate in kbps
GPIO3_TX_RATE_K [10:0]	1 to 2000	GPIO #3 Tx data rate in kbps
GPIO4_TX_RATE_K [10:0]	1 to 2000	GPIO #4 Tx data rate in kbps
GPIO5_TX_RATE_K [10:0]	1 to 2000	GPIO #5 Tx data rate in kbps

Single-wire RD Parameters	Values	Remarks
GPIO6_TX_RATE_K [10:0]	1 to 2000	GPIO #6 Tx data rate in kbps
GPIO7_TX_RATE_K [10:0]	1 to 2000	GPIO #7 Tx data rate in kbps
GPIO1_TX_WIDTH [3:0][1]	1 to 15	GPIO #1 Tx data width
GPIO2_TX_WIDTH [3:0][1]	1 to 15	GPIO #2 Tx data width
GPIO3_TX_WIDTH [3:0][1]	1 to 15	GPIO #3 Tx data width
GPIO4_TX_WIDTH [3:0][1]	1 to 15	GPIO #4 Tx data width
GPIO5_TX_WIDTH [3:0][1]	1 to 15	GPIO #5 Tx data width
GPIO6_TX_WIDTH [3:0][1]	1 to 15	GPIO #6 Tx data width
GPIO7_TX_WIDTH [3:0][1]	1 to 15	GPIO #7 Tx data width
GPIO1_RX_WIDTH [3:0][1]	1 to 15	GPIO #1 Rx data width
GPIO2_RX_WIDTH [3:0][1]	1 to 15	GPIO #2 Rx data width
GPIO3_RX_WIDTH [3:0][1]	1 to 15	GPIO #3 Rx data width
GPIO4_RX_WIDTH [3:0][1]	1 to 15	GPIO #4 Rx data width
GPIO5_RX_WIDTH [3:0][1]	1 to 15	GPIO #5 Rx data width
GPIO6_RX_WIDTH [3:0][1]	1 to 15	GPIO #6 Rx data width
GPIO7_RX_WIDTH [3:0][1]	1 to 15	GPIO #7 Rx data width
GPIO_TX_MAX_WIDTH [3:0]	1 to 15, shows the maximum bit width among all GPIO Tx channels	The value of Master FPGA has to be matched GPIO_RX_MAX_WIDTH of Slave FPGA, vice versa.
GPIO_RX_MAX_WIDTH [3:0]	1 to 15, shows the maximum bit width among all GPIO Rx channels	The value of Master FPGA has to be matched GPIO_TX_MAX_WIDTH of Slave FPGA, vice versa.
I2S1_SAMPLE_DEPTH	1 to 32	I²S Channel 1 Bit Depth per word line
I2S2_SAMPLE_DEPTH	1 to 32	I²S Channel 2 Bit Depth per word line
I2S1_BUFFER_DEPTH	Default = 6	I²S Channel 1 FIFO Buffer Depth
I2S2_BUFFER_DEPTH	Default = 6	I²S Channel 1 FIFO Buffer Depth
I2S1_SAMPLE_RATE_K	Default: 48	I²S Channel 1 Sample rate in KHz
I2S2_SAMPLE_RATE_K	Default: 48	I²S Channel 2 Sample rate in KHz

Note: GPIO*_TX_WIDTH in Maser FPGA has to match GPIO*_RX_WIDTH in Slave FPGA. GPIO*_RX_WIDTH in Master FPGA has to match GPIO*_TX_WIDTH in Slave FPGA when those GPIOs are used.

2.3. Configuration Examples

Two examples of compiler directives and parameter settings are shown below.

Example 1: MPU communicates with two I²C Slave Devices using two interrupts.

Table 2.3. Configuration Example 1

Master FPGA	Slave FPGA
Compiler Directives	
ICE_UP [1]	ICE_UP [1]
NUM_OF_I2C_CH_2 [2]	NUM_OF_I2C_CH_2 [2]
I2C1_TYPE_MB [2]	I2C1_TYPE_SB [2]
I2C2_TYPE_MB [2]	I2C2_TYPE_SB [2]
I2C3_TYPE_NA	I2C3_TYPE_NA
I2C4_TYPE_NA	I2C4_TYPE_NA
I2C5_TYPE_NA	I2C5_TYPE_NA
I2C6_TYPE_NA	I2C6_TYPE_NA
I2C7_TYPE_NA	I2C7_TYPE_NA
NUM_OF_GPIO_TX_CH_0	NUM_OF_GPIO_TX_CH_1 [2]
NUM_OF_GPIO_RX_CH_1 [2]	NUM_OF_GPIO_RX_CH_0
NUM_OF I2S_CH_0	NUM_OF_I2S_CH_0
NUM_OF_I2S_CLK_0	NUM_OF_I2S_CLK_0
NUM_OF_I2S_TX_CH_0	NUM_OF_I2S_TX_CH_0
NUM_OF_I2S_RX_CH_0	NUM_OF_I2S_RX_CH_0
Parameters	
FPGA_ID = 0 [2]	FPGA_ID = 1 [2]
NUM_OF_CH = 3[2]	NUM_OF_CH = 3 [2]
TX_LOOP_CNT_M1 = 0[1]	TX_LOOP_CNT_M1 = 0 [1]
MAX_PAYLOAD_LENGTH = 9[2]	MAX_PAYLOAD_LENGTH = 9 [2]
MAX_TX_LENGTH = 14[2]	MAX_TX_LENGTH = 14 [2]
CH1_TYPE = I2C_MB[2]	CH1_TYPE = I2C_SB[2]
CH2_TYPE = I2C_MB [2]	CH2_TYPE = I2C_SB[2]
CH3_TYPE = GPIORX[2]	CH3_TYPE = GPIOTX[2]
CH4_TYPE = UNUSED	CH4_TYPE = UNUSED
CH5_TYPE = UNUSED	CH5_TYPE = UNUSED
CH6_TYPE = UNUSED	CH6_TYPE = UNUSED
CH7_TYPE = UNUSED	CH7_TYPE = UNUSED
I2C1_FREQ_K = 10'd400	I2C1_FREQ_K = 10'd400[2]
I2C2_FREQ_K = 10'd400	I2C2_FREQ_K = 10'd400[2]
I2C3_FREQ_K = 10'd400	I2C3_FREQ_K = 10'd400
I2C4_FREQ_K = 10'd400	I2C4_FREQ_K = 10'd400
I2C5_FREQ_K = 10'd400	I2C5_FREQ_K = 10'd400
I2C6_FREQ_K = 10'd400	I2C6_FREQ_K = 10'd400
I2C7_FREQ_K = 10'd400	I2C7_FREQ_K = 10'd400
GPIO1_TX_TYPE = EVENTS	GPIO1_TX_TYPE = EVENTS[2]
GPIO2_TX_TYPE = EVENTS	GPIO2_TX_TYPE = EVENTS
GPIO3_TX_TYPE = EVENTS	GPIO3_TX_TYPE = EVENTS
GPIO4_TX_TYPE = EVENTS	GPIO4_TX_TYPE = EVENTS
GPIO5_TX_TYPE = EVENTS	GPIO5_TX_TYPE = EVENTS
GPIO6_TX_TYPE = EVENTS	GPIO6_TX_TYPE = EVENTS
GPIO7_TX_TYPE = EVENTS	GPIO7_TX_TYPE = EVENTS

Master FPGA	Slave FPGA
GPIO1_TX_RATE_K = 11'd1	GPIO1_TX_RATE_K = 11'd1
GPIO2_TX_RATE_K = 11'd1	GPIO2_TX_RATE_K = 11'd1
GPIO3_TX_RATE_K = 11'd1	GPIO3_TX_RATE_K = 11'd1
GPIO4_TX_RATE_K = 11'd1	GPIO4_TX_RATE_K = 11'd1
GPIO5_TX_RATE_K = 11'd1	GPIO5_TX_RATE_K = 11'd1
GPIO6_TX_RATE_K = 11'd1	GPIO6_TX_RATE_K = 11'd1
GPIO7_TX_RATE_K = 11'd1	GPIO7_TX_RATE_K = 11'd1
GPIO1_TX_WIDTH = 4'd1	GPIO1_TX_WIDTH = 4'd2[2]
GPIO2_TX_WIDTH = 4'd1	GPIO2_TX_WIDTH = 4'd1
GPIO3_TX_WIDTH = 4'd1	GPIO3_TX_WIDTH = 4'd1
GPIO4_TX_WIDTH = 4'd1	GPIO4_TX_WIDTH = 4'd1
GPIO5_TX_WIDTH = 4'd1	GPIO5_TX_WIDTH = 4'd1
GPIO6_TX_WIDTH = 4'd1	GPIO6_TX_WIDTH = 4'd1
GPIO7_TX_WIDTH = 4'd1	GPIO7_TX_WIDTH = 4'd1
GPIO1_RX_WIDTH = 4'd2[2]	GPIO1_RX_WIDTH = 4'd1
GPIO2_RX_WIDTH = 4'd1	GPIO2_RX_WIDTH = 4'd1
GPIO3_RX_WIDTH = 4'd1	GPIO3_RX_WIDTH = 4'd1
GPIO4_RX_WIDTH = 4'd1	GPIO4_RX_WIDTH = 4'd1
GPIO5_RX_WIDTH = 4'd1	GPIO5_RX_WIDTH = 4'd1
GPIO6_RX_WIDTH = 4'd1	GPIO6_RX_WIDTH = 4'd1
GPIO7_RX_WIDTH = 4'd1	GPIO7_RX_WIDTH = 4'd1
GPIO_TX_MAX_WIDTH = 4'd1	GPIO_TX_MAX_WIDTH = 4'd2[2]
GPIO_RX_MAX_WIDTH = 4'd2[2]	GPIO_RX_MAX_WIDTH = 4'd1
I2S1_BIDIR = 0	I2S1_BIDIR =0
I2S1_SAMPLE_DEPTH = 0	I2S1_SAMPLE_DEPTH = 0
I2S1_BUFFER_DEPTH = 0	I2S1_BUFFER_DEPTH = 0
I2S1_SAMPLE_RATE_K = 0	I2S1_SAMPLE_RATE_K = 0
I2S2_SAMPLE_DEPTH = 0	I2S2_SAMPLE_DEPTH = 0
I2S2_BUFFER_DEPTH = 0	I2S2_BUFFER_DEPTH = 0
I2S2_SAMPLE_RATE_K = 0	I2S2_SAMPLE_RATE_K = 0

Notes:

1. Directives and parameters that should not be changed by users.
2. Active Directives/parameters active in this configuration.

Example 2: Add 3rd I2C (opposite Master/Slave) and UART GPIO (4 bits/3 bits) against Example 1.

Table 2.4. Configuration Example 2

Master FPGA	Slave FPGA
Compiler Directives	
ICE_UP[1]	ICE_UP[1]
NUM_OF_I2C_CH_3[2]	NUM_OF_I2C_CH_3[2]
I2C1_TYPE_MB[2]	I2C1_TYPE_SB[2]
I2C2_TYPE_MB[2]	I2C2_TYPE_SB[2]
I2C3_TYPE_SB[2]	I2C3_TYPE_MB[2]
I2C4_TYPE_NA	I2C4_TYPE_NA
I2C5_TYPE_NA	I2C5_TYPE_NA
I2C6_TYPE_NA	I2C6_TYPE_NA
I2C7_TYPE_NA	I2C7_TYPE_NA
NUM_OF_GPIO_TX_CH_1[2]	NUM_OF_GPIO_TX_CH_2[2]
NUM_OF_GPIO_RX_CH_2[2]	NUM_OF_GPIO_RX_CH_1[2]
NUM_OF I2S_CH_0	NUM_OF_I2S_CH_0
NUM_OF_I2S_CLK_0	NUM_OF_I2S_CLK_0
NUM_OF_I2S_TX_CH_0	NUM_OF_I2S_TX_CH_0
NUM_OF_I2S_RX_CH_0	NUM_OF_I2S_RX_CH_0
Parameters	
FPGA_ID = 0[2]	FPGA_ID = 1[2]
NUM_OF_CH = 5[2]	NUM_OF_CH = 5[2]
TX_LOOP_CNT_M1 = 0[1]	TX_LOOP_CNT_M1 = 0[1]
MAX_PAYLOAD_LENGTH = 9[2]	MAX_PAYLOAD_LENGTH = 9[2]
MAX_TX_LENGTH = 14[2]	MAX_TX_LENGTH = 14[2]
CH1_TYPE = I2C_MB[2]	CH1_TYPE = I2C_SB[2]
CH2_TYPE = I2C_MB[2]	CH2_TYPE = I2C_SB[2]
CH3_TYPE = I2C_SB[2]	CH3_TYPE = I2C_MB[2]
CH4_TYPE = GPTXRX	CH4_TYPE = GPTXRX
CH5_TYPE = GPIORX	CH5_TYPE = GPIOTX
CH6_TYPE = UNUSED	CH6_TYPE = UNUSED
CH7_TYPE = UNUSED	CH7_TYPE = UNUSED
I2C1_FREQ_K = 10'd400	I2C1_FREQ_K = 10'd400[2]
I2C2_FREQ_K = 10'd400	I2C2_FREQ_K = 10'd400[2]
I2C3_FREQ_K = 10'd400[2]	I2C3_FREQ_K = 10'd400
I2C4_FREQ_K = 10'd400	I2C4_FREQ_K = 10'd400
I2C5_FREQ_K = 10'd400	I2C5_FREQ_K = 10'd400
I2C6_FREQ_K = 10'd400	I2C6_FREQ_K = 10'd400
I2C7_FREQ_K = 10'd400	I2C7_FREQ_K = 10'd400
GPIO1_TX_TYPE = SAMPLE[2]	GPIO1_TX_TYPE = SAMPLE[2]
GPIO2_TX_TYPE = EVENTS	GPIO2_TX_TYPE = EVENTS[2]
GPIO3_TX_TYPE = EVENTS	GPIO3_TX_TYPE = EVENTS
GPIO4_TX_TYPE = EVENTS	GPIO4_TX_TYPE = EVENTS
GPIO5_TX_TYPE = EVENTS	GPIO5_TX_TYPE = EVENTS
GPIO6_TX_TYPE = EVENTS	GPIO6_TX_TYPE = EVENTS
GPIO7_TX_TYPE = EVENTS	GPIO7_TX_TYPE = EVENTS
GPIO1_TX_RATE_K = 11'd1[2]	GPIO1_TX_RATE_K = 11'd1[2]
GPIO2_TX_RATE_K = 11'd1	GPIO2_TX_RATE_K = 11'd1
GPIO3_TX_RATE_K = 11'd1	GPIO3_TX_RATE_K = 11'd1

Master FPGA	Slave FPGA
GPIO4_TX_RATE_K = 11'd1	GPIO4_TX_RATE_K = 11'd1
GPIO5_TX_RATE_K = 11'd1	GPIO5_TX_RATE_K = 11'd1
GPIO6_TX_RATE_K = 11'd1	GPIO6_TX_RATE_K = 11'd1
GPIO7_TX_RATE_K = 11'd1	GPIO7_TX_RATE_K = 11'd1
GPIO1_TX_WIDTH = 4'd4[2]	GPIO1_TX_WIDTH = 4'd3[2]
GPIO2_TX_WIDTH = 4'd1	GPIO2_TX_WIDTH = 4'd2[2]
GPIO3_TX_WIDTH = 4'd1	GPIO3_TX_WIDTH = 4'd1
GPIO4_TX_WIDTH = 4'd1	GPIO4_TX_WIDTH = 4'd1
GPIO5_TX_WIDTH = 4'd1	GPIO5_TX_WIDTH = 4'd1
GPIO6_TX_WIDTH = 4'd1	GPIO6_TX_WIDTH = 4'd1
GPIO7_TX_WIDTH = 4'd1	GPIO7_TX_WIDTH = 4'd1
GPIO1_RX_WIDTH = 4'd3 [2]	GPIO1_RX_WIDTH = 4'd4[2]
GPIO2_RX_WIDTH = 4'd2 [2]	GPIO2_RX_WIDTH = 4'd1
GPIO3_RX_WIDTH = 4'd1	GPIO3_RX_WIDTH = 4'd1
GPIO4_RX_WIDTH = 4'd1	GPIO4_RX_WIDTH = 4'd1
GPIO5_RX_WIDTH = 4'd1	GPIO5_RX_WIDTH = 4'd1
GPIO6_RX_WIDTH = 4'd1	GPIO6_RX_WIDTH = 4'd1
GPIO7_RX_WIDTH = 4'd1	GPIO7_RX_WIDTH = 4'd1
GPIO_TX_MAX_WIDTH = 4'd4[2]	GPIO_TX_MAX_WIDTH = 4'd3[2]
GPIO_RX_MAX_WIDTH = 4'd3[2]	GPIO_RX_MAX_WIDTH = 4'd4[2]
I2S1_BIDIR = 0	I2S1_BIDIR =0
I2S1_SAMPLE_DEPTH = 0	I2S1_SAMPLE_DEPTH = 0
I2S1_BUFFER_DEPTH = 0	I2S1_BUFFER_DEPTH = 0
I2S1_SAMPLE_RATE_K = 0	I2S1_SAMPLE_RATE_K = 0
I2S2_SAMPLE_DEPTH = 0	I2S2_SAMPLE_DEPTH = 0
I2S2_BUFFER_DEPTH = 0	I2S2_BUFFER_DEPTH = 0
I2S2_SAMPLE_RATE_K = 0	I2S2_SAMPLE_RATE_K = 0

Notes:

1. Directives and parameters that should not be changed by users.
2. Directives and parameters that are active in this configuration.

Regarding channel assignments, I²C channels always have to be assigned to lower channels against GPIO channels.

Example 3: 1 I²S channel from Master to Slave, 2 I²C and UART GPIO (4 bits/3 bits).

Table 2.5. Configuration Example 3

Master FPGA	Slave FPGA
Compiler Directives	
ICE_UP[1]	ICE_UP[1]
NUM_OF_I2C_CH_3[2]	NUM_OF_I2C_CH_3[2]
I2C1_TYPE_MB[2]	I2C1_TYPE_SB[2]
I2C2_TYPE_MB[2]	I2C2_TYPE_SB[2]
I2C3_TYPE_SB[2]	I2C3_TYPE_MB[2]
I2C4_TYPE_NA	I2C4_TYPE_NA
I2C5_TYPE_NA	I2C5_TYPE_NA
I2C6_TYPE_NA	I2C6_TYPE_NA
I2C7_TYPE_NA	I2C7_TYPE_NA
NUM_OF_GPIO_TX_CH_1[2]	NUM_OF_GPIO_TX_CH_2[2]
NUM_OF_GPIO_RX_CH_2[2]	NUM_OF_GPIO_RX_CH_1[2]
NUM_OF I2S_CH_1	NUM_OF_I2S_CH_1
NUM_OF_I2S_CLK_1	NUM_OF_I2S_CLK_0
NUM_OF_I2S_TX_CH_1	NUM_OF_I2S_TX_CH_0
NUM_OF_I2S_RX_CH_0	NUM_OF_I2S_RX_CH_1
I2S1_CLK_SEND	I2S1_RX_MASTER
I2S1_CONTROLLER	—
Parameters	
FPGA_ID = 0[2]	FPGA_ID = 1[2]
NUM_OF_CH = 5[2]	NUM_OF_CH = 5[2]
TX_LOOP_CNT_M1 = 0[1]	TX_LOOP_CNT_M1 = 0[1]
MAX_PAYLOAD_LENGTH = 32[2]	MAX_PAYLOAD_LENGTH = 32[2]
MAX_TX_LENGTH = 35[2]	MAX_TX_LENGTH = 35[2]
CH1_TYPE = I2C_MB[2]	CH1_TYPE = I2C_SB[2]
CH2_TYPE = I2C_MB[2]	CH2_TYPE = I2C_SB[2]
CH3_TYPE = I2C_SB[2]	CH3_TYPE = I2C_MB[2]
CH4_TYPE = GPTXRX	CH4_TYPE = GPTXRX
CH5_TYPE = GPIORX	CH5_TYPE = GPIOTX
CH6_TYPE = UNUSED	CH6_TYPE = UNUSED
CH7_TYPE = UNUSED	CH7_TYPE = UNUSED
I2C1_FREQ_K = 10'd400	I2C1_FREQ_K = 10'd400[2]
I2C2_FREQ_K = 10'd400	I2C2_FREQ_K = 10'd400[2]
I2C3_FREQ_K = 10'd400[2]	I2C3_FREQ_K = 10'd400
I2C4_FREQ_K = 10'd400	I2C4_FREQ_K = 10'd400
I2C5_FREQ_K = 10'd400	I2C5_FREQ_K = 10'd400
I2C6_FREQ_K = 10'd400	I2C6_FREQ_K = 10'd400
I2C7_FREQ_K = 10'd400	I2C7_FREQ_K = 10'd400
GPIO1_TX_TYPE = SAMPLE[2]	GPIO1_TX_TYPE = SAMPLE[2]
GPIO2_TX_TYPE = EVENTS	GPIO2_TX_TYPE = EVENTS[2]
GPIO3_TX_TYPE = EVENTS	GPIO3_TX_TYPE = EVENTS
GPIO4_TX_TYPE = EVENTS	GPIO4_TX_TYPE = EVENTS
GPIO5_TX_TYPE = EVENTS	GPIO5_TX_TYPE = EVENTS
GPIO6_TX_TYPE = EVENTS	GPIO6_TX_TYPE = EVENTS
GPIO7_TX_TYPE = EVENTS	GPIO7_TX_TYPE = EVENTS
GPIO1_TX_RATE_K = 11'd1[2]	GPIO1_TX_RATE_K = 11'd1[2]

Master FPGA	Slave FPGA
GPIO2_TX_RATE_K = 11'd1	GPIO2_TX_RATE_K = 11'd1
GPIO3_TX_RATE_K = 11'd1	GPIO3_TX_RATE_K = 11'd1
GPIO4_TX_RATE_K = 11'd1	GPIO4_TX_RATE_K = 11'd1
GPIO5_TX_RATE_K = 11'd1	GPIO5_TX_RATE_K = 11'd1
GPIO6_TX_RATE_K = 11'd1	GPIO6_TX_RATE_K = 11'd1
GPIO7_TX_RATE_K = 11'd1	GPIO7_TX_RATE_K = 11'd1
GPIO1_TX_WIDTH = 4'd4[2]	GPIO1_TX_WIDTH = 4'd3 [2]
GPIO2_TX_WIDTH = 4'd1	GPIO2_TX_WIDTH = 4'd2 [2]
GPIO3_TX_WIDTH = 4'd1	GPIO3_TX_WIDTH = 4'd1
GPIO4_TX_WIDTH = 4'd1	GPIO4_TX_WIDTH = 4'd1
GPIO5_TX_WIDTH = 4'd1	GPIO5_TX_WIDTH = 4'd1
GPIO6_TX_WIDTH = 4'd1	GPIO6_TX_WIDTH = 4'd1
GPIO7_TX_WIDTH = 4'd1	GPIO7_TX_WIDTH = 4'd1
GPIO1_RX_WIDTH = 4'd3 [2]	GPIO1_RX_WIDTH = 4'd4 [2]
GPIO2_RX_WIDTH = 4'd2 [2]	GPIO2_RX_WIDTH = 4'd1
GPIO3_RX_WIDTH = 4'd1	GPIO3_RX_WIDTH = 4'd1
GPIO4_RX_WIDTH = 4'd1	GPIO4_RX_WIDTH = 4'd1
GPIO5_RX_WIDTH = 4'd1	GPIO5_RX_WIDTH = 4'd1
GPIO6_RX_WIDTH = 4'd1	GPIO6_RX_WIDTH = 4'd1
GPIO7_RX_WIDTH = 4'd1	GPIO7_RX_WIDTH = 4'd1
GPIO_TX_MAX_WIDTH = 4'd4[2]	GPIO_TX_MAX_WIDTH = 4'd3[2]
GPIO_RX_MAX_WIDTH = 4'd3[2]	GPIO_RX_MAX_WIDTH = 4'd4[2]
I2S1_BIDIR = 0	I2S1_BIDIR =0
I2S1_SAMPLE_DEPTH = 32	I2S1_SAMPLE_DEPTH = 32
I2S1_BUFFER_DEPTH = 6	I2S1_BUFFER_DEPTH = 6
I2S1_SAMPLE_RATE_K = 48	I2S1_SAMPLE_RATE_K = 48
I2S2_SAMPLE_DEPTH = 0	I2S2_SAMPLE_DEPTH = 0
I2S2_BUFFER_DEPTH = 0	I2S2_BUFFER_DEPTH = 0
I2S2_SAMPLE_RATE_K = 0	I2S2_SAMPLE_RATE_K = 0

Notes:

1. Directives and parameters that should not be changed by users.
2. Directives and parameters that are active in this configuration.

Regarding channel assignments, I^2S channels always have to be assigned to lower channels followed by I^2C channels and then GPIO channels.

2.4. Top-Level I/Os

Table 2.6 shows the top-level I/Os of Single-wire RD with typical I²C, I²S and GPIO configuration. Actual I/Os depend on customer's channel configuration. All necessary I/O ports are automatically declared by compiler directives and parameter settings mentioned above.

Table 2.6. Single-wire Top-Level I/O

Port Name	Default	Direction	Clock Domain	Description
Reset				
rst_n	1'b1	Input	Async	Asynchronous system reset, active low
Status				
status[1:0]	2'b00	Output	N/A	Indicate the link connection status. See the Link Status section.
Link				
link	Hi-Z	Inout	tx_clk	Single-wire link between FPGAs Requires strong pullup
I²C Interface				
scl[#-1:0]	Hi-Z	Inout	Async	I²C Clock
sda[#-1:0]	Hi-Z	Inout	Async	I²C Data
I²S Interface				
I2s[#-1:0]_tx/rx_ws	Hi-Z	Input or Output	Async	I²S Word Select
I2s[#-1:0]_tx/rx_sd	Hi-Z	Input or Output	Async	I²S Data
I2s[#-1:0]_tx/rx_sclk	Hi-Z	Input or Output	Async	I²S Clock
GPIO Interface				
gpio#_in[GPIO_TX_WIDTH[#-1]-1:0]	—	Input	Async	GPIO # input
gpio#_out[GPIO_RX_WIDTH[#-1]-1:0]	0	Output	N/A	GPIO # output

Note: # denotes the number of respective channels configured.

3. Detailed Description

This RD can take up to 7 TX/RX channels to aggregate and communicate over a single wire between two FPGAs. The two FPGAs have to be properly configured such as number of channels, data content on specified channels, data width, to transmit and retrieve proper information. The Single-wire link must be pulled up by the external strong resistor, for example, 200 Ω. The ~24 MHz clock is generated by the internal oscillator and geared up to ~60 MHz by the on-chip PLL. For design with I²S channel, an external clock must be used to be feed to the PLL. This ~60 MHz clock is used as a sampling clock on the RX side and ~15 MHz clock is used as TX clock. Two TX clock cycles are required for bi-phase mark coding to transmit one-bit data. Therefore, the transmission data rate is ~7.5 Mbps. In case of iCE40 devices, the tolerance of the internal oscillator is ± 10%, which means both RX sides have to assume -~20 % to +~20 % of clock frequency difference of the opponent clock speed with regards to their own clock speed. The clock speed limitation depends on channel configuration.

The link must have a strong pull-up resistor, since it is not driven high for whole *1* period. FPGA drives the link low for whole *0* period, but drives high only for a short time (< 10 ns of beginning of *1* period).

3.1. Link Establishment upon Power up and Reset Release

When the FPGAs are powered up and reset is released after configuration, the internal oscillator begins generating ~24 MHz clock (for design with I²S channel, external clock is used) and PLL generates ~60 MHz clock using the oscillator clock. This clock is divided by 4 to provide TX clock of ~15 MHz. Figure 3.1 shows the transactions for Link Establishment. After PLL is locked and proper TX clock is generated, the Master FPGA pulls the link low for 3 TX clock cycles to be discovered by the Slave FPGA. It repeats this in every 32 clock cycles until it detects 5 cycles long or more of link = 0 as a sign of connection acknowledgement. Upon the reset release, the Slave FPGA waits for link = 0 for 2 TX clock cycles long, then pulls the link low for 7 TX clock cycles.

Figure 3.1. Link Establishment

For I²S application, I²S clock learning/training is done after link acknowledgement. To achieve this, a Master or a Slave FPGA I²S sends eight SCK pulses on the link, which the other FPGA receives and processes as shown in Figure 3.2.

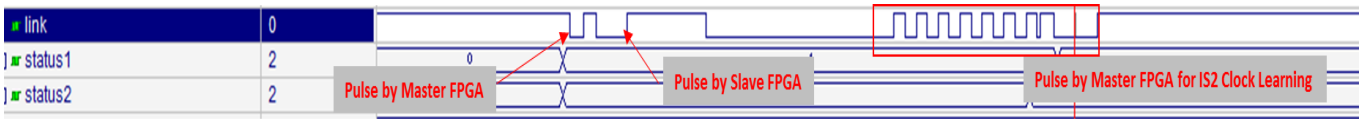


Figure 3.2. I²S Clock Training

3.2. Link Status

Single-wire provides two-bit status outputs to indicate four conditions:

- 00 : powered up with rst_n = 0
- 01 : Discovery stage, in which the FPGA is trying to establish the Single-wire connection.
- 10 : Connected state, shows the link connection by the pulse exchange shown in the Link Establishment upon Power up and Reset Release section.
- 11 : Active state indicates the payload data transmission on Single-wire.

3.3. TX Rights Negotiation

Figure 3.3 shows an example of TX rights negotiation between two FPGAs. After the link connection is confirmed, both sides can request the TX transaction by pulling the link low for two TX clock cycles. The other side pulls the link low for 5 TX clock cycles as a grant. In case that both sides send TX request at the same time, the long pulse cannot be detected on both sides. In that case, the side previously on RX side gets the TX right and send TX request pulse again. The other side does not send the TX request pulse again, and waits for TX request pulse coming from the other side, then send the grant pulse. If this case happens in the very first transaction after reset release, Slave FPGA will give up sending a new TX request pulse and Master FPGA will have the TX rights.

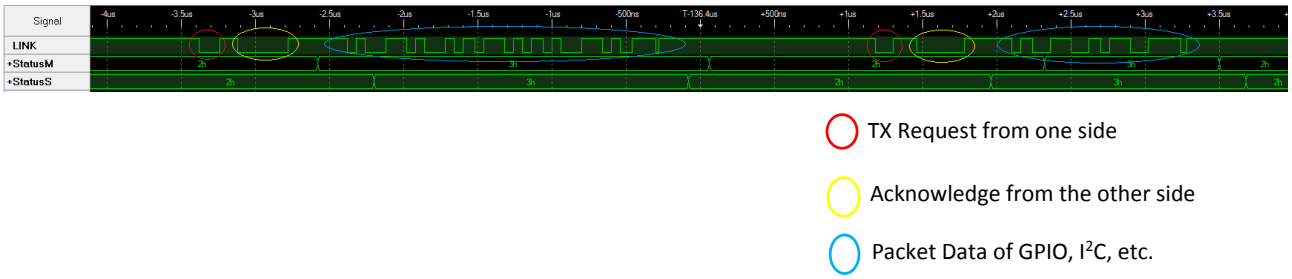
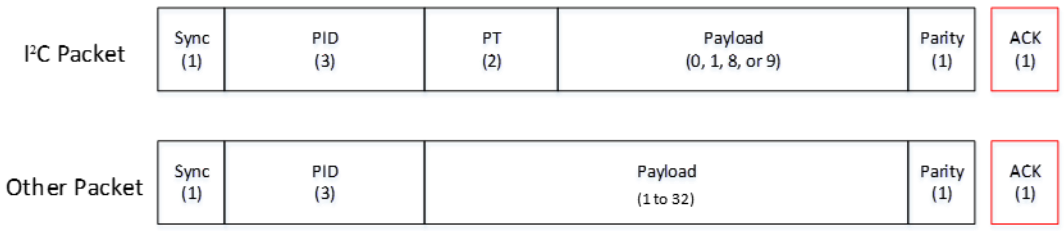


Figure 3.3. TX Rights Negotiation and Packet Transmission

3.4. Packet Transmission

Single-wire employs packet-based TDM data transmission. Figure 3.4 shows a packet structure. Every packet has a start bit, payload ID (PID), and a parity bit. The length of the payload data depends on the PID, Payload Type (in case of I^2C), and data width (in case of GPIO). Packet structure is different between I^2C, I^2C and GPIO. In case of I^2C, two-bit payload type (PT) indicates the type of payload, since those payloads have different data lengths. PID assignments have to be matched between both FPGAs. Otherwise, the RX side cannot retrieve the correct data. These assignments are compile options and cannot be changed dynamically. Parity polarity is determined by the payload length to end the parity bit as high all the time. After the completion of the packet transmission, RX side returns a short pulse, 4 cycles of Rx clock, as an acknowledge bit (ACK) to notify Parity check is OK. TX side retransmits the same packet data again if it does not receive ACK from RX side. Please note for I^2S data, retransmit the data is not possible.

Bi-phase mark encoding is used to transmit the packet data. Figure 3.5 shows an example of Bi-phase mark encoding and Figure 3.6 shows an example of bit pattern of I^2C packet. The link status is always high in idle state. Therefore, the Sync bit, data = 1, is always encoded as 01 followed by PID data. Even parity is used when the number of payload bit is even. Odd parity is used when the number of payload bit is odd. In this method, the parity bit pattern is either 01 or 11, so ACK bit can be easily recognizable on TX side. Two TX cycles are assigned to detect ACK bit on TX side considering the clock phase difference and frequency tolerance between two FPGAs.



Remarks:
- ( ) denotes bit length.
- PID : Payload ID, PID = 7 is reserved and cannot be used for customer purposes.
- PT : Payload Type for I²C
  Mater-to-Slave --- 00 : Start/Repeated start with byte data(8), 01 : write data(8), 10 : ACK/NACK bit(1), 11 : stop (0)
  Slave-to-Master --- 00 : ACK + read data (9), 01 : read data (8), 10 : ACK bit (1), 11 : reserved
- Payload length in non I²C packet is pre-determined by the data width associated with PID.
- Parity : Even Parity is used when payload length is even; Odd Parity is used when payload length is odd
- ACK : ACK bit is returned from RX side to TX side when Parity Check on Rx side is OK. TX side retransmits the same packet if it does not receive ACK bit.

Figure 3.4. Packet Structure

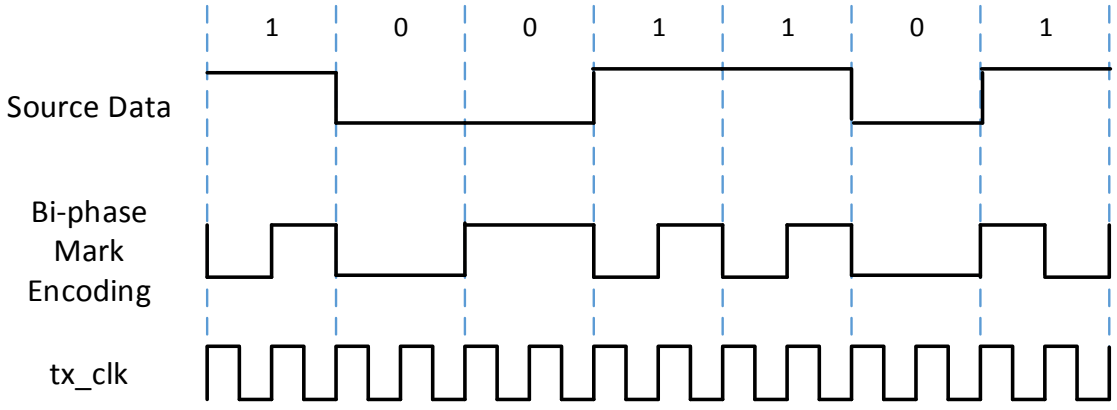


Figure 3.5. Example of Bi-phase Mark Encoding

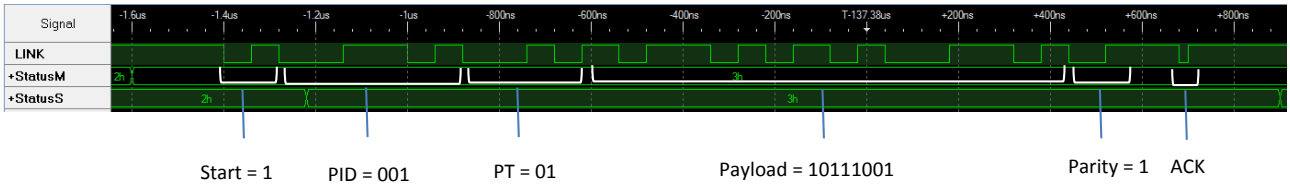


Figure 3.6. Example of I²C Packet

3.5. TX Rights Release

TX side can send the packet of all channels if those are ready to be sent once that FPGA obtains the TX rights. Starting from lower number of the channels, it keeps sending packets one after another until all available TX channel data are sent. After that, that FPGA releases TX rights. Therefore, a new negotiation is necessary when it needs to send the next data. RX side sets the waiting period after it returns ACK for the current TX data reception. If it does not receive the start bit within that period and has the internal TX requests, it sends TX request to the other side.

3.6. System Level I²C Transactions

Figure 3.7 and Figure 3.8 show an example of system-level I²C transactions. Two I²C master devices are connected to Single-wire Master FPGA, such as SCL1M/SDA1M and SCL2M/SDA2M, and two I²C slave devices are connected to Single-wire Slave FPGA, such as SCL1S/SDA1S and SCL2S/SDA2S. In Figure 3.7, both I²C masters issue Start command followed by I²C address 0x60 and write command. Then SCL is pulled low, clock stretching, by Single-wire Master, while *Start Command + I²C address + write command* is forwarded to I²C slave device through the link. Single-wire Slave FPGA, which makes I²C ACK from I²C slave device forwarded to I²C master device through Single-wire Slave FPGA, link, and Single-wire Master FPGA. In other words, I²C master device SCL is held low after I²C master sends a byte data until I²C ACK comes from the other end through the link in case of write transactions. Figure 3.8 shows Repeated Start command and read transactions. In case of read transaction, master I²C SCL is held low until Master FPGA gets *I²C ACK+ read data* from slave side through the link. Since both sides have to replicate the transactions originated from other sides, I²C transactions take at least ×2 of time comparing to non-Single-wire configuration. The overheads by Single-wire depends on other link transactions.

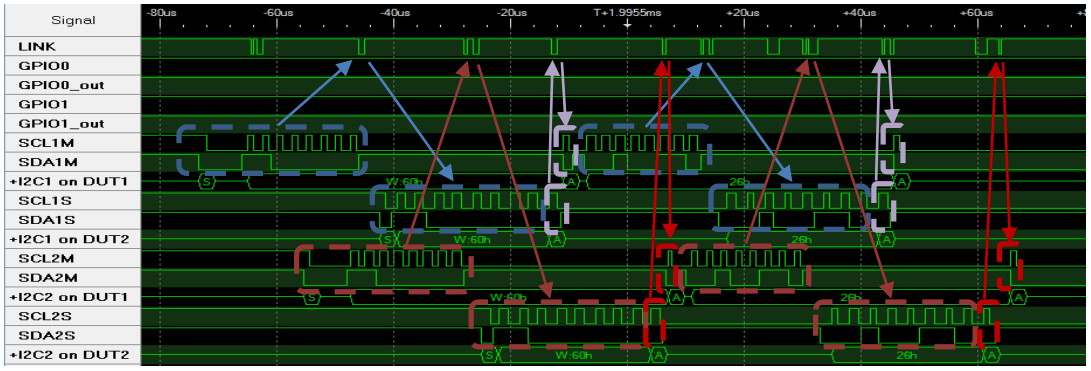


Figure 3.7. I²C Transaction #1 (Sub-address Write for Read Transaction)

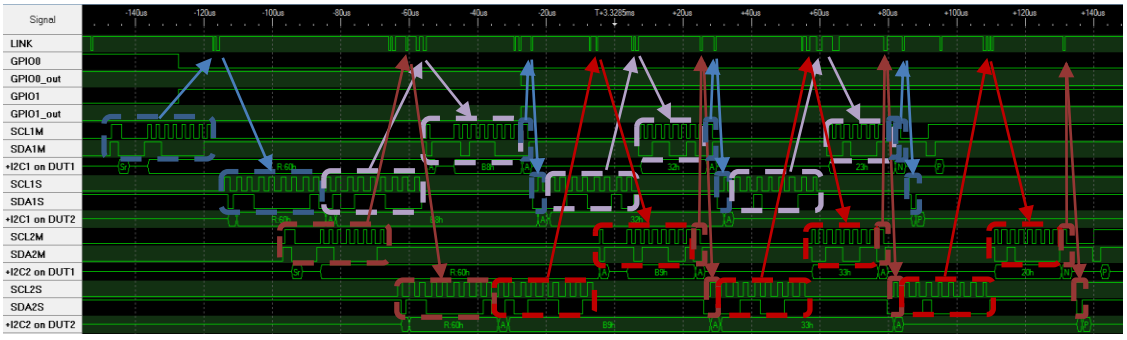


Figure 3.8. I²C Transaction #2 (Repeated Start Followed with Read Transaction)

Figure 3.9 and Figure 3.10 show the examples of link delay in case of I²C Start and I²C ACK. Actual delay time depends on several conditions, such as data sample timing, TX Request collision, link occupancy, TX cue, and can vary.

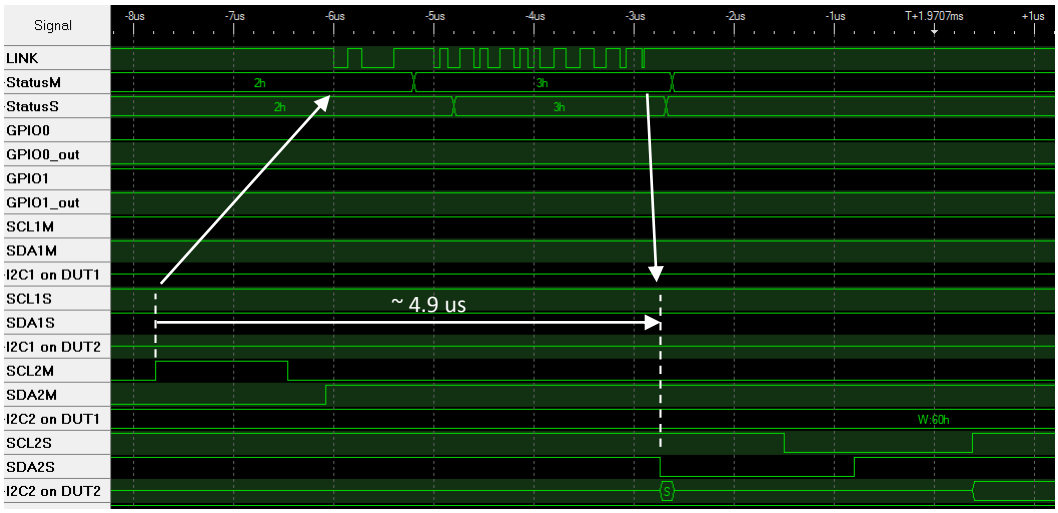


Figure 3.9. Link Delay Example #1 (I²C Start)

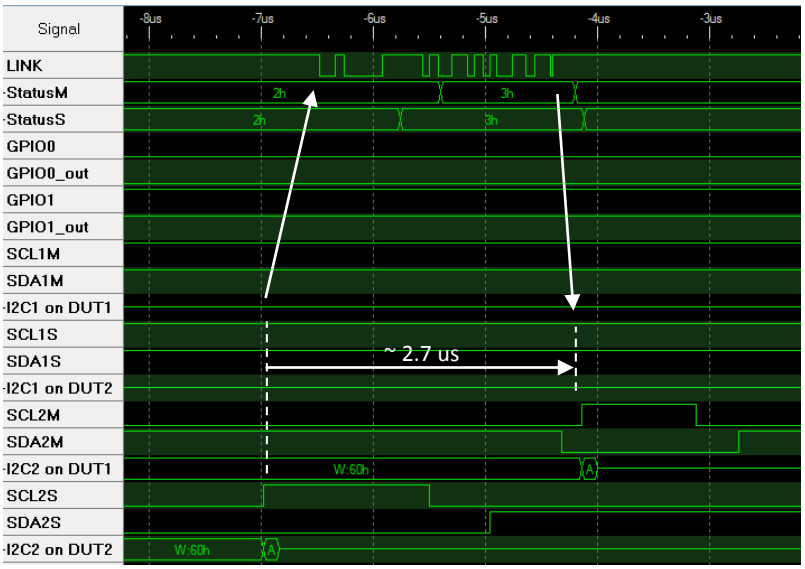


Figure 3.10. Link Delay Example #2 (I²C ACK)

3.7. System Level I²S Transactions

Figure 3.11 shows an example of a system-level I²S transaction. One I²S Transmitter is connected to the Master FPGA while its I²S Receiver is connected to the Slave FPGA. I²S data are sent to the Single wire link every Word line, which correspond to the I²S WS. If there are other types of data (I²C for example) connected on the system, data transmission is handle in a round robin manner. For example, in Figure 3.11, the I²C Packet is sent to the Single line after the first packet of I²S data is sent to the Single wire.

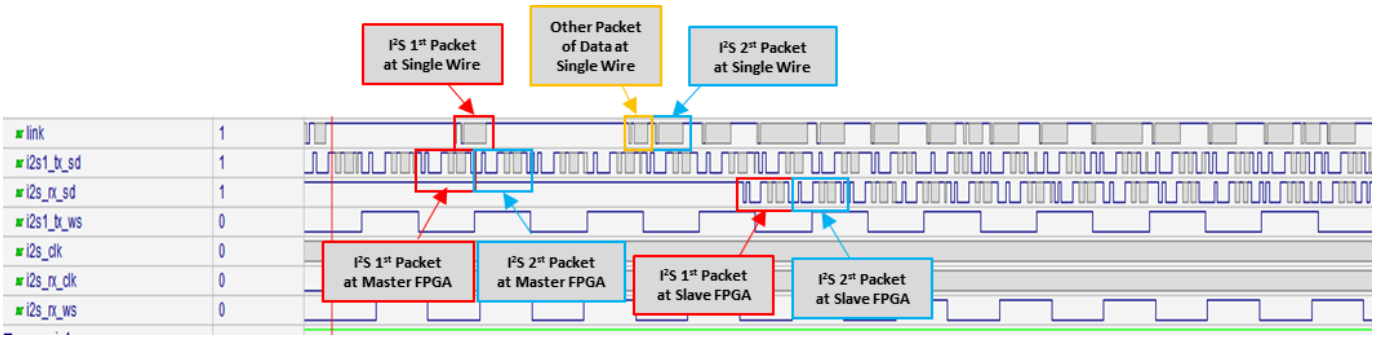


Figure 3.11. **System Level I²S Transaction**

Figure 3.12 shows an example of I²S delay from Master to Slave FPGA with a sample rate of 48 kHz, 32 bits I²S word length. Delay may vary according to the other configurations attached on the link.

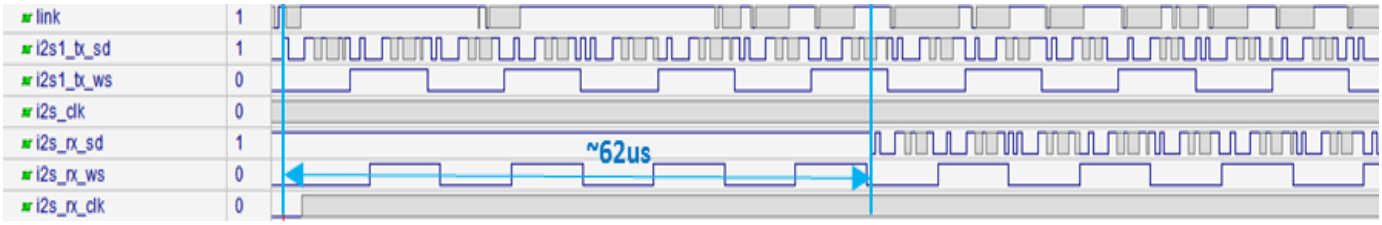


Figure 3.12. I²S Delay from Master to Slave FPGA

4. Packaged Design

Single-wire Signal Aggregation Reference Design for iCE40 UltraPlus is available on latticesemi.com. Figure 4.1 shows directory structure. The design is targeted for iCE40UP5K-SG48I and configured for one I^2S, two I^2C and 2 bits TX and RX GPIOs:

CH #0: I^2S Master on Master FPGA and I^2S Slave on Slave FPGA

CH #1 : I^2C Master on Master FPGA and I^2C Slave on Slave FPGA

CH #2 : I^2C Master on Master FPGA and I^2C Slave on Slave FPGA

CH #3 : GPIO TX (2 bits) & RX (2 bits) on both FPGA

There exist two projects for Master FPGA and Slave FPGA. The user can change the configuration by modifying compiler directives and parameters in two top-level Verilog files; singlewire_master.v and singlewire_slave.v.

Functional simulation setup for Aldec Active-HDL is also included. The script can be executed from Active-HDL window through Radiant. The testbench and related files need to be modified for different configurations along with two top-level Verilog design files.

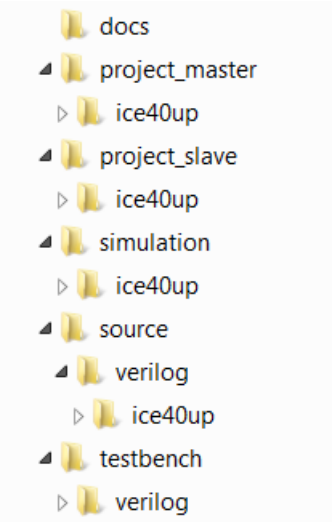


Figure 4.1. Packaged Design Directory Structure

5. Resource utilization Resource Utilization

Resource utilization depends on the configurations. Table 5.1 shows the utilizations under some typical configurations by iCE40 UltraPlus. Actual usage can vary.

Table 5.1. Resource Utilization Examples

Configuration	FPGA	LUT	FF	EBR	PLL	I/O
CH#0 : I^2C (Master on M) CH#1 : GPIO (1 bit Interrupt), S to M only	M	545	261	0	1	7
	S	626	286	0	1	7
CH#0: I^2S; CH#1: I^2C (Master on M) CH#2: I^2C (Master on M) CH#3: GPIO (4 bits), M to S only	M	892	449	1	1	13
	S	1030	494	1	1	13

Note: M denotes Master FPGA. S denotes Slave FPGA.

References

For more information on FPGA device, visit http://www.latticesemi.com/Products/FPGAandCPLD/iCE40UltraPlus

For complete information on Lattice Radiant Project-Based Environment, Design Flow, Implementation Flow, and Tasks, as well as on the Simulation Flow, see the Lattice Radiant User Guide.

Technical Support Assistance

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

Revision History

Revision 1.1, September 2018

Section	Change Summary
Introduction	• Updated Features List section. • Updated Figure 1.1. Single-wire Block Diagram Example.
Parameters and Port List	• Updated Table 2.1. Top-level Compiler Directives. Added a few rows at the end of GPIO Rx channel count category. • Updated Table 2.2. Top-level Parameters. Added a few rows at the end of GPIO_RX_MAX_WIDTH [3:0]. • Updated Table 2.3. Configuration Example 1. Added a few rows at the end of Compiler Directives and Parameters. • Updated Table 2.4. Configuration Example 2. Added a few rows at the end of Compiler Directives and Parameters. • Added Table 2.5. Configuration Example 3. • Updated Table 2.6. Single-wire Top-Level I/O. Added I²S Interface section.
Detailed Description	• Updated text in Detailed Description section. • Updated text in Link Establishment upon Power up and Reset Release section. • Added Figure 3.11. System Level I2S Transaction • Added Figure 3.12. I²S Delay from Master to Slave FPGA.
Packaged Design	Updated text in Packaged Design section.

Revision 1.0, May 2018

Section	Change Summary
All	Initial release.

7th Floor, 111 SW 5th Avenue
Portland, OR 97204, USA
T 503.268.8000
www.latticesemi.com