# An Strategy and Substantiation of Wish bone I2C Protocol

Mr. G Ahmed Zeeshan<sup>1</sup>, Mrs. Nuzath Unnisa Begum<sup>2</sup>, Ms. Sana Fatima<sup>3</sup>

<sup>1</sup>Associate Professor & Head<sup>,</sup>, <sup>2</sup>Associate Professor, Assistant, Professor<sup>3</sup>

<sup>1,2,3</sup> Department of ECE,

<sup>1,2,3</sup> Global Institute of Engineering & Technology, Moinabad, Rangareddy Dist., Telangana State.

**ABSTRACT:** On-chip synchronised serial connections allow for realistic on-chip communication between the processor, computerised converters, basic and complex converters, memory, and other building blocks on the chip. Serial interfaces are supported by a wide range of Integrated Circuit (IC) manufacturers. This includes the Serial Peripheral Interface (SPI), Inter-Integrated Circuits (I2C), Universal Asynchronous Receive Transmitter (UART), and Universal Serial Bus (USB), among others (USB). SPI is widely used because of its advantages over other serial interfaces, including its ease of use, low power consumption, synchronised clock, and lack of intrusion on the fast data transmission rate. The host controller is an open source PC transport called Wishbone, which enables parallel information trading for faster communication. A master-slave configuration underpins both equipment transports, making the transport interface simpler to manage.

**KEYWORDS**— SPI (serial peripheral interface);

Wishbone; Verilog HDL.

## I. INTRODUCTION

Realistic on-chip communication between the CPU. computerised converters. basic and complicated converters, memory, and other chip building blocks is possible thanks to synchronised serial connections on the chip. Many Integrated manufacturers Circuit (IC)provide serial connections. Serial Peripheral Interface (SPI), Universal Inter-Integrated Circuit (I2C), Asynchronous Receive Transmitter (UART), and Universal Serial Bus (USB) are some examples (USB). Due to SPI's low power consumption, synchronised clock and absence of interference with rapid data transfer rate, it is frequently utilised in applications that need a serial interface. An open-source PC transport called Wishbone allows for parallel information swapping to speed up communication. Both equipment transfers are supported by a master-slave arrangement, making the transport interface easier to administer.

## II. SERIAL PERIPHERAL INTERFACE

Microcontrollers and peripherals may communicate with each other via the Serial Peripheral Interface (SPI) created by Motorola. Others in the industry subsequently adopted this technique. Microprocessor/microcontroller peripheral chips utilise the Serial Peripheral Interface (SPI-bus) to communicate with each other through a simple four-wire serial communications interface. SPI may link two processors even though it was originally designed to communicate between the host CPU and peripherals. Using SPI, you may use either a single-master or multi-master protocol. Most of the SPI Bus's functionality is confined to the PCB. The SPI Bus is a high-speed data transmission interface for IC circuits. According to most studies, the interface can only transmit data in blocks of eight or sixteen bits wide. however Motorola microcontrollers enable transfers in any range between two and sixteen bits wide. It is possible to transport more than sixteen bits of data at a time via control signals because of the serial nature of the interface. An important feature of the SPI protocol is its use of four signal lines (as shown in Fig. 1).



#### Fig 1: SPI block diagram

- MOSI Output data from the master to the inputs of the slaves.
- MISO Output data from a slave to the input of the master.
- SCLK Clock driven by the master to slaves, used to synchronize the data bits.

The master sends a choose signal to each slave, which the slaves then utilise to pick the desired one. Master sets a low SS line for the slaves to whom the data must be transferred if there are several slaves. The SPI Master is responsible for driving the SCLK line and controlling the flow of data. To choose a slave, the Slave Select (SS) line must be low. When using SPI, you can only use a single protocol to communicate with the one master. This means that just one central master initiates all communications with the slaves. After selecting a slave to communicate with, the SPI master pulls the SS line low, which activates the clock recurrence that both master and slave may use to exchange information. Because of this, SPI

is the ideal choice for implementing communication between an integrated circuit like a microcontroller and a group of peripheral devices. There's no doubt that SPI will remain a dominant technology in the future of computerised hardware frameworks.

## WISHBONE INTERFACE

The communication of the SPI master and slave devices with the processor on a chip through the Wishbone bus is shown in Fig. 2.



Fig 2 : Processor-SPI Interaction

Using this SPI WISHBONE controller, a microprocessor with a WISHBONE bus may communicate with an SPI device. SPI Master or SPI Slave: The controller may perform either role. Verilog HDL code parameters are used to specify Master or Slave mode. [8] There is just one module used in the design.

#### A. SPI DATA TRANSFER MODES:

The peripheral devices and the microcontroller or CPU establish a full-duplex connection. Each clock cycle, the data word or character is moved out sequentially one bit at a time between master and slave. Only once a slave peripheral device is chosen by the microcontroller or CPU can the data sampling and shifting be synced by the SCLK. It's possible to set the clock phase and polarity in the SPI control register with only two bits. There are four ways to use this device. The modalities of data transport are represented in the table.3, as shown. There should be no phase or polarity differences between master and slave throughout the transfer. There are two edges of the serial clock, one for shift and the other for capture, since SPI is synchronised with the clock. Ideally, the clock phase dictates just two data transport forms.

## UGC Care Group I Journal Vol-8 Issue-02 2019

TABLE IL APPLICATIONS OF SPI

Device	Application						
Sensors	Temperature, Pressure, Touch Screens, controllers, ADCs						
Control Devices	CODECs, DACs, Digital Potentiometers						
Communications	Ethemet, USB, CAN, USART, IEEE 802.11, IEEE 802.15.4						
Memory	Flash, EEPROM, SD card						
Clocks	Real Time Clocks						

TABLE III	MODES OF OPER ATION
TROLL III.	MODES OF OPERATION

Mode Polarity (CPOL)		Phase (CPHA)	Data Shift	Data Sample
0	0	0	Falling (Negedge)	Rising (Posedge)
1	0	1	Rising (Posedge)	Falling (Negedge)
2	1	0	Rising (Posedge)	Falling (Negedge)
3	1	1	Falling (Negedge)	Rising (Posedge)

## **IV. METHODOLOGY**

The design methodology of the SPI follows the below flow



Fig 3 : Flow chart of the VLSI design

## **V. ARCHITECTURE**

The whole SPI architecture is shown in the diagram below. The RTL design for each of the three blocks has been developed and tested. Then comes the top-level integration and the top-level testing. After that, the integrated architecture's timing analysis is carried out.

# $clk \rightarrow adr$ $rst \rightarrow dout$ sPI Top $din \rightarrow wishbone$ $ack \rightarrow we$ $err \rightarrow sel$ SPI Slave TB

The Micro-Architecture is shown below is classified into three bocks

- ➢ Wishbone
- SPI Master
- Read FIFO
- ➢ Write FIFO
- Control Circuits

Read Address Register climbs over the Write Address Register, at which time the FIFO is referred to as "empty."

As long as the read and write address LSBs match, and there are differences in the additional MSBs, the FIFO is full. There are three primary control registers in the control circuit block, which are the SPCR, the SPDCR, and the SPI Data Register (SPDR).



Fig 5 : SPI top module.

SPI Slave contains Control Circuits, the below figure shows the SPI slave block. It contains the control circuit block in which the control registers are there.



Fig 4 : SPI Slave Mode

# RTL SCHEMATIC AND SIMULATION RESULTS

WISHBONE MASTER RTL SCHEMATIC





								(22)	14.05							
Name	Value	7,200,000 ps	7,200,500 ps	7,201,000;	s 7,20	1,500 ps	7,212,000 ps	R 202,500	ps (7,20							
l <mark>a</mark> iot	1	ΠΠ	ТПТ		Ш	Ш	TTT	ПП								
l <mark>a</mark> i rest	1															
🔓 i jubjąk	1															
dajub jab	1															
l <mark>a</mark> i vojve	1															
📲 i vojadotišti	101111				101111											
📲 ij volgata (31.0	1011100001			10111001			1									
🕌 i nb seBQ	un				011											
la ovo sal	1															
l <mark>e</mark> i jäged	1	UUUU		UUU				տետ								
l <mark>e</mark> ijižojsta	1															
le ojätjsti	1															
🔓 ojižejsta	1															
		K1: 7,202,572.ps														
▶ 🍯 wr_a	ddr[6:0]	011110	0								0111100					
▶ 📷 wr_se	el[3:0]	1111		1111	1111	11	11	1111	1111	1111	1111	1111	1111	1111 11	11	1111 / 11
▶ 📷 wr_d	lata[31:0]	101110	01001						10111	.0010010	100 10 100	10 100 10 10	0101			
🛯 🌡 wr_ir	nc	0														
🐻 r_wri	ite_lock	0														
🐻 start	_request	0	-													
🕨 📷 newo	dev[7:1]	000000	0								000000					
🌆 newr	rx_txn	0												í		
🕨 🍯 newa	adr[6:0]	000000	0								000000					
🔚 r_bu																
	sy	0														
1. last	sy op	0 X														
] 🔓 last i	sy op	0 X														
I last	sy od ANT_SPE 0	0 X							0							
CONSTA	sy od Ant_spe o Only[0:0 o	0 X							0							
CONSTA	SY OD ANT_SPE 0 ONLY[0:0 0 S]5:0] 01	0 X							0 0 010100							
I ast ■ CONSTA ■ CONSTA ■ READ_O ■ READ_O ■ READ_O ■ READ_O ■ READ_O	SY ANT_SPE 0 ONLY[0:0 0 S]5:0] 03 5_PER_TI 00	0 X						0000	0 0 010100 000000111	1101000						
1 → last CONSTA READ_O READ_O CLOCKS CLOCKS MEM_AI	SY ANT_SPE ONLY[0:0 0 S[5:0] 03 S_PER_TI 00 DDR_BF 00	0 X 10100					0	0000	0 0 010100 000000111	1 10 1000	0111					
I last I las	SY ANT_SPE 0 SNLY[0:0 0 S[5:0] 02 S[5:0] 02 S[PER_TI 00 DDR_BT 00 E[2:0] 00	0 X 0100 000000000 0000000000					0	00000	0 0 010100 000000111 000000000 00000000	1 10 1000	0111					
Last     Last     Lost     Lot     Lost     Lost     Lost     Lost     Lost     Lost	SY ANT_SPE 0 2001/(0:0 5)5:0] 03 5,PER_T1 00 5,PER_T1 00 600R_BT 00 612:0] 00 (ADDR_2 00	0 x 00100 0000000000 00 00 00					0	0000	0 0 010100 0000000111 000000000 00000000	1101000						
	SY ANT_SPE 0 XNLY(0:0 0 S]5:0] 0: S_PER_T1 0: DDR_B7 0: S[2:0] 0: (ADDR]2 0: STOP[2:0 0]	0 X 0000000000000000000000000000000000					D	0000	0 0 0 0 000 000 000 000 000 001 001 010	1301000						
	SY ANT_SPF 0 MULY[0:C 0 S]5:0] 01 S[5:0] 01 S[2:0] 01 C[2:0] 01 GTOP[2:C 01 NEV[2:0] 01	0 x 10100 1000000001 1000000000 10 11 10 11					0	0000	0 010100 0000000111 0000 0001 001 010 011	1101000						
	SY ANT_SPF 0 SNL[(0:C 0 S]5:0] 0: S[5:0] 0: S[2:0] 0: (ADDR_B7 0: (ADDR]2: 0: STOP[2:C 0: STOP[2:C 0: STOP[2:C 0: (ATA[2:0] 1:	0 x 10100 1000000000 10 10 10 10 10 10 10							0 010100 000000111 0000 0001 001 011 01	1101000	D111					
	SY ANT_SPE 0 ONLIV[0:0 0 S[5:0] 0 S[5:0] 0 S[2:0] 0 GDR_BF 0 GDDR_BF 0 GDDR_BF 0 GDDR_BF 0 GDDR_BF 0 GDDR_BF 0 GDR_BF 0 GDR_BF 0	0 x 10100 1000000000 1000000000 100000000					0	0000	0 0 010100 000 000 001 010 011 100 101	101000	0111					

**AREA REPORT** 

Slice Logic Utilization:				
Number of Slice Registers:	284	out of	93120	0%
Number of Slice LUTs:	493	out of	46560	1%
Number used as Logic:	427	out of	46560	0%
Number used as Memory:	66	out of	16720	0%
Number used as RAM:	64			
Number used as SRL:	2			
Slice Logic Distribution:				
Number of LUT Flip Flop pairs used:	544			
Number with an unused Flip Flop:	260	out of	544	47%
Number with an unused LUT:	51	out of	544	9%
Number of fully used LUT-FF pairs:	233	out of	544	42%
Number of unique control sets:	19			
IO Utilization:				
Number of IOs:	118			
Number of bonded IOBs:	117	out of	240	48%
IOB Flip Flops/Latches:	1			
Specific Feature Utilization:				
Number of Block RAM/FIFO:	1	out of	156	0%
Number using Block RAM only:	1			
Number of BUFG/BUFGCTRL/BUFHCEs:	1	out of	104	0%

## TIMMING REPORT

Timing Summary:

Speed Grade: -2

Minimum period: 2.605ns (Maximum Frequency: 383.855MHz) Minimum input arrival time before clock: 2.03lns Maximum output required time after clock: 1.133ns Maximum combinational path delay: No path found

Device Utilization Summary (estimated values)										
Logic Utilization	Used	Available	Utilization							
Number of Slice Registers	284	93120		0%						
Number of Slice LUTs	493	46560		1%						
Number of fully used LUT-FF pairs	233	544		42%						
Number of bonded IOBs	117	240		48%						
Number of Block RAM/FIFO	1	156		0%						
Number of BUFG/BUFGCTRL/BUFHCEs	1	104		0%						

## **SLAVE**



<u> </u>													
		PC.	0	11	0	B-		a	e -	12	0	-	¢
	÷		8	2	e	D.		8	0		a.	8	a
			è-	2	۰	10	¢		٩	a	e.		¢
					۵	-0	0	0	<u>n</u> :	0	0	8	a
					18	0	-		100	0	œ		
					-	0		- 10-	0	0	e.		
						0	1300		0		0		
						-	0	0	C.B.	D	0		
						0	8	۵	-0	Test	1		
						5.B		.0			a.		
						6		0					
								9					
							the second second	10					

		1,3	332.787 ns				^
Name Valu	e 1. 1.	332 ns	1,333 ns	1,334 ns	1,335 ns	1,336 ns	1,337 ns
l <mark>a</mark> i_dk 🛛 0	JUU	נוחחחחח					
l <mark>a</mark> i_reset 🛛 1							
🖫 i_wb_cyc 🛛 1							
ີ <mark>a</mark> i_wb_stb 0							
l <mark>a</mark> i_wb_we 🛛 1							
i_wb_addr[5:0] 1001:	11			10011	1		
🕨 📑 i_wb_data[31:0] 1110:	1010010		1	101010010101010101	01111111000010		
▶ <table-of-contents> i_wb_sel[3:0] 1011</table-of-contents>				1011			
ါ o_wb_stall ၀							
l 🔓 i ji2c_sck 🛛 1							
l <mark>a</mark> i_i2c_sda 🛛 0							
▶ 👹 o_dbg[31:0] 1110:	1010010		1	10 10 100 10 10 10 10 10 1	01111111110000		

Page | 22

Copyright @ 2019 Authors

🕨 👹 r_we[3:0]	0000			0000		
🕨 👹 r_data[31:0]	11101010010		1	101010010101010101	01111111000010	
🕨 👹 r_addr[5:0]	100111			10011	1	
▶ 📑 i2c_pipe[3:0]	1010			1010		
last_sck	1					
🚡 last_sda	0					
▶ 📑 i2c_state[2:0	000			000		
▶ 📑 WB READ O	NLO			0		
▶ ₩ 12C READ O	NEO			0		
	ES 1010000			10100	0	
JEAVE ADD	1010000			10100		
NUMBER ADDO	PC coccessore				000000000000000000000000000000000000000	
MEM_ADDR	BF 0000000000		0	000000000000000000000000000000000000000	00000000001000	
Kead_only	BF 000000000000000000000000000000000000		0	000000000000000000000000000000000000000	00000000001000	
Kem_ADDR     MEM_ADDR     Read_ONLY     Kead_ONLY     Kead_ONLY	BF 000000000000000000000000000000000000		0	00000000000000000000000000000000000000	000000000001000	
Ken_ADDR     MEM_ADDR     READ_ONLY     Ken_AD_ONLY     Ken_ADA     READ_ONLY     Ken_ADA     READ_ONLY	BT 00000000000 0:0 000 1 001		0	000000000000000000000000000000000000000	000000000001000	
MEM_ADDR,     MEM_ADDR,     MeM_ADDR,     ReAD_ONLY     Mile I2CIDLE[2:0]     Mile I2CSTART[2:0]     Mile I2CADDR[2:0]	BF 000000000000000000000000000000000000		0	00000000000000000000000000000000000000	00000000001000	
MEM_ADDR,     MEM_ADDR,     Mig READ_ONLY     Mig I2CIDLE[2:0]     Mig I2CSTART[2:1]     Mig I2CADDR[2:0]     Mig I2CADDR[2:0]     Mig I2CSACK[2:0]	BF 00000000000 00 0 000 0 001 0 010 011		0	00000000000000000000000000000000000000	00000000001000	
MEM_ADDR,     MEM_ADDR,     Min READ_ONLY	BF 00000000000 00 0 000 000 001 010 011 100		0	00000000000000000000000000000000000000	00000000001000	
MEM_ADDR,     MEM_ADDR,     Min READ_ONLY     Min READ_ONLY     Min READ_ONLY     Min READ_ONLY     Min READ_ONLY     Min READ_NY     Min READ_NY     Min READ_NY     Min READ_NY     Min READ_NY	BF 0000000000 0 000 0 0 0 0 100 10		0	00000000000000000000000000000000000000	00000000001000	
MEM_ADDR,     MEM_ADDR,     Management     Read_onlui     Salaria     Icolde[2:0]     Management     Icolde[2:0]     Ma	BF 000000000000000000000000000000000000			00000000000000000000000000000000000000	0000000001000	
MEM_ADDR,     MEM_ADDR,     Management     Read_onlui     Salaria     Icolue[2:0]     Management     Icolue[2:0]     Ma	BF 00000000000 000 000 001 011 100 100 110 011 110			00000000000000000000000000000000000000	00000000001000	
MEM_ADDR,     MEM_ADDR,     MEM_ADDR,     MEM_COLLE[2:0]     MEM_I2COLLE[2:0]      MEM_I	BF 000000000000000000000000000000000000			00000000000000000000000000000000000000	000000000000000000000000000000000000000	

## Area report

Device Utilization Summary (estimated values)										
Logic Utilization	Used	Available	Utilization							
Number of Slice Registers	144	93120	0%							
Number of Slice LUTs	204	46560	0%							
Number of fully used LUT-FF pairs	129	219	58%							
Number of bonded IOBs	115	240	47%							
Number of Block RAM/FIFO	1	156	0%							
Number of BUFG/BUFGCTRL/BUFHCEs	1	104	0%							

```
Device utilization summary:
Selected Device : 6vcx75tff484-2
Slice Logic Utilization:
                                                             144 out of 93120
204 out of 46560
137 out of 46560
67 out of 16720
 Number of Slice Registers:
Number of Slice LUTs:
                                                                                                             0%
0%
0%
                                                                                                                  0%
       nber of Slice Registers:
nber of Slice LUTs:
Number used as Logic:
                                                                        67
64
      Number used as Memory:
           Number used as RAM:
Number used as SRL:
                                                                            3
Slice Logic Distribution:

      Number of LUT Flip Flop pairs used:
      219

      Number with an unused Flip Flop:
      75 out of

      Number with an unused LUT:
      15 out of

      Number of fully used LUT-FF pairs:
      129 out of

                                                                                                219
                                                                                                              34%
                                                                                                    219
                                                                                                                   6%
                                                                                                   219
                                                                                                              58%
    Number of unique control sets:
                                                                          12
IO Utilization:

      Number of IOs:
      117

      Number of bonded IOBs:
      115 out of 240 47%
```

Timming report

Timing Sunmary:

Speed Grade: -2

Minimum period: 2.411ns (Maximum Frequency: 414.740MHz) Minimum input arrival time before clock: 1.216ns Maximum output required time after clock: 0.791ns Maximum combinational path delay: 0.350ns

2007-000 - 10

## APPLICATIONS

- Communication protocol
- Networking applications

## **ADVANTAGES**

- Low area
- Efficient
- Better timing delay

## **CONCLUSION**

Our work focuses on the performance analysis of SPI along with the RTL Design work. Design part involves around the specifications and codes written in the Verilog. Serial peripheral interface RTL divided into three blocks i.e. Master, Slave and SPI top module. RTL code written for the complete architecture using the wishbone interface model which open source, work done using XILINX VIVADO Design suite tool which gives simulation and synthesis validation specification. The complete RTL has been designed for the complete architecture and design summary reports has to recorded.

## **FUTURE SCOPE**

The development of verification environment can be extended to the verification other Wishbonecompliant peripherals that support additional communication protocols.

#### REFERENCES

[1] A.K. Oudjida, M.L. Berrandjia, A. Liacha, R. Tiar, K. Tahraoui& Y.N. Alhoumays, "Design and Test of General-Purpose SPI Master/Slave IPs on OPB Bus," 7th International Multi-Conference on Systems, Signals and Devices, 2010. [2] Freescale SemiconductorInc., "MC68HC912D60 manual V 4.0," September 2010.

[3] "Design and Implementation of Serial Peripheral Interface Protocol Using Verilog HDL", International Journal of Engineering Development and Research, Volume 3, Issue 3, pp 2321-9939, 2015

[4] S.Sarns and J. Woehr, "Exploring 12C," Embedded Systems Programming, vol. 4, p. 46, Sept. 1991.

[5] A. K. Oudjida, M. L. Berrandjia, R. Tiar, A. Liacha, K. Tahraoui, "FPGA implementation of 12C & SPI protocols: A comparative study," in Proc. 16th IEEE International Conference on Electronics, Circuits, and Systems, Dec. 2009, pp.507-510.

[6] Konstantin Mikhaylov, J. T. and Fadeev, D. "development of energy efficiency aware applications using commercial low power embedded systems". In Embedded Systems - Theory and Design Methodology, 2012, pages 407–430.

[7] ŽilvinasNakuti, "Embedded Systems Power Consumption Measurement Methods Overview" . MATAVIMAI, 44(2):29– 35, 2013.

[8] SPI WISHBONE Controller, Reference Design RD1044, Lattice Semiconductor, March 2014.

[9] F.Leens, "An Introduction to I2C and SPI Protocols,"IEEE Instrumentation & Measurement Magazine, February 2009, pp. 8-13.

[10] N. Anand, G. Joseph, S. S. Oommen, and R. Dhanabal, " Design and implementation of a high speed serial peripheral interface", International Conference on Advances in Electrical Engineering (ICAEE), IEEE, 2014, pp. 1-3.

[11] K. Aditya, M. Sivakumar, F. Noorbasha, and T. P. Blessington, "Design and functional verification of a spi master slave core using system verilog", International Journal of Soft Computing and Engineering, vol. 2, no. 2, 2012, pp. 558-563.

[12] Manish Kundu, Abhijeet Kumar, "A Review on Low Power SPI Protocol", International Journal of VLSI System Design and Communication Systems, Vol. 02, Issue. 03, 2014, pp. 0193-0195