Development of Logic gates with the help of I2C Bus

Pavel Hubenov
Technical University of Gabrovo
5300 Gabrovo, 4 Hadji Dimitar Street, Bulgaria
{pavel_hubenov@yahoo.com}

ABSTRACT: *The communication between two hardware driven instruments is possible when logic gates are introduced through the I2bus. We tried to do it in this paper. When the driver is assembled should be used among peripheral devices captured on this bus. The scope here is high level messages based on designed modules. Avoided is description of low level design which is fundamental for all blocks. Accordingly, the specification of the system, master and slave devices could be different quantity but synchronized. Requirements to the master devices are higher because of its functionality. We have supported the illustration where the driver communicates with EEPROM memory 24C02 and we have given extensive analysis.*

Keywords: Logic Gates, I2C, EEPROM, CPLD

Received: 1 September 2020, Revised 19 November 2020, Accepted 27 November 2020

DOI: 10.6025/jes/2021/11/1/9-14

1. Introduction

Inter-Integrated Circuit (shorted as I2C), pronounced Isquared- C, also called TWI, two wire interface, is a multimaster, multi-slave, single-ended, serial computer bus. Differentiating from other buses this one is for onboard communication.

According to the speed there are several types of the bus from original 100kHz till reach 5MHz ultra-fast mode. Its application is spread outwards of computer boards. Small projects which include just a minor functional processor are equipped with outside located sensors, switches, memories and etc., communicated via this bus. As a requirement of the developed driver is accepted original 100kHz clock frequency because of the wider usage [2].

There are two lines ensured the communication itself – SDA, serial data and SCL, serial clock, Figure 1. First one provides messages which contained strongly defined structured data. The second one guarantees the clock which is responsible for synchronization of the provided data. The driver development is based on strongly recommended structured messages for I2C bus, Figure 2.

Properly communication demands to be realized messages with strongly recommended structured data, Figure 2. According to the example using eeprom memory 24C02 and its specification, there are two different modes – read and write. The Low Significant Bit statement of the DEVICE ADDRESS message, LSB, defines which mode is selected – to write, if it is high level, or to read – low level. Next 3 bits set hardware adjusted address of the device. Most Significant bits “1010” are hard cored default starting statements of the message, Figure 2*a*.

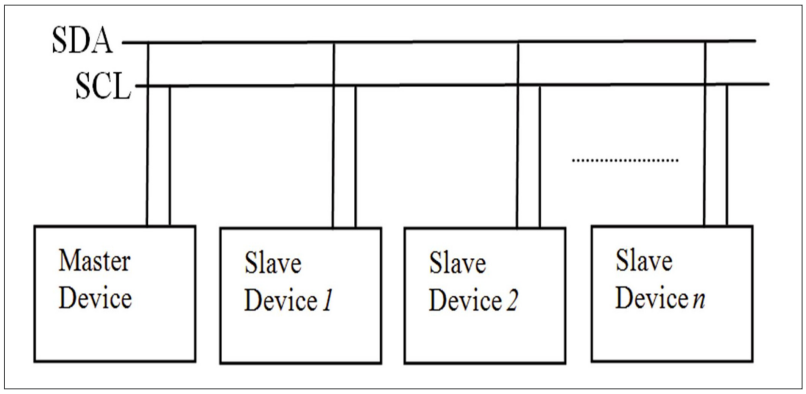

Figure 1. Communication via SDA and SCL lines

When this message is sent on the I2C bus, a slave device, which has same hardware adjusted address on own pinouts, will receive and process the contained data. Then it will response with very short pulse as an acknowledge bit, ACK, which has to be detected by the master device within SCL is low levelled. After this bit is processed by the master device, it sends the second message, containing the address of the desired byte in the memory, Figure 2*b*. An acknowledge bit is followed to enable the master device's last message providing the byte which should be saved, Figure 2*c*.

All three messages have to be synchronized referred to serial clock line, SCL. Basically, every transition occurred on SDA line – raised and falling edge of the messages has to be triggered when SCL line is low levelled.

Other case, when "Read" mode is desired, to read the current written memory cell is necessary to run first message from Figure 2*a* with high level of the low significant bit, LSB. Then after the acknowledge bit, the memory sent a response which master device can process [3].

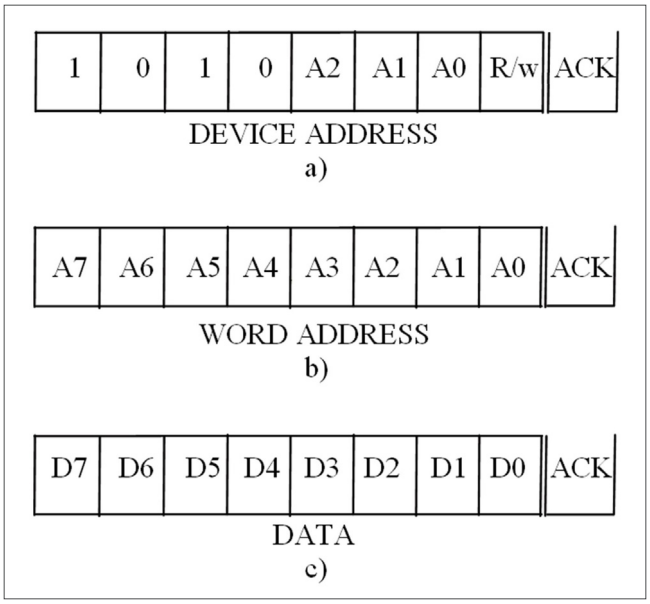

Figure 2. SDA line, message structure

2. Development Model of the Basic Modules

The block schematic is presented in Figure 3. A multiplexer MUX provides all input data to the inputs of a comparator, which routes it to the inputs of the shift register with parallel inputs and serial output, PISO. Control Module enables the shift

register sending and the whole 8 bits word loaded on the multiplexer inputs, is transmitted sequentially on the SDA bus outwards to the memory. Pulses from this bus are delivered via internal feedback bus FB_{In} to Control Module which pass them through shift register with serial inputs and parallel outputs, SIPO, to the Comparator. Different pinout is used, as an input, externally connected outside the integrated circuit to SDA bus, for monitoring the acknowledge signal from the memory, called external feedback FB_{Ex}. If both compared messages by the comparator are different an error occurred. If not, Control Module waits for the low level of the acknowledge signal, via FB_{Ex} and if so, select next channel of the multiplexer and the same execution is following [4].

2.1 Multiplexer

Multiplexer, MUX, module is a tripled 8 bits' multiplexer with adjacent data inputs, address inputs and 8 bits output. The number of the inputs depends on the desired functionality. Presented one in Figure 4 is based on 8 multiplexers with 3 inputs each [5].

The number of the inputs of each multiplexer, N_{MUXin}, is in function of the number of messages, N_{MESS}, to be loaded:

$$N_{MUXin} = N_{MESS} \tag{1}$$

According to the specification of the memory device, there is a mode of sequential reading with more than 3 words. After that is defined the number of the common addresses inputs, N_{MUXadd}:

$$N_{MUXin} = 2^{NMUXadd} \tag{2}$$

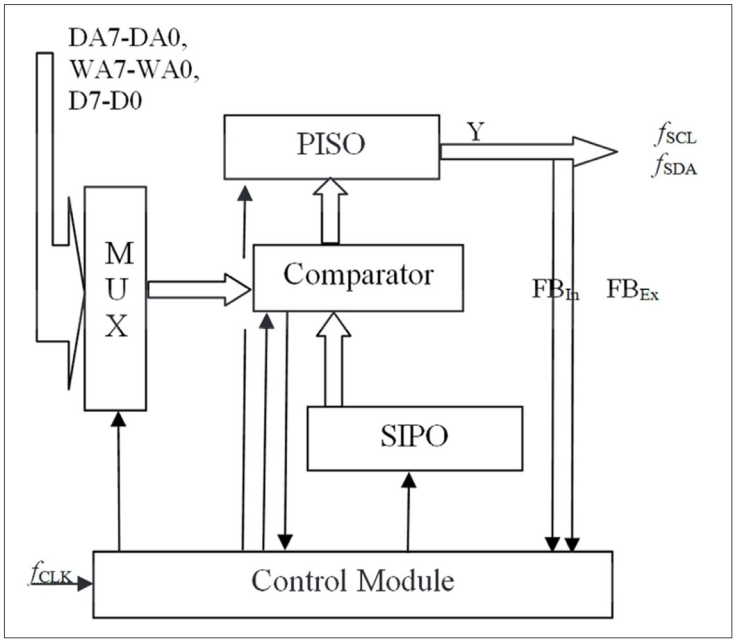

Figure 3. Block schematic of the driver

The number of the used multiplexers, N_{MUX}, depends on the number of the bits, $N_{MESSbits}$, of each message, i.e. its length:

$$N_{MUX} = N_{MESSbits} \tag{3}$$

As per the Figure 2, to load mentioned three 1-byte messages simultaneously, the multiplexer needs 3x1 byte inputs. In this case, first message has to be load to inputs labeled as DA7- DA0(Data Address). Second one, to inputs WD7-WD0(Word Data) and third one to D7-D0(Data). Another approach is using one 1-byte input, and dynamically changed data which transition

should be in the meantime between the last bit of the message and approximately at the same time or around the acknowledge, ACK, received bit. Disadvantage is rescued synchronization and complicated extra module to ensure this delay.

2.2 Comparator

Basically, this module compares both 8 bits messages – input and output one of both shift registers, Figure 5. They have to be absolutely the same transferred via internal feedback, FB_{In} or external feedback FB_{Ex}. Otherwise, there is an internal problem for the environment itself, or external one over the hardware connections, if FB_{Ex} has failed. In both cases, Control Module should stop processing and generate warning on *Y*. It decides which feedback to be used.

To develop this module is necessary to be defined inputs for compared words, N_{COMPin}:

$$N_{COMPin} = N_{MESSbits} \tag{4}$$

This module is peripheral, for verification only, connected to multiplexer – shift register bus.

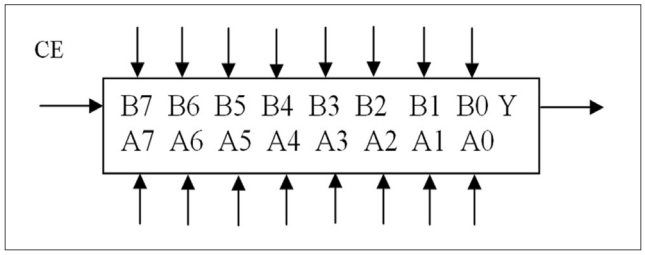

Figure 5. Comparator of two 1-byte messages

2.3 Shift Register with Parallel Inputs and Serial Outputs

Register with parallel inputs and serial output, *PISO*, provides messages in pulse sequence, based on D-latches synthesis, Figure 6. It is a standard register. The input numbers, N_{PISOin}, are as follow:

$$N_{PISOin} = N_{COMPin} \tag{5}$$

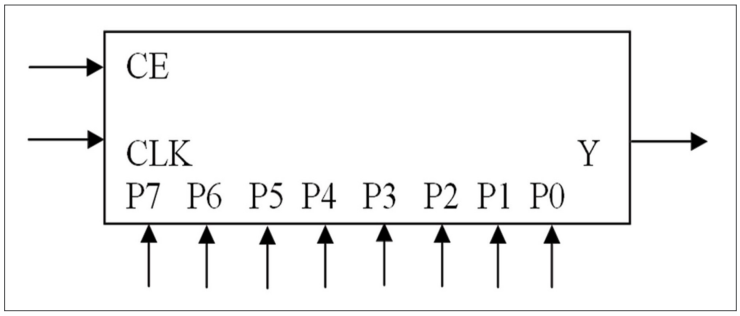

Figure 6. Shift register with parallel inputs and serial output

When is designed this module, should be taken into account whether it has to be able to memorize the message or not.

If it captures all P7 - P0 statements internally, and within message sending, input statements are enabled to be changed, the internal triggers have to be reset after the last bit is sent and before acknowledge bit negative pulse is expired.

2.4 Shift Register with Serial Input and Parallel Outputs

Register with serial input and parallel outputs, SIPO, provides messages from pulse sequence to 1-byte parallel statements, based on D-latches synthesis. Used is a standard register schematic, with serial input and parallel outputs, Figure 7. The

number of outputs, $N_{SIPOout}$, is in function of number of messages length:

$$N_{SIPOout} = N_{COMPin} \tag{6}$$

2.5 Control Module

This module is responsible to process the exact input clock frequency, to provide it reduced to different modules, ensures synchronization among all of them, manage enabling or disabling their functions. Based on the requirement for f_{SCL} = 100 *kHz*, the input counter has to divide fundamental frequency, f_{CLK}, n times, Figure 8, as follow:

$$n = T_{SCL}/T_{CLK} \tag{7}$$

where n is the difference in times. Since SDA and SCL pulses are shifted to 90 degrees, the shifter sub-module has to ensure a delay for accurate transition of SDA exactly when SCL is in the middle of its low level. The delay time m is as follow:

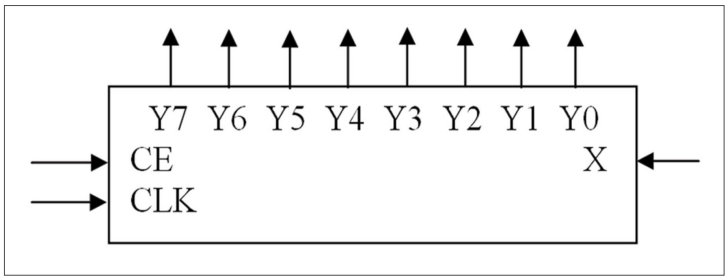

Figure 7. Shift register with parallel inputs and serial output

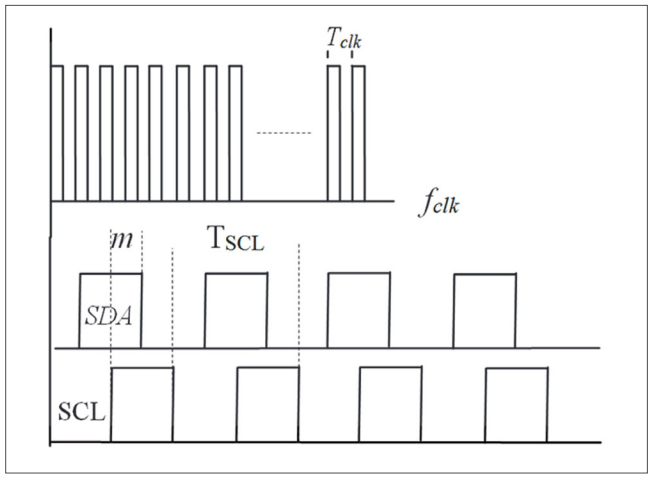

Figure 8. SDA and SCL synchronization

$$m = n/4, \text{ [pulses of } T_{CLK}\text{]} \tag{8}$$

$$T_{0SCL} = T_{0SDA} + m \tag{9}$$

where, T_{0SCL} and T_{0SDA} are starting zero times in the very beginning.

Another feature of Control Module is ACK bit detection. For that purpose, a special sub-module is designed based on a counter which observes the signal via feedback, currently selected one. Reaching the last bit, a counter enables the input of

a latch which has to be triggered by the acknowledge bit. The triggered output runs next operation.

3. Conclusion

The main purpose of the article is to present how is assembled a driver for communication dedicated to I2C bus and realized using logic gates functions. The scope of the designed block schematic is high levelled relationship among the blocks, because it is mostly byte level but a bit level. How they communicate onboard and their behavior with peripheral devices. The strategy is defined according to the complexity of the project. In the current case, algorithm and modules which process it, covered basic communications between the driver and outside located memory. A stress test has been run and upgrade options are arisen.

The developed schematic ensures sending of structured frames according to I2C communications messages. It could be used with different peripherals with minimum changes if necessary.

An advantage to develop a driver like this is the upgrade option. All of the mentioned equations are messages based on, which is fundamental for more complexity communication bus if is necessary. This approach allows to be implemented between two hardware devices on one board, which communicate over upgraded to two bytes' messages bus for example, i.e. acknowledge bit confirms after 2 received bytes.

Customized in this way limits it to be applicable within, between or among, only a group of devices with upgraded modules but not standards. This upgrade is not a benefit which is deserved to be achieved at any cost. Another upgrade option is frequency incrementation. Both should lead to faster and bigger volume of data transfer, but for customized needs. Realized communication on this test example is referenced on f_{SCL} = 100 *kHz*, but easy recalculating to f_{SCL} = 2 *MHz* (this is the maximum allowed frequency as per equations (7), (8) and (9), if f_{CLK} = 8 *MHz*) and testing, provide successfully data transfer result from Comparator Module. In this case is used internal feedb ack to check if bytes are properly transferred and accepted. And for customized needs this method is more successful than the first one [6].

Intentionally low-level design and development of every module is avoided in the description, because using standard units like registers and multiplexers is recommended. Mostly, an extra functionality could be added, but it is referred to Boolean algebra rules and logic synthesis, which could be added as extra schematics to someone of the standards [7].

For realization of the driver is used complex programmable logic device. This environment is based on logic gates functions synthesis and allows low level designing enough.

References

[1] Karbab, E., Djenouri, D., Boulkaboul, S., Bagula, A. (2015). Car Park Management with Networked Wireless Sensors and Active RFID, *Electro Information Technology EIT*, 9.

[2] Kuphaldt, T. (2007). *Lessons in Electric Circuits*, 4th edition, p. 433- 470, 7.

[3] Philips Semiconductors, Application Note I2C Bus, AN10216-01, I2C manual, p. 11-50, 5.

[4] Maini, A. (2007). Digital Electronics - Principles, Devices and Applications, John Wiley & Sons Ltd, p. 69-115, p. 269-298, Chichester, England, 2.

[5] Aleksandrov, A., Goranov, G., Hubenov, P. (2016). Mathematical Model of Control System of Drilling Machine, *ICEST*, Ohrid, 3.

[6] Oberg, J., Hu, W., Irturk, A., Tiwari, M., Sherwood, T., Kastner, R. (2011). Information Flow Isolation in I2C and USB, *Design Automation Conference DAC*, 8.

[7] Lee, S. (2006). Advanced Digital Logic Design, Nelson, 1120 Birchmount Road, Toronto, Ontario, p. 6-148, 6.