I^{2}C Controller IP

User Guide

System Level Solutions, Inc. (USA)
14100 Murphy Avenue
San Martin, CA 95046
(408) 852 - 0067

http://www.slscorp.com

IP Core Version: 3.0
Document Version: 1.3
Document Date: June 2009

IP Usage Note

The Intellectual Property (IP) core is intended solely for our clients for physical integration into their own technical products after careful examination by experienced technical personnel for its suitability for the intended purpose.

The IP was not developed for or intended for use in any specific customer application. The firmware/software of the device may have to be adapted to the specific intended modalities of use or even replaced by other firmware/software in order to ensure flawless function in the respective areas of application.

Performance data may depend on the operating environment, the area of application, the configuration, and method of control, as well as on other conditions of use; these may deviate from the technical specifications, the Design Guide specifications, or other product documentation. The actual performance characteristics can be determined only by measurements subsequent to integration.

The reference designs were tested in a reference environment for compliance with the legal requirements applicable to the reference environment.

No representation is made regarding the compliance with legal, regulatory, or other requirements in other environments. No representation can be made and no warranty can be assumed regarding the suitability of the device for a specific purpose as defined by our customers.

SLS reserves the right to make changes to the hardware or firmware or software or to the specifications without prior notice or to replace the IP with a successor model to improve performance or design of the IP. Of course, any changes to the hardware or firmware or software of any IP for which we have entered into an agreement with our customers will be made only if, and only to the extent that, such changes can reasonably be expected to be acceptable to our customers.

ug_ipi2cc_1.3

About this Guide

Introduction

This user guide informs you how to use I^2C Controller IP with the Nios II processor and develop applications.

Table below shows the revision history of this user guide.

Version	Date	Description
1.3	June 2009	Updated document as per new refernce design.
1.2	May 2009	Updated document for IP Core version 3.0.
1.1	March 2008	Updated document for the Quartus 7.2
1.0	July 2007	First Publication of the I^2C Controller IP User Guide

How to find Information

- The Adobe Acrobat Find feature allows you to search the contents of a PDF file. Use Ctrl + F to open the Find dialog box. Use Shift + Ctrl + N to open to the Go To Page dialog box.
- Bookmarks serve as an additional table of contents.
- Thumbnail icons, which provide miniature preview of each page, provide a link to the pages.
- Numerous links shown in Navy Blue color allow you to jump to related information.

How to Contact SLS

For the most up-to-date information about SLS products, go to the SLS worldwide website at http://www.slscorp.com. For additional information about SLS products, consult the source shown below.

Information Type	E-mail
Product literature services, SLS literature services, Non-technical customer services, Technical support.	support@slscorp.com

Typographic Conventions

The user guide uses the typographic conventions as shown below:

Visual Cue	Meaning
Bold Type with Initial Capital Letters	All headings and Sub headings Titles in a document are displayed in bold type with initial capital letters; Example: **Block Diagram.**
Bold Type with Italic Letters	All Definitions, Figure and Table Headings are displayed in Italics. Examples: **Table 5-1. Register Details,** **Figure 2-2. I^{2}C Controller IP Architecture**
1., 2.	Numbered steps are used in a list of items, when the sequence of items is important. such as steps listed in procedure.
• ■	Bullets are used in a list of items when the sequence of items is not important.
☞	The hand points to special information that requires special attention
⚠ CAUTION	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
⚠ WARNING	The warning indicates information that should be read prior to starting or continuing the procedure or processes.
👣	The feet direct you to more information on a particular topic.

Contents

1. Introduction

I^{2}C (Inter-Integrated Circuit) Controller is a two-wire, bi-directional serial bus that provides a simple and efficient method of data transmission over a short distance between many devices. Avalon complient I^{2}C Controller IP core provides an interface between a Nios processor and an I^{2}C device. It can work as a master/Slave transmitter or Master/Slave Receiver depending on working mode determined by Nios processor. The I^{2}C Controller IP core incorporates all features required by the latest I^{2}C specification including clock synchronization, arbitration, multi-master systems and Fast-speed transmission mode.

The I^{2}C Controller IP core can be configured as a **Master/Salve**. This user guide will provide you with some basic technical details of I^{2}C Controller device core acting as Avalon slave in Altera's SOPC builder. SOPC builder is a software tool that allows for the creation of a Nios II system module or a more general Multi-master system On A Programmable Chip Module. A complete Nios II system module contains a Nios II (soft core 32 bit RISC) processor and its associated system peripherals.

Features

- Avalon Bus Compliant
- Compatible with Philips I^{2}C(PCF 8584) standard
- Supports both Master and Slave mode
- Automatic detection and adoption to bus interface type
- Multi-master Operation
- Byte-by-byte data-transfers is driven by Interrupt or Bit-polling
- Arbitration-lost interrupt with automatic transfer cancellation
- Start/Stop/Repeated Start/Acknowledge generation
- Start/Stop/Repeated Start detection
- Bus-Busy detection
- Supports 7 bit addressing mode
- Operates from a wide range of input clock frequencies

- Static synchronous design
- Implementation in Verilog RTL
- Fully Synthesizable

☞ The I^{2}C Controller core is tested with Real Time Clock (RTC) as I^{2}C Master and NIOS processor as Master on UP3 Cyclone Edition board.

Core Resources

Table 1-1 shows the LE usage of the core.

Table 1-1. Core LE Usage Summary

Device Name	LE	Memory Bits
Cyclone	278	0
Cyclone II	278	0
Cyclone III	277	0

2. Core Architecture

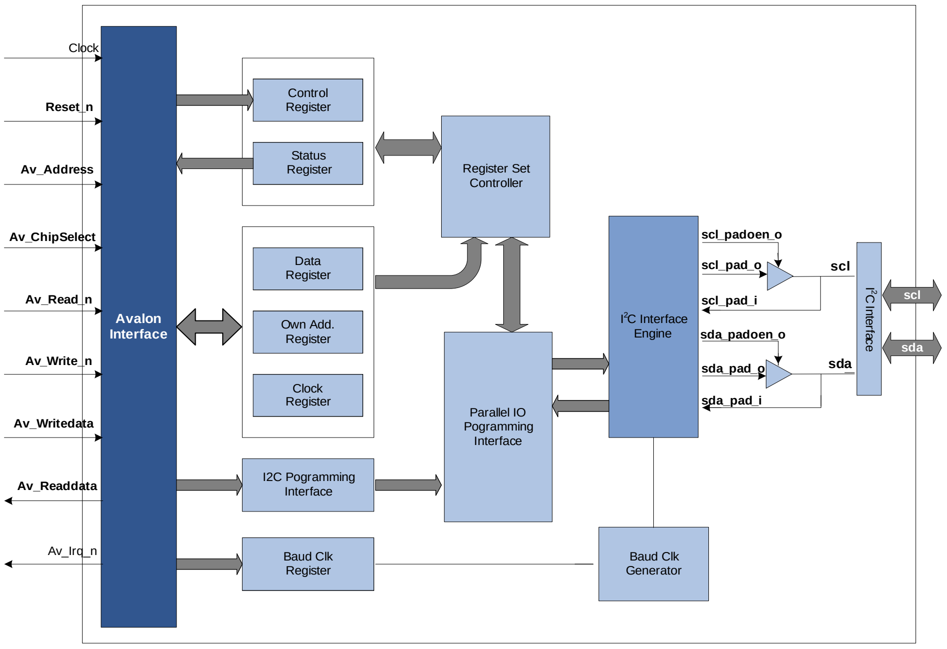
Block Diagram

The I^{2}C Controller IP is built around five primary blocks:

- Baud Clock Generator
- Register Set Module
- Parallel IO Programming Interface
- I^{2}C Programming Interface
- I^{2}C Interface Engine

All other blocks are used for interfacing or for storing temporary values. See Figure 2-1.

Figure 2-1. I^{2}C Controller IP Architecture



Baud Clock Generator

Input clock to I^2C base module is variable. A programmable clock divider is employed to generate a fixed frequency to the interface

The clock divider is programmed for different frequencies. You must set the dividing factor such that the input clock provides a frequency as close to 25 MHz as possible.
(For e.g., system_clk/25 Mhz = dividing factor)

Register Set Module

The register set module contains the register set, which can be accessed by the parallel interface side of the controller. This includes the following register:

- S0d - Data Shift register
- S0a - Own address register
- S1c - Control register
- S1s - Status register
- S2 - Clock register

Parallel IO Programming Interface

The parallel IO programming interface module processes the programming interface commands for register access. It dose not assume a synchronous strobe interface but will work with either synchronous or asynchronous strobes. It also contains register to select synchronization logic.

I^2C Programing Interface

The I^2C programing Interface module sets the different register address like following register:

- Baud Clock register.
- Own address, Data register, and Clock register.
- Status register and Control register.

I^2C Interface Engine

The I^2C Interface Engine Module executes the actual I^2C transfers both Master and Slave. It also accomplishes arbitration, handshaking and filtering. This module includes the state machines which follows and generates the I^2C serial protocol

This module generates the I^2C Clock (SCL). This is derived from the clk_i intermediate clock which is normally 25 MHz.

There are five levels of I^2C bus tracking employed:

- Operation Handshake

- Transaction tracking
- Macro frame tracking
- Micro frame tracking
- Signal tracking

Operation Handshake

The operation handshake logic detects START, STOP commands from the parallel interface.

Transaction Tracker

The transaction tracker determines the overall mode and direction of the transaction. The modes are SRM, STM, MRM, and MTM. These are defined as follows:

SRM = Slave Receive Mode (DEFAULT)

STM = Slave Transmit Mode

MRM = Master Receive Mode

MTM = Master Transmit Mode

Macro Frame Tracker

The macro frame tracker keeps track of the phases within the frame, and also the type of frame, ie.address or data.

Micro Frame Tracker

The Micro Frame tracker (“tracker“) follows the I^2C clock and data inputs. These provide the ability to follow the progress of the frame as it proceeds. Thus by examining the tracker, the sequence of bit can be determined.

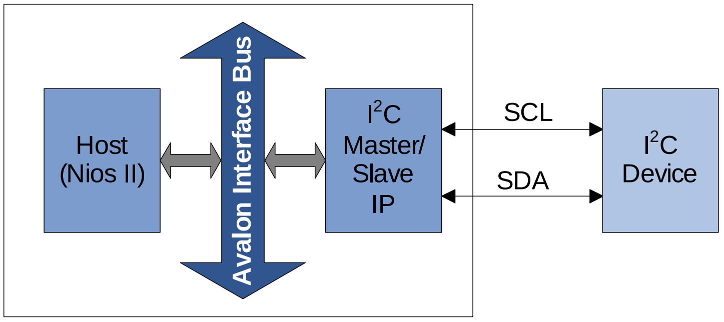
Signal Tracker

Signal tracker determines the start, stop, and clock change conditions which transpire on the bus. The control counter is used to settle and validate bus signals.

3. Operation

Figure 3-1. shows a block diagram of avalon system which makes use of I^{2}C Controller IP core.

Figure 3-1. I^{2}C Controller IP Core Block Diagram with Avalon System



It mainly consists of three blocks:

Host

It act as master in whole system which decides to transmit/receive data to/ from salve device in master mode and master device in slave mode.

Avalon Interface Bus

Avalon interface bus is ALTERA’s standard interface bus used in SOPC Builder, through which all avalon compatible cores can easily communicate. Please refer chapter 4 for more detail on Avalon Interface.

I^{2}C Controller IP

This IP acts as a mediator between Host and I^{2}C device (Master/Slave). It accepts data from I^{2}C device when it is asked by Host and interrupts host when any data is received from device.

4. Registers

This section describes all registers inside I^{2}C Controller IP core. Registers are mainly used for:

- Configuring I^{2}C Controller IP core.
- Communicating between I^{2}C Controller & Nios II CPU.

Table 4-1 shows the register details. The **Registers** field describes the name of the register. The A**ddress** field describes the address of the register in hex. The **Access** field describes the type of access to the registers i.e. read or write. **Width** field describes the bit occupied by the registers.

Table 4-1. Register Details

Registers	Address	Width	Access
Baud Clock Register	0X02	4	W
Data Register (S0)	0X00	8	RW
Own address Register (S0’)	0X00	8	RW
Control Register (S1)	0X01	8	W
Status Register (S1’)	0X01	8	R
Clock Register (S2)	0X00	8	RW

Baud Clock Register

This register programs the clock frequency. It receives different frequencies as per the equation shown as below:

system_clk/25 MHz = dividing factor

When this register is set according to the following Table 4-2 , generates a 25 MHz Clock to the I^{2}C core.

☞ For the sake of clarity, the baud register is explained before the internal registers of the I^{2}C core.

Table 4-2. Clock Dividing Factor for Register Settings

Reg Setting	Dividing Factor
00	1
01	2
02	3
03	4
04	5
05	6
06	7
07	8
08	9
09	A
0A	B
0B	C
0C	D
0D	E
0E	F
0F	10

☞ Change the value of the Baud register only when the core is disabled.

Data Register

This register acts as a transmitter and receiver. While data is transmitted from I^{2}C Master to I^{2}C Slave they are written to this register. Data stored in this registers are transmitted only, when bit 7,bit 6,bit 2 and bit 0 in Control register is ‘1’. While data register receives data only when bit 6 of Control register is ‘1’ and bit 0 of Control register is ‘0’. Table 4-3 shows the details of Data register.

Table 4-3. Data Register (S0) Details

Bit	Access	Description
7:0	R/W	Data shift register/read buffer • This register acts as a serial shift register in receive mode and buffer register in transmit mode. • All read/write operations to/from the I^2C bus are done via this register. • I^2C bus data is always shifted in or out of shift register is copied to the S0 register during the acknowledge phase. Further reception of data is inhibited (SCL held Low) until the S0 read buffer is read.

Note: to Table 4-3

1). Reset Value: 0x00

Own Address Register

Sets the slave's own address. This register must be loaded with the 7-bit I^2C bus address to which the own address is to respond. During initialization, the own address register S0' must be written to, regardless whether it is later used. The Addressed As Slave (AAS) bit in status register S1 is set when this address is received (the value in S0 is compared with the value in S0'). Note that the S0 and S0' registers are offset by one bit; hence, programming the own address register S0' with a value of 55H will result in the value AAH being recognized. Programming of S0' is accomplished via the parallel-bus when A0 is LOW, with the appropriate bit combinations set in control status register S1 (S1 is written when pin A0 = HIGH). After reset, S0' has default address 00H. Table 4-4 shows the Own Address Register details.

Table 4-4. Own Address Register (S0') Details

Bit	Access	Description
7:0	R/W	Sets the slave's own address

Note: to Table 4-4

1). Reset Value: 0x00

Control register

The core responds to new commands only when the 'ES0' (Enable Serial Output) bit is set and pending commands are finished. Refer Table 4-5 for control register details. 'ES0' bit is cleared only when no transfer is in progress, i.e. after a STOP command, or when the control register has the STO bit set. When halted during a transfer, the core can hang the I^{2}C bus. The bit value stored in the control register indicates whether the I^{2}C IP core behaves as a Transmitter or Receiver. Start and Stop bits are generated using this register. Table 4-5 shows the control register details.

Table 4-5. Control Register (S1) Details

Bit	Access	Description
7	W	**PIN** Pending Interrupt Not Writing 1 causes all status bit resets to Logic 0 (software reset)
6	W	**ESO** Enable Serial Output This bit enables or disables the serial I^{2}C bus I/O. when ESO is LOW register access for initialization is possible when ES0 is high I^{2}C bus Communication is enabled
5:4	W	**Es1** and **Es2** Enable Serial1 and Enable Serial2 These bits control the selection of other registers for initialization and control. After these bits are programmed, access to the desired register is done through address 0

Table 4-5. Control Register (S1) Details

Bit	Access	Description
3	W	**ENI** Enable Interrupt This bit enables the external hardware-interrupt output. Interrupt is generated when the PIN bit is active i.e. Logic 0
2	W	**STA** Start This bit generates a start condition and initiates transmission of slave address
1	W	**STO** Stop This bit generates a stop condition
0	W	**ACK** Acknowledgement This bit must be set normally to logic 1. It causes the I^{2}C-bus controller to send an acknowledge automatically after each byte (during 9th clock pulse).This bit must be reset when operating in master receiver mode and requires no further data to be sent from the slave transmitter. It cause a negative acknowledge on the ^{I2}C-bus, which halts further transmission from the slave bus.

Note: to Table 4-6

1). Reset Value: 0x00.

2). To access the internal register S0, S0’, S2 we must first update the control register bits ES0, ES1, ES2 to point to the appropriate register.

☞ One can only write to the Control register

Register Access Control

Access other registers as per the settings shown in the following Table 4-6 .

Table 4-6. Register Access Control Details

Register Name	Serial Interface	ES0	ES1	ES2	Access
Control Register (S1)	Off	0	0	X	R/W
Own address Register (S0’)		0	0	0	R/W
Clock Register (S2)		0	1	0	R/W
Control Register (S1)	On	1	0	X	W
Status Register (S1’)		1	0	X	R
Data register (S0)		1	0	0	R/W
Clock Register (S2)		1	0	X	R/W

Note: to Table 4-6

1). With ES0 = 0, bits ENI, STA, STO and ACK of S1 can be read for test purposes.

2). ‘X’ if ENI = 0.

Table 4-7 shows the instructions for serial bus control.

Table 4-7. Instructions for Serial Bus Control

STA	STO	PRESENT MODE	FUNCTION	OPERATION
1	0	SLV/REC	Start	transmits (START + address), remains MST/ TRM if $R/\overline{W} = 0$ Go to MST/REC if $R/\overline{W} = 1$
1	0	MST/TRM	Repeat Start	same as for SLV/REC
0	1	MST/REC; MST/TRM	Stop Read; Stop Write	transmission STOPs and go to SLV/REC Mode.
1	1	MST	Data Chaining	sends STOP,START bit and address after last master frame without STOP sent; See *Note 2*

Table 4-7. Instructions for Serial Bus Control

0	0	ANY	Nop	No operation

Note: to Table 4-7

1). In master receive mode, the last byte be terminated with ACK bit HIGH ('negative acknowledge').

2). If both STA and STO are set HIGH simultaneously in master mode, a STOP condition followed by a START condition + address will be generated. this allows 'chaining' of transmissions without relinquishing bus control

Status Register

This register shows the status of the Transmit & Receive register. Table 4-8 shows the status register details.

Table 4-8. Status Register (S1') Details

Bit	Access	Description
7	R	**PIN** Pending Interrupt Not Reading from S0 sets this bit to 1 in receiver mode. writing to S0, sets this bit to 1 in transmitter mode. Sets STA bit to set this bit to 1 in transmitter mode.(Read indicates status when read with A0 = 1) It is used for polling the PIN bit if the serial transmission/reception has been completed. PIN bit is set to 0 after a transmission of a byte as transmitter or after reception of a byte as a receiver. This bit is set to 0, if I^{2}C bus STOP condition occurs as a slave or if bus error occurs.
6	R	Reserved bit
5	R	**STS** Slave Stop When in slave receiver mode, this flag is asserted when an externally generated STOP condition is detected (used only in slave receiver mode).

Table 4-8. Status Register (S1') Details

Bit	Access	Description
4	R	**BER** Bus Error a misplaced START or STOP condition has been detected. resets $\overline{BB}$ (to logic 1; inactive), sets PIN =0(active).
3	R	**LRB/AD0** Last Receiver Bit or Address 0 Bit This status bit serves as a dual function, and is valid only while PIN = 0 1. LRB holds the value of the last received bit over the I^{2}C bus while AAS = 0 (not addressed as slave) normally this will be the value of the slave acknowledgement; This checking for acknowledgement is done via testing of LRB. 2. AD0 when AAS = 1(addressed as slave condition) the I^{2}C-bus controller has been addressed as slave. Under this condition, this bit becomes the AD0 bit and will be set to logic 1 if the slave address received was the general call(00) address or logic 0, if it was I^{2}C-bus controller own slave address.
2	R	**AAS** Address as a Slave Bit. Valid only when PIN = 0. When acting as a slave receiver, this flag is set when an incoming address over the I^{2}C-bus matches the value in own address register S0' or if the I^{2}C-bus general call address(00) has been received.
1	R	**LAB** Lost Arbitration Bit This bit is set when, in multi-master operation, arbitration is lost to another master on the I^{2}C bus

Table 4-8. Status Register (S1') Details

Bit	Access	Description
0	R	$\overline{BB}$ Bus Busy bit This is an active low bit. This is a read only flag When asserted low, indicates that the bus is busy and access is not possible. This bit is set and reset by the STOP/START conditions.

Note: to Table 4-8

1). Reset Value: 0x00

2). Please note that all reserved bits are read as zeros. To ensure forward compatibility, they should be written as zeros.

☞ One can only read from the status register

Clock Register

This register is used to SCL clock frequency. It provides Up to 50 KHz to 1.5 MHz. User can choose the SCL clock speed as per the following equation:

CLK = 25 MHz, Desired SCL = 1.5MHz (As per user's choice), then

$$ClockRegister = (25MHz)/(1.5MHz \times 2) = 8.3(dec) = 8(hex)$$

User can use up to 8 bit of register to select SCL frequencies. See Table 4-9 .

Table 4-9. Clock Register (S2) Details

Bit	Access	Description
7:0	W	Sets the SCL clock frequency

5. I^{2}C Controller IP Interface Signals

Master Interface Signals

I^{2}C Interface uses a Master/Slave interface signals for data transfer between I^{2}C Controller IP and Avalon Bus.The IP features an AVALON compliant interface. All output signals are registered. Each access with 1 clock cycle latency.Table 5-1 describes the details of master interface signals for Avalon bus.

Table 5-1. Master/slave Interface Signals

Signal	Width	Direction	Description
clk	1	input	Master Clock
reset_b	1	input	Asynchronous reset active low
Av_address	8	input	Address
Av_write_b	1	input	Write enable active low
Av_read_b	1	input	Read enable active low
Av_writedata	8	input	Write data
Av_readdata	8	output	Read data
Av_chipselect	1	input	Chip Select Signal for the I^{2}C Core
Av_irq_b	1	output	Interrupt active low

I^{2}C Bus Physical Connection Signals

The I^{2}C interface uses a bi-directional serial data line (SDA) and a serial clock line (SCL) for data transfers. All devices connected to these two signals must have open drain or open collector outputs. Both lines must be pulled-up to VCC by external resistors. Table 5-2 describes the physical interface signals of I^{2}C Controller Interface

Table 5-2. Physical Interface Signals

Signal	Width	Direction	Description
SCL	1	Bi-directional	Serial Cock Line
SDA	1	Bi-directional	Serial Data

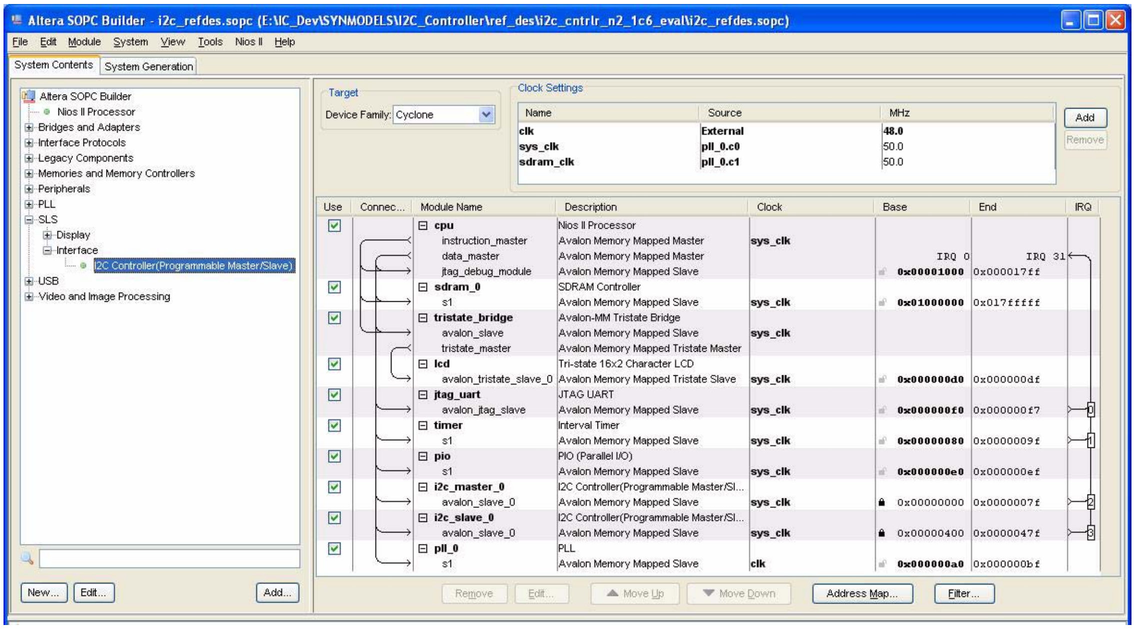
6. Using I^{2}C Controller IP in SOPC Builder

To use I2C Controller IP Core in SOPC Builder, follow the steps below:

1. Run **i2c_cntrlr_**<*licensetype*>**_v<version #>.exe**.

2. It will automatically place the I^{2}C Controller Component into SOPC Builder GUI. You can see that I^{2}C Controller component is now added in the final view of the SOPC builder system. See Figure 6-1. Where <*licensetype*> is the IP Core license type and <*version #*> is the IP Core version number. The licensetype can be ocp_eval or oc_full.

Figure 6-1. I^{2}C Controller IP Core in SOPC Builder



Altera SOPC Builder - i2c_refdes.sopc (E:\IC_Dev\SYNMODELS\I2C_Controller\ref_des\i2c_cntrlr_n2_1c6_eval\i2c_refdes.sopc)

Device Family: Cyclone

Name	Source	MHz
clk	External	48.0
sys_clk	pll_0.c0	50.0
sdram_clk	pll_0.c1	50.0

Use	Connec...	Module Name	Description	Clock	Base	End	IRQ
✓		cpu	Nios II Processor				
		instruction_master	Avalon Memory Mapped Master	sys_clk			
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31	
		jtag_debug_module	Avalon Memory Mapped Slave		0x00001000	0x000017ff	
✓		sdram_0	SDRAM Controller				
		s1	Avalon Memory Mapped Slave	sys_clk	0x01000000	0x017fffff	
✓		tristate_bridge	Avalon-MM Tristate Bridge				
		avalon_slave	Avalon Memory Mapped Slave	sys_clk			
		tristate_master	Avalon Memory Mapped Tristate Master				
✓		lcd	Tri-state 16x2 Character LCD				
		avalon_tristate_slave_0	Avalon Memory Mapped Tristate Slave	sys_clk	0x000000d0	0x000000df	
✓		jtag_uart	JTAG UART				
		avalon_jtag_slave	Avalon Memory Mapped Slave	sys_clk	0x000000f0	0x000000f7	0
✓		timer	Interval Timer				
		s1	Avalon Memory Mapped Slave	sys_clk	0x00000080	0x0000009f	1
✓		pio	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	sys_clk	0x000000e0	0x000000ef	
✓		i2c_master_0	I2C Controller(Programmable Master/Sl...				
		avalon_slave_0	Avalon Memory Mapped Slave	sys_clk	0x00000000	0x0000007f	2
✓		i2c_slave_0	I2C Controller(Programmable Master/Sl...				
		avalon_slave_0	Avalon Memory Mapped Slave	sys_clk	0x00000400	0x0000047f	3
✓		pll_0	PLL				
		s1	Avalon Memory Mapped Slave	clk	0x000000a0	0x000000bf	

3. Since the core is a slave, there must be an Avalon master. Usually, this master is the NIOS processor. Using the SOPC builder tool, include the NIOS processor (must have the NIOS development kit). Then add any other peripherals required for the final design and assign the base address and irqs.

☞ For further reference refer, I^{2}C Controller Core Registers Section

4. Run the provided C File, by uploading to the processor via the NIOS IDE or the ***.elf** through Cygwin bash shell and *nios-run* command for testing the behavior of I^{2}C Controller IP Core.

☞ The I^{2}C IP core will work on the basis of interrupting the NIOS processor. That means whatever operation, either read or write, to any of the specified device would be driven by the NIOS processor since it is the master. The I^{2}C core will deliver commands to the NIOS processor through interrupts. The SLS supplied Sample C file (Look under <*I^{2}C Controller Installation Directory*> \embedded\example\niosII**)** runs on top of the NIOS processor allowing it to read the specified registers and decode opcode and commands.