I2C BUS PROTOCOL DESIGN

1. Overview I2C bus protocol

a. Feature.

I2C BUS PROTOCOL

CLOCK   
DIVIDE

I2C SLAVE

I2C MASTER

START

GENERATE

b. Top module.

TOP MODULE

I2C PROTOCOL BUS

clk\_ex

rst\_n sda\_o

[6:0] address scl

[7:0] writedata

[7:0] readata sda\_i

cs

rd

wr

A diagram of a rectangular object with lines

Description automatically generated with medium confidencec. Block diagram.

d. Inputs / Outputs pin.

- Input :

* clk\_ex : clock of CPU.
* rst\_n : reset negedge of CPU.
* [6:0] address : 7 bit of address (I2C Slave).
* [7:0] writedata : 8 bit of input data.
* cs : chip select posedge.
* rd : read posedge.
* wr : write posedge.
* sda\_i : serial data for read.

- Ouput :

* sda\_0 : serial data for write.
* scl : serial clock.
* [7:0] readata : 8 bit of output data.

e. Register specifiaction.

|  |  |
| --- | --- |
| Address = 7’b0000000 | - writedata [7:0] : data to give information of clock’s external frequency (divide for 4 and transform before give it to writedata) . At this address , internal frequency is 400KHz  - State register will be set 1 to enable internal clock for I2C |
| Address = 7’b0000001 | - writedata [7:0] : data to give information of clock’s external frequency (divide for 4 and transform before give it to writedata) . At this address , internal frequency is 100KHz  - State register will be set 1 to enable internal clock for I2C |
| Address = 7’b0000011 | External clock will be equal internal clock |
| Address = 7’b0000100 - Address = 7’b1111111 | Address for Slaves |

\* I2C limit frequency maximum input is 100MHz.

2. START GENERATE.

a. Inputs / Outputs pin.

- Inputs :

* clk : clock internal.
* clk\_ex : clock external.
* rst\_n : reset negedge from external.
* rd : signal read from bus.
* wr : signal write from bus.
* cs : signal chipselect form bus.
* stretch [9:0] : count to synchronous start from clk\_ex and clk.
* writedatain [7:0] : write data to register internal and latch data for start.
* addressin [6:0] : address data to register internal and latch address for start.

- Outputs :

* start : trigger when cs = 1 and rd = 1, or cs = 1 and wr = 1.
* rw : send signal rd or wr to FSM I2C.
* stop : hold for readata wait response from master.
* writedataout [7:0] : write data latched will be send with clock internal.
* addressout [6:0] : address data latched will be sent with clock internal.

b. Functional.

- Detect when cs = 1 and rd = 1 or cs = 1 and wr = 1 to start process into FSM I2C

to enable process transfer. Especially when cs = 1 and rd = 1 , signal “stop” will

be high until signal “hold” becomes 0 ( readatavalid ) .

- I stipulate wr will drive rw to 0 and read drive rw to 1.

c.Top module.

A green rectangular object with white text

Description automatically generated

Figure 2a. Top module start\_gen.

A diagram of a computer

Description automatically generatedd. Design circuit.

- Number 1 : We have 3 latch D to catch and keep read , write and chipselect

becomes for start latch D to enable collect Data from external and

receive “stretch counter” (from clock\_divide) to count for next

A computer screen shot of a diagram

Description automatically generatedpositive edge of clk internal ( clock from clock\_divide).

- Number 2 : we have start\_ex latch to confirm signal from Number 1 and

stretch\_counter will be counted and decided enable for start\_sync .

A diagram of a circuit

Description automatically generated

- Number 3: Read hold will be decided by “wr” and “rd” , if have wr read hold

will be setted 0 or have rd read hold will be setted 1.

A computer screen shot of a diagram

Description automatically generated

**“rd”**

**“wr”**

- Number 4 : register (latch D) to restore address and data in by start\_ex to prepare

for data for master , start\_sync and stop reg will work with internal clock to

sychonorous of master to write and read data exactly. “stop\_reg” wait for driven

by hold from master, signal “read\_hold” register is stored read\_hold which driven

A computer screen shot of a diagram

Description automatically generatedby two mux block before.

e. Timming chart.

A screenshot of a computer

Description automatically generated

Figure 2b. Timing when have signal cs, wr, rd and hold.

3. CLOCK DIVIDE.

a. Inputs / Outputs pin.

- Inputs :

* clk : clock external.
* rst\_n : reset negedge from external.
* [6:0] address : 7 bit of address.
* [7:0] writedata : 7 bit of datain.

- Outputs :

* Clock\_out : clock for I2C internal.

b. Functional.

- Base on Address and writedata to caculate delay for clock internal to 400KHz or

100KHz mode.

- Using counter to delay the clock input . calculate by count register.

- Frequency external will be transform with ratio 1 unit / 100 KHz.

For example : Clock external is 10 MHz is caculate as 100 unit.

- Writedata of frequency will be divided into 4 before writing to block “clock

divide” . Because writedata just has width is 8 bit so this ratio to ensure suitable

with writedata Bus.

c. Top module.

A green rectangle with black text

Description automatically generated

Figure 3a. Top module clock\_divide.

d. Design circuit.

A diagram of a keyboard

Description automatically generated

Number 1: Decoder simply for “switch case“ to decode 7 bit address , so we have 128 address will be XOR.

A diagram of a computer

Description automatically generated

Number 2 : D Latch will be driven by counter .

A diagram of a computer

Description automatically generated

Number 3 : Counter to count with count register will be caculated from writedata. State ( decode from address , block “or”) decide data for “counter”

A computer screen shot of a diagram

Description automatically generated

Number 4: Compare counter is equal with count , clock\_out put will be selected by mux to 1 or 0. Mux will be driven by signal compare if block count to “counter”.

A diagram of a computer

Description automatically generated

e. Timming chart.

A screen shot of a computer

Description automatically generated

Figure 3a. Example 10MHz to 100KH

A screenshot of a computer

Description automatically generated

Figure 3b. Example 20MHz to 400KH

4. I2C MASTER.

a. Inputs / Outputs pin.

- Inputs :

* ACK\_ADDR : Acknowlege from Slave for address.
* [6:0] address : 7 bit address.
* clk : clock internal from clock\_divide.
* [7:0] idata : 8 bit of data to write.
* rst\_n : rst\_n from system.
* sda\_in : serial data in for read.
* start : start signal from start\_gen.

- Outputs :

* hold : hold signal for start to wait readdata.
* i2c\_scl : serial clock for slave.
* i2c\_sda : serial data for slave.
* m\_stop : signal to give slave stop transfer.
* [7:0] odata : out data for readdata.

b. Functional.

- The I2C master is responsible for initiating and controlling communication on

the I2C bus :

+ i2c\_scl : The master generates the clock signal when have start signal to

enble slave to active. clock synchronizes data transmission

between the master and slave devices. clock frequency is

400KHz or 100KHz.

+ i2c\_sda : The master will send each bit of data in this bus with serial

data . It’s will be 7 steps : start, address sending, read/write

signal , acknowlegde address , data sending, acknowledge data

and stop transfer.

+ m\_stop will be setted 1 when finishing transfer progress.

+ hold will be setted 1 when reading progress is happening.

c. Top module.

A green rectangular object with red lines and red text

Description automatically generated

Figure 4a. Top module I2C Master.

d. Design circuit.

A diagram of a computer network

Description automatically generated

Block 3

Block 2

Block 1

Block 1:

+ store data address and data in in reg\_in and address store to repair for write

address and data write transfer.

Reg\_in :

A computer screen shot of a diagram

Description automatically generated

Address\_store:

A screenshot of a computer

Description automatically generated

Next we have block “add” and “equal” to count up data bits which sent and compare to

count was setted . if done, we go to next step. In progress each bit of address will be

sent first to find out the matching slave . it’s will response to master by ACK\_ADDR.

If it didn’t finish to find out , master will become state “IDLE” which don’t do any

transfer.

A diagram of a computer program

Description automatically generated + Especially we have a state machine to solve all this progress in Master.

A yellow square with red text

Description automatically generated

We have 8 state :

1. IDLE statement :

- At this statement, i2c\_sda will be always setted 1 if we don’t have start signal. Just start

signal will drive to next statement. Code :

2. START statement:

- At this statement, i2c\_sda will be always setted 0 prepare for slave active receive

address . set “count” = 7’d6 and pull down m\_stop to 0 to slave know it can receive and

check address is macthed with its.

3. ADDR .

- Statement ADDR will send each bits of address to slave for storing and checking. It

consumes about 6-7 period to finish transfer 7 bit of address . after finish it will change to

next stage to send signal read or write to drive slave read or write comfortablely.

4. RW.

- Statement RW just give rw signal from start\_gen to serial data line bus and change

statement becomes ACKADRR wait slave check address and send ACK\_ADDR for

master.

5.ACKADDR.

- Statement ACKADDR will accept for being driven by sda\_in (from Slave) and

ACK\_ADDR (from Slave ). This statement will be fully decided by Slave . When Slave

receive address , it will set sda\_in of master is 1 to keep master wait progress . if Slave’s

address is correct , it will response a validaddress to ACK\_ADDR to confirm and let’s

master drive for reading or writing data. Otherwise, it will set sda\_in of master is 0 to let

master know address is not valid for any progress.

- If rw = 1 , count will set 9 (delay 1 period to receive data ) for read and count = 7 for writing.

6. DATA.

- After checking successfully for address slave . Data will be sent to slave for writing or

receive data in and keep hold by 1 for reading .if count = 0 go for next statement.

7. ACKDATA.

- serial data will be send bit 1 to confirm write or read successfully and go to stop.

8. STOP.

- i2c\_sda will be set to 0 to before coming to IDLE to inform transfer is done for slave.

- hold will be set 0 for read progress complete to read .

- m\_stop will set 1 for ensuring slave IDLE ( not working ).

A diagram of a network

Description automatically generated

**ĐK 9**

**ĐK 5**

**ĐK 8**

**ĐK 7.2**

**ĐK 6.1**

**ĐK 7.1**

**ĐK 6.3**

**ĐK 6.2**

**ĐK 4.2**

**ĐK 4.1**

**ĐK3**

**ĐK2**

**ĐK1**

|  |  |
| --- | --- |
| Condition | Detail |
| ĐK1 | start = 0 |
| ĐK 2 | start = 1 |
| ĐK 3 |  |
| ĐK 4.1 | count = 0 |
| ĐK 4.2 | count != 0 |
| ĐK 5 |  |
| ĐK 6.1 | ACK\_ADDR = 1 |
| ĐK 6.2 | sda\_in = 1 |
| ĐK 6.3 | ACK\_ADDR = 0 và sda\_in =0 |
| ĐK 7.1 | count = 0 |
| ĐK 7.2 | count != 0 |
| ĐK 8 |  |
| ĐK 9 |  |

Block 2:

- Receive data for reading progress , it collects data from sda\_in to reg\_out of master . It

also has compare to count for data bits in to ensure collect correctly data at the same

A diagram of a computer

Description automatically generatedtime.

**sda\_in**

Block 3:

- Contains register for hold, i2c\_sda, reg\_out and m\_stop , especially we can see i2c\_scl will be decided by i2c\_enable because if statement is IDLE, START or STOP , i2c\_scl should be sustainable.

A diagram of a computer

Description automatically generated

e. Timming chart.

A screen shot of a computer

Description automatically generated

Figure 4b. signal master when ACKADDR is not successful.

\* Write for clock divide first time will consume at least 13 period after this we can start a progress because now we have clock internal for master’s work, if progress don’t be finish we consumes 13 period to start new progress

A screen shot of a computer

Description automatically generated

Figure 4c. signal master writedata.(100KHz)

\* write progress consume at least 20 period to complete.

A screen shot of a computer

Description automatically generated

Figure 4d. signal master readdata.(100KHz)

\* read progress consume at least 22 period to complete.

5. I2C SLAVE.

a. Inputs / Outputs pin.

- Inputs :

* i2c\_scl : serial clock from master.
* i2c\_sda : serial data from master.
* m\_stop : master stop transfer.
* reset\_n : reset from system

- Outputs :

* sda\_out : serial data out for master read.
* valid\_address : signal high when address correctly.

b. Function

- This slave have suitable roles for interacting with master and store data

write from master.

- suitable with i2c bus .

c. Top module.

A green rectangle with white text

Description automatically generated

Figure 5a. Top module I2C Slave.

A diagram of a computer

Description automatically generatedd. Design circuit.

Block 3

Block 2

Block 1

- Block 1:

Contains register for address data from master and rw bit for deciding

read or write in slave. Address received will be compared with address initial in

slave in block 3.

A diagram of a circuit

Description automatically generated

- Block 2 :

Contains Finite State Machine for Slave and latch D for each bits of data input if

Address is correctly.

A diagram of a computer

Description automatically generated

FSM of Slave has 6 statements:

A yellow box with black text

Description automatically generated

1. IDLE.

- Keep Slave unable , wait for start from m\_stop, i2c\_sda and i2c\_scl from master.

Changes to next stage when i2c\_sda = 0 & i2c\_scl = 1 & m\_stop = 0.

2. ADDR.

- Received 7 bits address and store addr\_buffer which is repaired for next

statement comparation, bit\_count will count to 0 and change to next statement.

3.RW

- Received read or write signal from master on line i2c\_sda and store to rw\_bit

register, if address\_buffer = slave\_address , changes to statement “bool” and set

count = 7 for write or read progress , pull up sda\_out to 1. Unless condition is

right, statement will be setted “IDLE” and sda\_out will be pulled down to 0.

4. bool.

- This statement drives next statements belong to rw\_bit register . if it is 1 , that

means master want to read , others master want to write, statements will change to

DATA\_WRITE or DATA\_READ too.

5. DATA\_WRITE.

- This statement will write 7 bits of data to data\_buffer of Slave . Statements will

be setted “IDLE” when bit\_count is 0.

6. DATA\_READ.

- This statement will read data\_buffer’s 7 bits and send each bits to sda\_out .

Statements will be setted “IDLE” when bit\_count is 0.

A diagram of a diagram

Description automatically generated

**ĐK1**

**ĐK3.2**

**ĐK6.2**

**ĐK7.2**

**ĐK7.1**

**ĐK6.1**

**ĐK5.2**

**ĐK4.2**

**ĐK3.3**

**ĐK5.1**

**ĐK4.1**

**ĐK3.1**

**ĐK2**

|  |  |
| --- | --- |
| Condition | Detail |
| ĐK1 | !i2c\_sda = 0 & !i2c\_scl = 1 & !m\_stop = 0 |
| ĐK 2 | i2c\_sda = 0 & i2c\_scl = 1 & m\_stop = 0 |
| ĐK 3.1 | count = 0 |
| ĐK 3.2 | count != 0 |
| ĐK 4.1 | addr\_buffer = slave\_address |
| ĐK 4.2 | ~ (addr\_buffer = slave\_address) |
| ĐK 5.1 | rw\_bit = 1 |
| ĐK 5.2 | rw\_bit = 0 |
| ĐK 6.1 | count = 0 |
| ĐK 6.2 | count != 0 |
| ĐK 7.1 | count = 0 |
| ĐK 7.2 | count != 0 |

- Block 3:

Contains data buffer and sda\_out latch D to send bit to sda\_out wire, Compare

block to decide valid\_address ( compare with address\_buffer )

A diagram of a computer program

Description automatically generated

e. Timming chart.

A screen shot of a computer

Description automatically generated

Figure 5b. signal Slave when progress is not successful.(wrong address)

A computer screen shot of a black screen

Description automatically generated

Figure 5c. signal Slave writedata. (100KHz)

\* write progress also consume at least 20 period to complete.

A screen shot of a computer

Description automatically generated

Figure 5c. signal Slave readdata. (100KHz).

\* read progress also consume at least 22 period to complete.

6. I2C BUS Timing chart ( Write clock, Write data and Read Data).

- Bus will set to run internal is 400KHz.

- Address of slave is 1010100.

- Data write is 10110011.

- Read data expectation is 10110011.

A screen shot of a computer

Description automatically generated

Readdata at Slave has address is 1010100

Expectation is 10110011

Writedata 10110011 to Slave

Writedata for clock divide to create clock internal