I2C-Master / Slave Core Specification

Authors

TooMuch Semiconductor Solutions (info@toomuchsemi.com)

Copyright & License

Revision No.

1.0

Disclaimer

Introduction

I^{2}C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. It is most suitable for applications requiring occasional communication over a short distance between many devices.

Devices controlling the buses are called as Master. Master is responsible for generation of bus control and synchronizing signals. Slaves just follow the Master. Any I^{2}C device can be either receiver or transmitter. The I^{2}C standard is a true multi-master bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously.

Design implementation

This design serves both as I^{2}C compatible master and slave. It has two state machines,first is the main state machine which is same for either master or slave , while second is responsible for generating clock signal in master mode. These state machines have been interface with a module which is used for decoding the commands from processor. On top of these, a module has been created which actually interfaces with the processor. The communication between design and processor follow the Wishbone protocol.

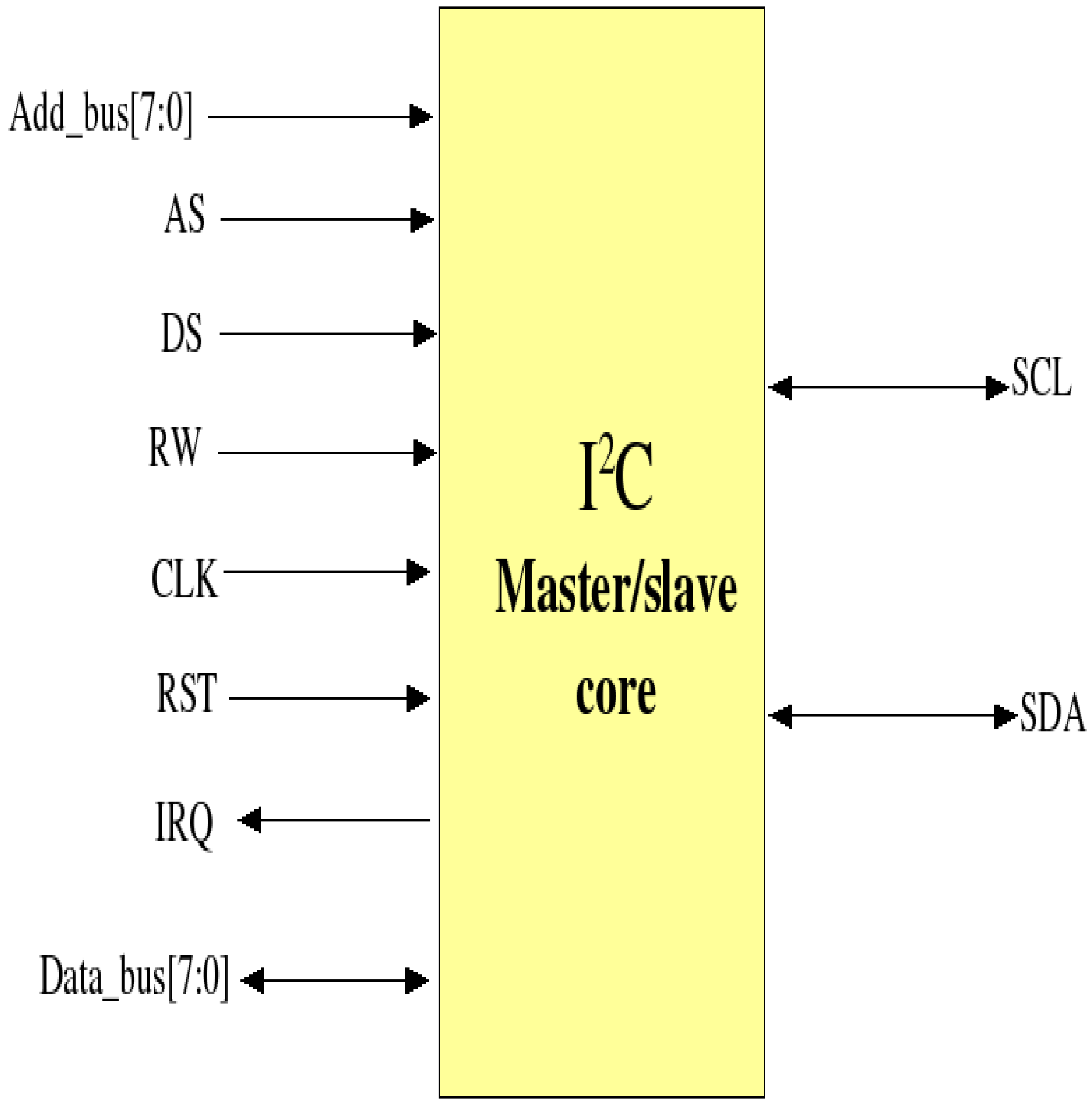
This Design supports the following functionalities:

- Both Master and slave operation
- Both Interrupt and non interrupt data-transfers
- Start/Stop/Repeated Start generation
- Fully supports arbitration process
- Software programmable acknowledge bit
- Software programmable time out feature
- programmable address register
- Programmable SCL frequency
- Soft reset of I2C Master/Salve
- Programmable maximum SCL low period
- synthesis core

Limitation

Design does not support clock stretching .

Pin Interfaces



Processor interface signals

Port	Width	Direction	Description
clk	1	input	system clock
reset	1	input	system reset
add_bus	[7:0]	input	processor address bus
data_bus	[7:0]	inout	processor data bus
as	1	input	When asserted indicates address present on bus is valid.
ds	1	input	When asserted indicates data on data bus is valid
rw	1	input	“1” indicates read,”0” indicates write,from processor side.
irq	1	output	interrupt bit for processor

I2C connections

Port	Width	Direction	Description
SCL_oe	1	out	Serial Clock enable line
SCL_o	1	out	Serial clock line output
SCL_In	1	In	Serial clock input line
SDA_oe	1	Out	Serial data enable line
SDA_o	1	Out	Serial data output line
SDA_In	1	In	Serial data input line

Registers Implemented

Name	Width	Access	Description
PRER	8	RW	Clock Prescale register
CTR	8	RW	Control register
SR	8	R	Status register
TO	8	W	Time out Register
ADDR	8	W	Address Register
DTR	8	RW	Data Transmit Register
DRR	8	R	Data Receive Register

Register description

Pre-scale Register

This register is used to prescale the SCL clock line. Due to the structure of the I^2C interface, the core uses a 4*SCL clock internally. The prescale register must be programmed to this 4*SCL bit rate. This is used for programming the SCL frequency.

Control register

Bit	**Access**	**Description**
7	RW	**EN:** I2C core enable bit. This bit must be cleared before any other control bit has any effect. If this bit is set then all state machines will go into reset state. When set to ‘0’ the core is enabled. When set to ‘1’ the core is disabled.
6	RW	**IEN:** I2C core interrupt enable bit. Processor will response to interrupt from core if this bit is set. When set to ‘1’ interrupt is enabled. When set to ‘0’ interrupt is disabled.
5	RW	**Mode select:** When processor changes this bit from 0 to 1,controller will generate a START Condition in and will be in Master mode. Changing this bit from 1 to 0, stop condition will be generated and controller will switch to slave mode. Design will be in slave mode if this bit is “0”
4	RW	**Transmit direction :** In master mode if “1” selects an I2C master transmit “0” selects an I2C master receive.
3	RW	**Ack:**This bit specifies the value of SDA line during acknowledge cycle(9th SCL pulse).
2	RW	**Rep_start:**This bit indicates that processor wants a repeated START.
1	RW	**Inter_reset:** This bit serves as interrupt acknowledge. Processor must set this bit after serving the interrupt.
0	RW	**Halt:** This bit must be set by processor once design has transmitted 8 bits and acknowledge bit. Because design waits for new command once it is finish with previous command.

Note: Design automatically clears the last two bits of control register, so after every byte transfer processor should set these bits to continue the transfer,within timeout period. Otherwise because of timeout Master will release the bus and will go in idle mode.

Status register, SR

Bit	**Access**	**Description**
7	R	**TIP:**This bit indicates that one byte of data is being transferred. This bit will be set at rising edge of acknowledge cycle. “1” Byte transfer completed. “0” Byte transfer in progress.
6	R	**Addr_match:**This bit will be set when address of the core matches.
5	R	**Bus busy:**This bit indicates the bus is involved in transaction. This will be set at start condition and cleared at stop.
4	R	**Arb_lost:**This bit will go high if master has lost its arbitration.
3	R	**Time_out:** This bit indicates that maximum time for which SCL can be in low state, has elapsed
2	R	**slave rw:** “1” master receiving / Slave transmitting “0” master transmitting / Slave receiving
1	R	**Interrupt_pending:**This bit is set when the interrupt is pending. This will be set after transaction of one byte
0	R	**Ack_state:**This bit will indicate the status of SDA line during ack cycle.

Timeout Register, TO

The time-out register is used to determine the maximum time that SCL is allowed to be LOW before the I^{2}C state machine is reset. When the I^{2}C interface is operating, at every SCL low transition counter will be enabled and reseted at other transition.

1. In the master mode, the time-out feature starts every time the SCL goes LOW. If SCL stays LOW for a time period greater than the time-out value, the Core concludes there is a bus error and generates an interrupt and releases the buses. processor needs to read the status register and reset the core.

Bit	**Access**	**Description**
7-0	w	Time Out value

Address Register, ADDR:

ADDR is not affected by the Core hardware. The contents of this register are irrelevant when Core is in a master mode. In the slave modes, the seven most significant bits must be loaded with the micro-controller's address.

The most significant bit corresponds to the first bit received from the I^{2}C-bus after a start condition. A logic 1 in ADDR corresponds to a HIGH level on the I^{2}C-bus, and a logic 0 corresponds to a LOW level on the bus. The least significant bit is not used but should be programmed with a ‘0’.

Bit	**Access**	**Description**
7:0	w	Address of slave

Data Register,DR:

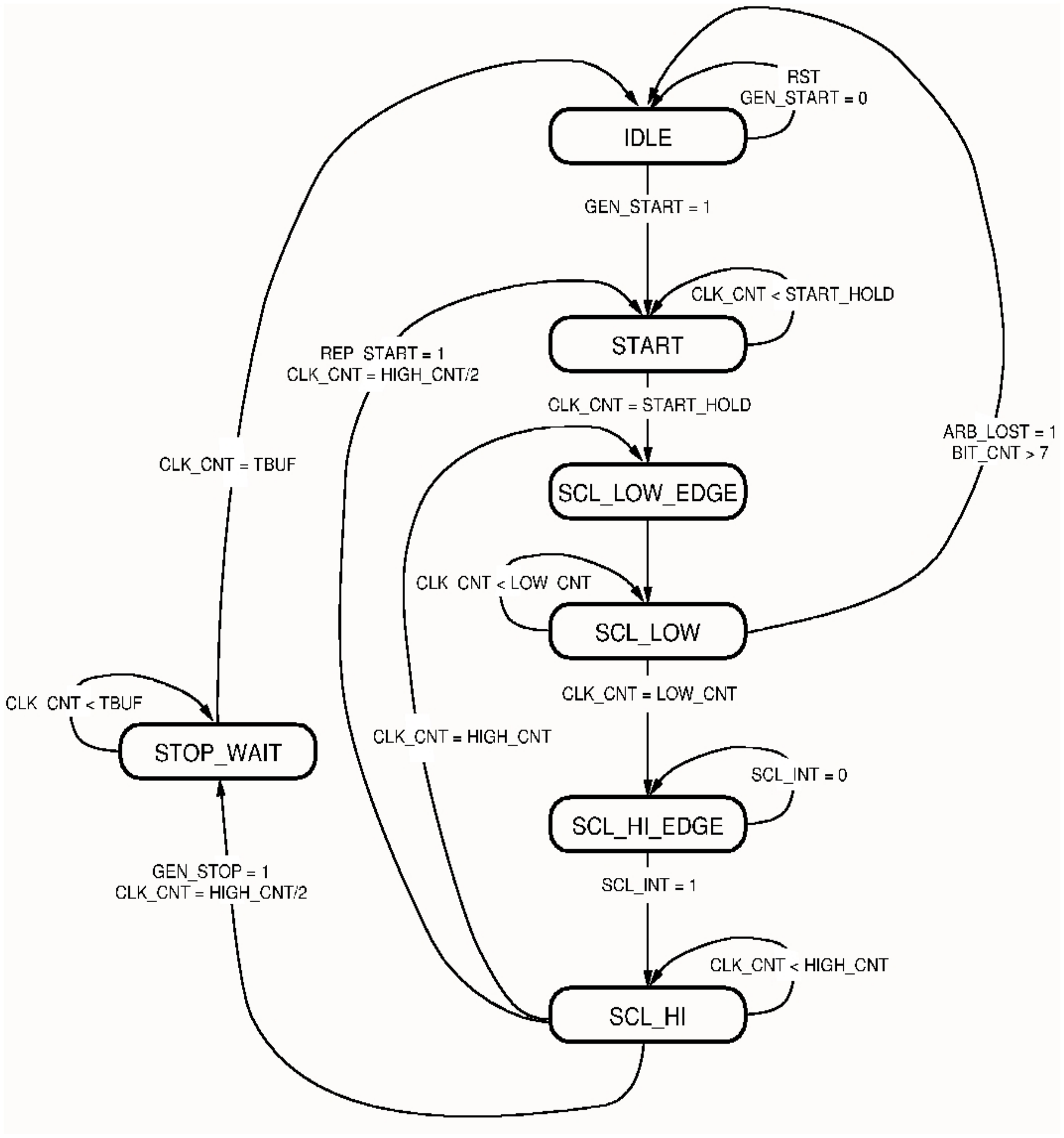
This register contains the data to and from I^{2}C bus. Tis is 8 bit wide. The received data after each complete transfer of byte is placed in this register. Core will not wait whether the placed data has been read or not.

Bit	**Access**	**description**
7-0	RW	I^{2}C data

The address bus used in design will be used by processor to indicate that which register processor wants to access.

State Machine

Scl generator process



This machine will be used only in master mode to generate SCL pulses of desired frequency determined by PRESCALE register.

IDLE: In this state machine will left the buses free and will be waiting for command. If there is a transaction in MODE bit from “0” to “1” core will go to START state,and will act as Master.

START: Here machine will generate the Start condition by pulling down the SDA line low while SCL is high. After start hold time has elapsed machine will go to SCL_LOW_EDGE.

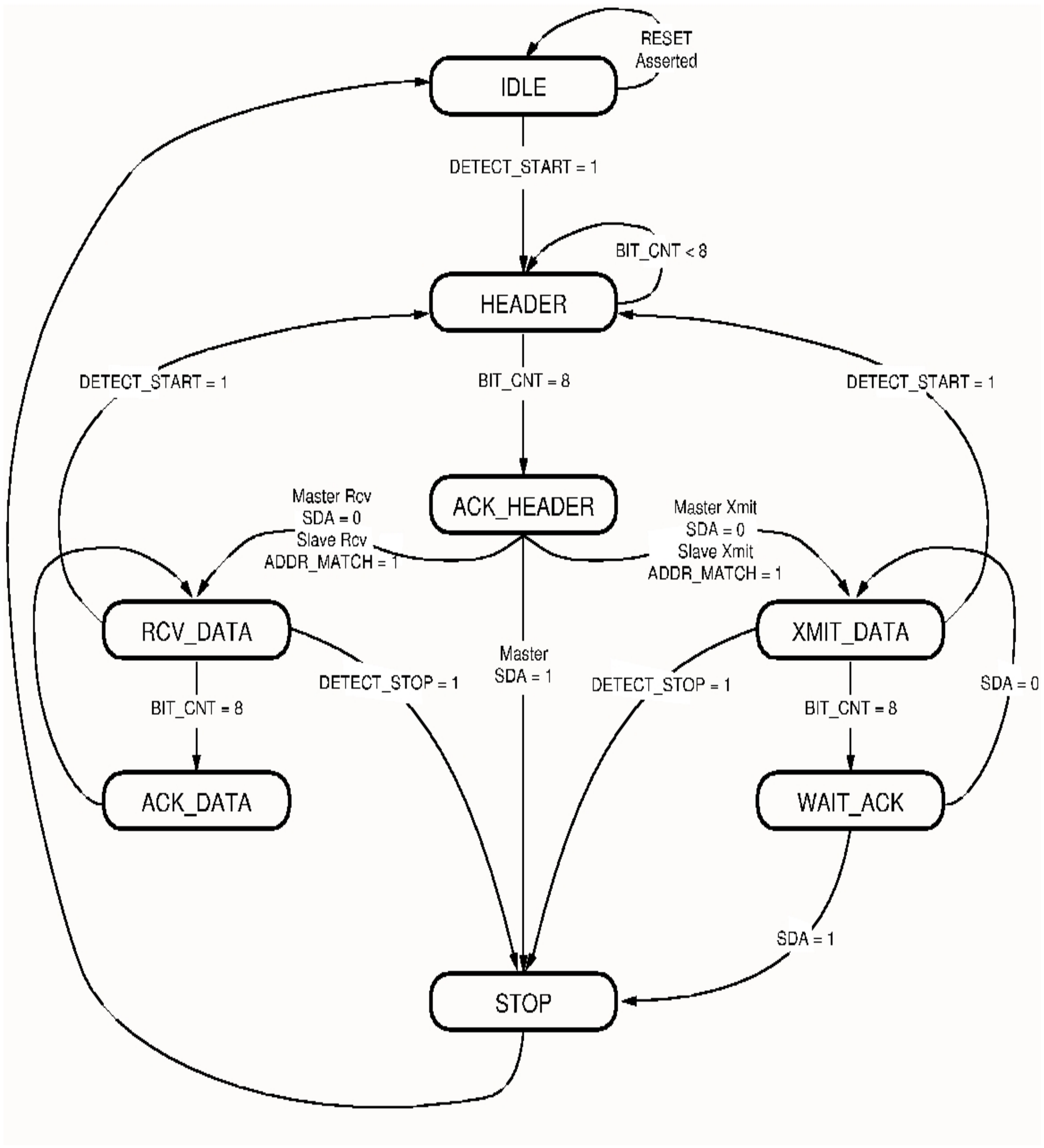
SCL_LOW_EDGE: This state will simply generate a low edge of SCL. In this state clock counters will be reseted. On the next clock pulse machine will move to SCL_LOW state.

SCL_LOW: In this state machine will keep SCL low and counters will be started. If Arbitration lost has been detected then machine will switch to IDLE state. If low time has been reached then machine will go into idle state after updating the status ragister. if byte transfer has completed and arbitration has lost then also machine will go to idle state,to ensure previous byte has been transferred. Otherwise machine will move to SCL_HIGH_EDGE. SDA line will be set in this state according to conditions.

SCL_HIGH_EDGE: This state also will reset the clock counter. In this state a high edge on SCL will be generated. On the next clock pulse machine will move to SCL_HIGH. But it will also check the incoming SCL line o obtain the clock synchronization.

SCL_HIGH: This state will keep SCL high,and clock counter will be started. Depending upon the command from the processor it will move to appropriate state after counter has reached to high hold time. If no specific command is given from processor it will move to SCL_LOW_EDGE. Otherwise it will remain in this state.

Master/slave state machine



Main state machine

Machine will be triggered at falling edge of SCL.

present state	input/condition	Next state
Idle	start_detected	address_shift
address_shift	bit_count=8	address_ack
	stop_detected	Idle
address_ack	arbitration_lost	Idle
	stop_detected	Idle
	acknowledge_receive && master && receive	receive_data
	acknowledge_receive && master && tx	transmit_data
	acknowledge_receive && slave && receive	receive_data
	acknowledge_receive && slave && Tx	transmit_data
	No acknowledge_receive	Idle
Receive_data	Bit_count=8	send_ack
	Stop_detected	Idle
transmit_data	Bit_count=8	wait_ack
	Stop_detected	Idle
send_ack	Stop_detected	Idle
		receive_data
wait_ack	Arbitration_lost	Idle
	Stop_detected	Idle
	Ack_receive	transmit_data
	No_ack_receive	Idle

Note: Start and stop condition will be detected if those has been generated by either the core itself or some other master.

Interrupt will be generated in following conditions:

Finish of core transmission or receive.

Arbitration lost condition

Addressed matched

Stop conditions

Time out condition

Repeated start condition

Operation of core:

The processor will start the communication by asserting address strobe and data strobe. Then in next pulse it will address the internal register and will set the command and data in respective registers. After getting interrupt from core, processor will check the status register and will send the command to take the further action.

Addresses of Registers

Register name	**Address**
prescale register	0000_0010
control register	0000_0100
status register	0000_1000
time out register	0000_1010
address register	0000_1100
data transmit register	0000_1110
Data receive register	0000_0000

Example

Core sequence:

1) Clear the software reset
2) Assert AS and DS
3) Write into prescale register for generating SCL clock frequency
4) Write into Timeout register
5) Change the mode to master mode by writing in control register
6) Write the data register transmitted as the slave address on SDA line
7) Read the status register and send command to master for transmitting and write into data register
8) Generate repeated start and send different address.
9) After the data has been transmitted change the mode to slave by writing in control register
10) Accept the address that being transmitted on SDA line
11)Check for address match flag bit.

Synthesis result

Technology	**Device**	**speed grade**	**Fmax**	**Resource used**
Xilinx	3s50pq208	-4	155.28MHz	LUTs:216