



Macrocad Development Inc. VCM 8010 I2C Synthesizable Model

Description:

The VCM 8010 model from Macrocad is a synthesizable behavior HDL for creating a serial interface controller based on the I²C bus specification. A synchronous parallel interface supports Intel and Motorola interface protocols. Implementations are made easy for both FPGAs and ASICs. Synchronous design and small module size assures worry free synthesis. Hierarchical state machine design is easily modified. Well commented code provides insight into operations. Digital filters accept and synchronize I²C signals. Bi-directional signals are contained in the buffer (shell) level, the core contains unidirectional signals only.

Features:

- ? Synthesizable RTL HDL code
- ? Available in Verilog and VHDL versions
- ? Synchronous design
- ? Digital signal filtering
- ? Well commented code for clarity
- ? Test bench is included
- ? 100KHz and 400KHz supported
- ? 10 bit addressing supported
- ? Master and slave
- ? Designed by hardware engineers for hardware engineers
- ? Approx. 2k gates (asic gates)

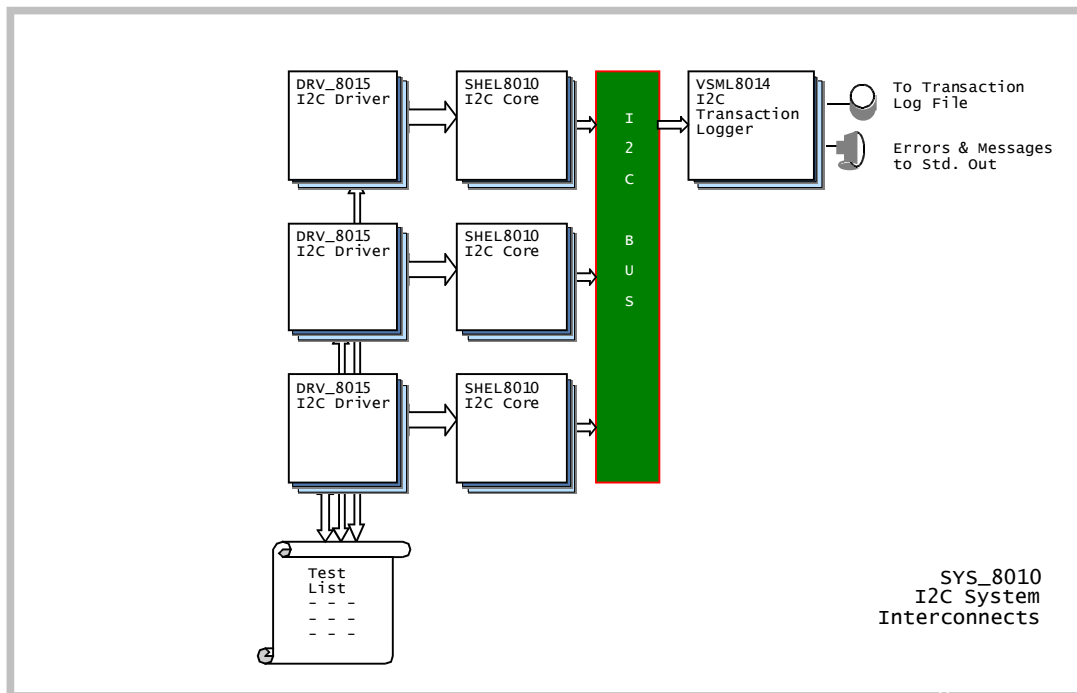


Figure 1

Figure 1 shows the development environment for the I2C bus.



Architecture

The VCM8010 is an interface controller model capable of interfacing the I2C serial bus with an 8 bit parallel programming bus. The top level is an optional bi-directional

shell which contains the only bi-directional and tristate signals in the model. The next level is the core level which contains the 4 functional blocks.

Pin List

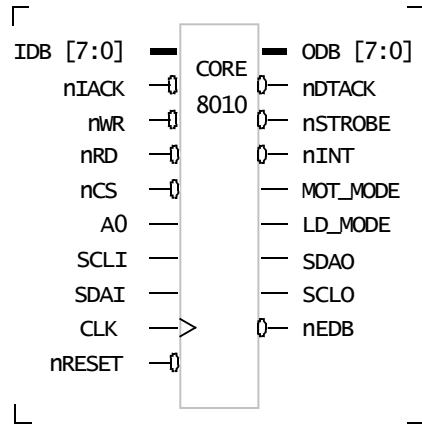


Figure 2

Shell	Core	Type	Description
DB7		Bi-direct	Programming Data bit 7
DB6		Bi-direct	Programming Data bit 6
DB5		Bi-direct	Programming Data bit 5
DB4		Bi-direct	Programming Data bit 4
DB3		Bi-direct	Programming Data bit 3
DB2		Bi-direct	Programming Data bit 2
DB1		Bi-direct	Programming Data bit 1
DB0		Bi-direct	Programming Data bit 0
	ODB [7:0]	Out	Programming Data bus output
	IDB [7:0]	In	Programming Data bus input
	NEDB	Out	read enable for data bus <not>
NRD	nRD	Bi-direct	Programming register read <active low>
	NDTACK	Out	DTACK <mot mode>
	nRD	In	Programming register read <active low>
SCL	SCLO	Bi-direct	I2C clock
	SCLI	Out	I2C clock output
	SCLI	In	I2C clock input
SDA	SDAO	Bi-direct	I2C data
	SDAI	Out	I2C data output
	SDAI	In	I2C data input
NIACK	nIACK	In	Interrupt acknowledge
A0	A0	In	Address 0 <register select>
NCS	nCS	In	Chip select
NWR	nWR	In	Programming register write enable <active low>
CLK	CLK	In	Master clock
NRESET	nRESET	In	Reset
NINT	nINT	Out	Interrupt
NSTROBE	nSTROBE	Out	Strobe
	LD_MODE	Out	Long distance mode
	MOT_MODE	Out	Motorola mode

table 1



Functional Blocks:

The core logic for the VCM8010 is made up of 4 functional blocks:

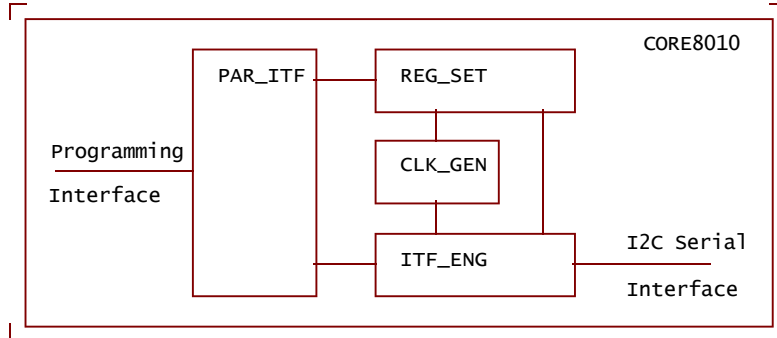


figure 3

PAR_ITF - Parallel Interface

This module interfaces the parallel programming bus to the internal registers and control logic. It includes the register access handshake and interrupt logic.

REG_SET - Register Set

This module contains 4 of the 6 registers, the static registers whose contents can only be modified through the programming interface.

CLK_GEN - Clock Generator

This module generates the intermediate clock for the internal tracking and timing logic. It is nominally 1.5MHz. It can accept 12, 8, 6, 4.33, or 3MHz input clock

frequencies. The clock select register must be programmed properly to generate the nominal 1.5MHz clock from these input frequencies.

ITF_ENG - I2C Interface Engine

This module drives the I2C serial bus. It contains 2 of the 6 registers, the control, status, and shift data registers. It tracks the bus, and takes care of arbitration and error reporting. It also generates the I2C signals if it is participating in a I2C bus transaction. It can monitor I2C bus activity without participating, can act as a master slave or both. It responds to general calls, and can pace the bus when needed.

Register Set:

The register set for the VCM8010 is comprised of 6 registers which are accessible via the programming interface.

Name	Description
S0d	Data shift register
S0a	Own Address register
S1c	Control register
S1s	Status register
S2	Clock register
S3	Interrupt vector register

table 2



Register Mapping:

The access to the register set is accomplished with only 1 address bit. It selects between the data and control register depending on the mode (transfer active or inactive, ES0), and the internal register pointer, (ES1 and ES2). The control

register is always write accessible at address 1. This is necessary since the control register contains the register pointers which are used to access the other registers.

Addr	Pointer						REGISTER DESCRIPTION
A 0	ES0	ES1	ES2	IACK	OP		
- PROGRAM MODE							
1	0	X	X	X	W/R	S1c	CONTROL REGISTER
0	0	0	0	X	W/R	S0a	OWN ADDRESS
0	0	0	1	X	W/R	S3	INTERRUPT VECTOR
0	0	1	0	X	W/R	S2	CLOCK REGISTER
0	0	1	1	X	-		not defined
- TRANSFER MODE							
1	1	0	X	1	W	S1c	CONTROL REGISTER
1	1	0	X	1	Ro	S1s	STATUS REGISTER
0	1	0	0	1	W/R	S0d	DATA REGISTER
0	1	0	1	1	W/R	S3	INTERRUPT VECTOR
X	1	0	X	0	Ro	S3	INTERRUPT VECTOR
- LONG DISTANCE MODE							
1	1	1	X	1	W	S1c	CONTROL REGISTER
1	1	1	X	1	Ro	S1s	STATUS REGISTER
0	1	1	X	1	W/R	S0d	DATA REGISTER

table 3

Control / Status Register Description:

The control register bits are comprised of 4 static bits (ES[2:0] and ENI) and 4 control bits used during I2C transfers, (PIN, STA, STO, and ACK). ES[2:1] are static register pointers, ES[0] enables the controller's participation in I2C transfers, and the ENI enables the parallel interface interrupt. All these bits are typically set prior to participation in I2C transfers, and are thus considered static. They should not change

while the controller is participating in I2C transfers. The PIN bit is the pending interrupt (not) bit. It not only controls the generation of interrupts, but also paces the I2C bus transfers. The STA (start) bit initiates an I2C transfers. The STO bit initiates a stop on the I2C bus. The ACK bit indicates that an ACK bit will be driven during the acknowledge phase of the transfer.

Control / Status word bit assignments:								
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
SET PIN	ES0	ES1	ES2	ENI	STA	STO	ACK	write control word
PIn	RYn	STS	BER	AD0 LRB	AAS	LAB	nBB	read status word

table 4



The following definitions apply to the control and status words:

BIT	NAME	DEFINITION
WRITE CONTROL WORD:		
7	PIN	set the PIN bit
6	ES0	register select bit 0 - enable I2C bus
5	ES1	register select bit 1
4	ES2	register select bit 2
3	ENI	enable interrupt
2	STA	initiate start I2C process
1	STO	initiate stop I2C process
0	ACK	assert ACK on I2C transaction
READ STATUS WORD:		
7	PIn	PIN bit state
6	RyN	not READY - 0 after first busy detected
5	STs	stop detected - slave mode
4	BEr	bus error - erroneous stop or stop detected
3	AD0	addressed as 0 <general call> when AAS = 1
	LRB	OR last received bit when AAS = 0
2	AAS	addressed as slave
1	LAB	lost arbitration bit
0	nBB	not busy bit

table 5



Programming and Transfer Operations:

To understand the controller, a thorough understanding of the I2C bus theory of operations and specifications is essential. The key to the programming and operation of the IIC bus controller is the PIN bit, (Pending Interrupt Not). This bit can be set by writing control word bit 7, and read by reading status bit 7. The start operation requires writing the data to be shifted first, unless it is a restart, or a compound stop /

start operation in which case the data to be shifted is written last. In this case, writing the data word, actually initiates the stop or start operation on the I2C bus. The operational handshake can be accomplished by polling the PIN bit in the status word, and can be established via an interrupt handshake, which is reflective of the PIN bit also.

Programming and serial communication sequences:		
PROGRAM INTERFACE	I2C STATE OPERATION	DESCRIPTION
3 BYTE WRITE:		
write data	idle	setup
write start	start	start I2C transaction
wait - PIN	I2C frame	transfer first byte
wait - PIN	clear PIN	byte done
write data	I2C frame	transfer next byte
wait - PIN	clear PIN	byte done
write data	I2C frame	transfer next byte
wait - PIN	clear PIN	byte done
write stop	stop	transaction done
	idle	wait
2 BYTE READ:		
write data	idle	setup
write start	start	start I2C transaction
wait - PIN	I2C frame	transfer first byte
wait - PIN	clear PIN	byte done
write data	I2C frame	transfer second byte
wait - PIN	clear PIN	byte done
write restart	wait	restart for read
write data	restart	resend first byte
wait - PIN	clear PIN	byte done
read data	I2C frame	read garbage - throw away
wait - PIN	clear PIN	byte done
read data	I2C frame	read first byte
wait - PIN	clear PIN	byte done
write stop	stop	transaction done
read data	idle	read second byte
	idle	wait
Single transaction, write, interrupt enabled		
1	Set up VCM8010 controller.	
2	Program data and transaction procedure.	
3	Enable serial controller.	
4	Serial transaction.	
5	Interrupt generated by controller.	
6	Service interrupt.	
7	End / next transaction.	

table 6



The figure below shows the programming interface access to the register set. The signals in angle brackets are signals internal to the I2C core.

