International Journal on Emerging Technologies **11**(4): 130-139(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Area Optimized TI2C Design for CSI's Camera Control Interface Protocol

***Utsav Kumar Malviya*[1] *and Ayaskanta Swain*[2]**
[1]*Assistant Professor in Department of Electronics and Telecommunication, Government Engineering College, Bilaspur, India.*
[2]*Department of Electronics and Communication, National Institute of Technology, Rourkela, India.*

(Corresponding author: Utsav Kumar Malviya)
(Received 04 May 2020, Revised 10 June 2020, Accepted 22 June 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Camera Serial Interface (CSI) is a protocol of the Mobile industry Protocol Interface (MIPI) standard for High-Speed Serial data communication between the camera device and mobile baseband processor. As per MIPI's CSI standard camera captured pixel data must be transmitted in unidirectional fashion to the baseband processor of mobile phone using CSI's high-speed Lanes. CSI uses the Camera Control Interface (CCI) standard for control commands communication between the camera device and the baseband processor. CCI uses general features of High-speed Interconnect Integrated Circuit (HI2C) protocol, hence CCI is a general-purpose module and requires more area and non-real-time command communication. This work is replacing HI2C with a design of the Tiny Interconnect Integrated Circuits (TI2C) protocol for CCI. TI2C exclusively designs for commands communication between the camera device and baseband processor. TI2C uses only required components of HI2C which for CCI and removes undesirable components, that reduce the area and processes the commands in real-time. TI2C also supports Camera Command Set (CCS) standard register set which was developed for multiple state-of-the-art camera sensors interfacing. For CCS support this work also develops CCS controller and new interfacing signal set named Camera Control Bridge Interface (CCBI) between TI2C with CCS controller. Xilinx's Vivado Design Suite v2018 has been used for RTL entry and Verification of the design. Zybo Z7-20 Zynq-7000 ARM/FPGA SoC Development Board has been used for Validation of the design. Simulation results of the design found correct and from synthesis results of TI2C RTL design found more area and speed optimized then HI2C RTL design.

Keywords: Camera Command, Camera Control Interface, CSI-2, DPRAM, RTL entry, Command registers, TI2C Salve, HI2C

Abbreviations: MIPI, Mobile Industry Protocol Interface; CSI, Camera Serial Interface; CCI, Camera Control Interface; CCS, Camera Command Set; CCBI, Camera Control Bridge Interface; TI2C, Tiny Interconnect Integrated Circuits; HI2C, High-speed Interconnect Integrated Circuit; RTL, Register transfer level; FPGA, Field Programmable Gate array; ASIC, Application-specific integrated circuit; SCL, Serial Clock; SDA, Serial Data; AMBA, Advanced Microprocessor Bus Architecture; AHB, Advance high-Speed Bus; APB; Advance Peripheral Bus; SMIA, Standard Mobile Imaging Architecture; FLT, Filter; DMA, Direct Memory Access; FIFO, First In First Out; PHY, Physical; FSM, Finite State Machine; DPRAM, Dual Port Random Access Memory, VHDL, Very high speed integrated circuit Hardware Description Language; SoC, System on a chip; ARM, Advance RISC Machine.

I. INTRODUCTION

Mobile Industry Protocol Interface (MIPI) standards are different protocols used in the mobile phone as shown in Fig. 1. MIPI uses the Camera Serial Interface (CSI) [1] protocol between baseband processor and camera device, and for interfacing of control signals, CSI uses a camera control interface CCI [2]. CCI does not need to communicate fast as of data (image pixels). The camera device is considered as a Slave and the baseband processor is considered as a Master in CCI. CCI can support up to 128 slaves (Camera sensor registers) with one master [2]. Fig. 2 shows the MIPI's CSI protocol communication scenario.

Fig. 3 is the elaborated layer diagram of CSI signaling [2]. It shows that CSI Transmitter (CSI-TX) has four different layers to convert pixels data of camera device into high-speed N-Lane data signals. The physical layer of CSI-TX transmits N-lane data synchronize with system clock serially and to CSI receiver (CSI-RX) physical layer. At CSI-RX these received N-lane data again get converted into pixels data for baseband processor. Fig. 3 also shows that I2C's SDA and SCL signal lines are used for CCI[2].

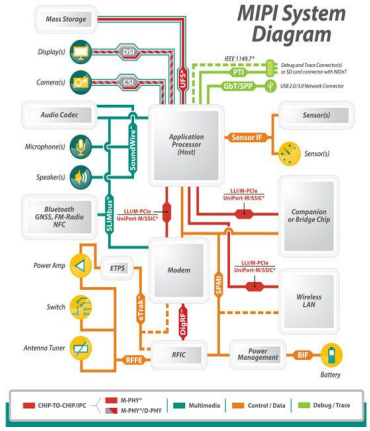

Fig. 1. MIPI Alliance Protocols in mobile phone systems [1].

A new design of Lane merger with Cascade-CRC unit to facilitate parallel processing in the receiver of a multi-lane MIPI CSI was design in 2010 [3]. Later a lane merger with 4 data Lanes and 1 Gb/s per data Lane, i.e. with maximum data rate 4 Gb/s, at 62.5 MHz which increases logic operations from 8 ns (125 MHz) to 16 ns (62.5 MHz) without throughput degradation was designed in 2013 [4].

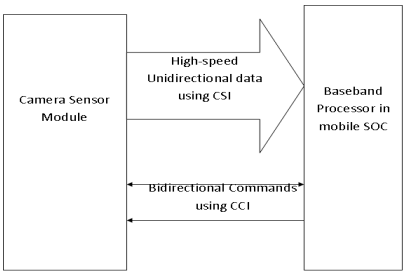

Fig. 2. Camera Data interfacing in MIPI standard.

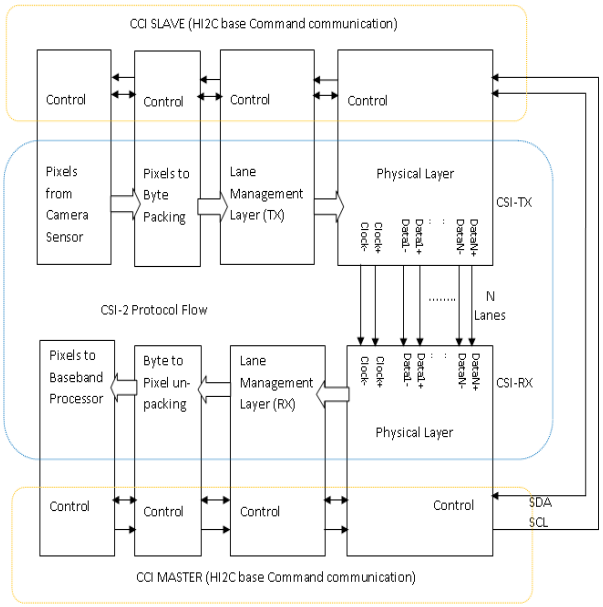

Fig. 3. MIPI CSI-2 Layers and CCI Signaling.

Another advance Lane merger unit which sequentially reorders incoming packet bytes from four data lanes and packs it into a four-byte word to hold a single clock rate regardless of the data lane configurations and to provide scalability from one to four data lanes as defined in 2015 [5]. A design of 2.5-Gbps/lane receiver bridge chip, which fully supports the protocol of the D-PHY version 1.2 for the mobile industry processor interface (MIPI) camera serial interface (CSI)-2 was tested and verified in 2017 [6], Lee's design [6] was verified on 7 series field-programmable gate array (FPGA) and their design convert four-lane high-speed data of scalable low-voltage signaling (SLVS) of the MIPI CSI-2 into 32 low-speed data of low-voltage CMOS (LVCMOS) signaling for a parallel interface with an FPGA chip. Xilinx MIPI CSI-2 Receiver Subsystem v3.0 Vivado Design suit guide defines the state of the art IP design of the CSI-2 protocol in 2018[7]. Xilinx's CSI-2 Defined IP [7] uses 2.5-Gbps/lane receiver bridge with D-PHY version 1.2 for Pixel data transfer on N lanes the same as of [6]. CSI-2 Defined IP [7] uses CCI which is developed on a fast mode variant of the I2C (HI2C) core interface with 2.4 MHz operation and 7-bit slave address.

Practical interfacing of CMOS camera (OV5647) sensor with MIPI CSI on an ARM processor, verification of camera interfacing for different camera resolutions, and formats describes in [8]. Serial T0 Encoding, for camera capture data, reduces the energy dissipation on a serial connection which can be integrated with CSI-2 for energy saving was define in 2016 [9].

High-speed I2C bus interfacing with one I2C master and multiple I2C slaves and different timings and synchronization controls described in 2011 [10] for Embedded sensors like temperature and CO2 sensors. Camera Control Interface (CCI) protocol using Basic I2C (400KHz) for general commands exchange between the camera module and processor was design in 2012 [11]. High-speed new advance I2C UM10204 bus protocol Modes, I2C signals, I2C signal states, frame format, Timing, frequency, current, voltage, and power ratings defined in user manual by NXP semiconductor in 2014 [12]. Modified SLIMbus protocol design for the MIPI alliance presented in 2016 [13]. SLIMbus [13] is a two-wire multimedia communication protocol design compatible with I2C. SLIMbus data always covert into HI2C before it maps on System Advanced Microprocessor Bus Architecture (AMBA) bus [13]. An eye diagram acquisition-based D-PHY protocol designed in 2019 [14] for faster transmission and receiving of the data on lane merger of CSI and DSI, Eye diagram allows fast acquisition of data from camera sensors. Time-domain reflection (TDR) based measurement of low power high-speed CSI lane data and control signal data was used in 2015[15], TDR allows live data measurement of data.

Problem Statement. State-of-the-art mobile phones use multiple camera sensors with advanced features and it becomes an indispensable part of mobile phones. However older camera control SMIA command set[29] is not well suited for State-of-the-art multiple camera sensors interfacing and controlling, hence a new Camera Command Set (CCS) [16] was proposed by MIPI in 2017 for multiple camera support. CCS register is used for specifying camera sensor functionality and control. All IP designs of CSI [4,5,6] before 2017 uses SMIA Command set and now with advance, CCS command set those designs becomes irrelevant and cannot be used in State-of-the-art mobile phones. CSI-2 IP design [7] supports CCS command set hence can be used in State-of-the-art mobile phones. CSI-2 core IP module [7] uses the HI2C module for CCI. CSI-2 IP core designs [3-7] uses of 2.5-Gbps/lane receiver bridge chip, which fully supports the protocol of the D-PHY version 1.2 [2, 14] and transfers Data(pixels) from camera device module to baseband processor unidirectional. However, these designs use a general-purpose HI2C core module for CCI with 2.4 MHz operations, 7-bit slave address and SMIA supports.

This work designs an application-specific TI2C and replaces HI2C for CCI with 2.4 MHz operations, 7-bit slave address, and also supports CCS register set for state-of-the-art mobile phone multiple camera sensor interfacing. Proposed TI2C is developed exclusively for

Camera command exchange between the baseband processor and Camera modules compatible with HI2C but cannot be used as general I2C protocol and it can access complete CCS register set for multiple camera control. TI2C is highly area and speed optimized as compare with HI2C for CCI standard.

II. CAMERA COMMAND REGISTER SET

State-of-art camera devices in mobile phones contain a wide range of different registers define in CCS specification [15] for various control and setup purposes. The CCS supports the following register widths:

- 8-bit – generic setup registers
- 16-bit – parameters like line-length, frame-length and exposure values
- 32-bit – high precision setup values

CCS Registers must be in a byte-oriented space, and the address of multi-byte registers must be point first-byte address.CCS registers are grouped as per their functions and group occupies a pre-allocated area in address space. Total address range is from 0x0000 to 0x3FFF, hence 16-bit index requires in TI2C message. The address space of CCS registers are categorized below:

- Configuration Registers 0 × 0000 to 0 × 0FFF
- Parameter Limit Registers 0 × 1000 to 0 x 1FFF
- Reserved Image Static Registers 0 × 2000 to 0 × 2FFF Manufacturer Specific Register 0 × 3000 to 0 × 3FFF

III. TI2C PROTOCOL

The command data transfer protocol TI2C is similar to the I2C standard. The START, DATA, REPEATED START, and STOP conditions are the same as of I2C. The slave address in the TI2C is 7-bit. The TI2C can support an 8-bit index with 8-bit data or 16-bit index with 8-bit data. The TI2C based camera controller must be able to support four different read operations and two different write operations as follow:

- Single read from a random location.
- Sequential read from a random location.
- Single read from the current location.
- Sequential read from the current location.
- Single write to a random location.
- Sequential write starting from a random location.

The index in the slave device has to be auto-increment after each read/write operation. Fig. 4 (a) and (b) show the one-byte data read/write operation in TI2C for multiple read /write the green shaded operations can keep repeating.

Proposed TI2C can perform atomic read and write operation when 16 or 32-bit control registers of CCS needs to access. When a multi-byte register is accessed, TI2C follows re-timing rules. For a write operation, the updating of the register shall be deferred to a time when the last bit of the last byte has been received and for a Read operation, the value read shall reflect the status of all bytes at the time that the first bit of the first byte has been read. Without re-timing data may be corrupted. TI2C does not Support DMA operation and Multi-master operations because these operations are not required when the camera command needs to exchange between processor and camera device.

Free	Start	Slave Addr.	$\frac{R}{W}$	A	CCS Reg. Addr. (7:0)	A	CCS Reg. Addr (15:8)	A	Data	$\frac{A}{\bar{A}}$	Stop
N-Cycle SDA=1 SCL=1	1 cycle SCL=1 SDA 1 to 0	7 Cycle SDA= Change SCL=0	1 Cycle SCL=0 SDA 1 to 0 for write	1 cycle SCL=0 SDA = 0 (PAck)	8 Cycle SDA= Change SCL=0	1 cycle SCL=0 SDA = 0 (PAck)	8 Cycle SDA= Change SCL=0	1 cycle SCL=0 SDA = 0 (PAck)	8 Cycle SDA= Change SCL=0	1 cycle SCL=0 SDA = 1 (NAck)	1 cycle SCL=1 SDA 0 to 1

(a)

Free	Start	Slave Addr.	$\frac{R}{W}$	A	CCS Reg. Addr. (7:0)	A	CCS Reg. Addr (15:8)	Slave Addr.	$\frac{R}{W}$	A	Data	$\frac{A}{\bar{A}}$	Stop
N-Cycle SDA=1 SCL=1	1 cycle SCL=1 SDA 1 to 0	7 Cycle SDA= Change SCL=0	1 Cycle SCL=0 SDA 1 to 0 for write	1 cycle SCL=0 SDA = 0 (PAck)	8 Cycle SDA= Change SCL=0	1 cycle SCL=0 SDA = 0 (PAck)	8 Cycle SDA= Change SCL=0	7 Cycle SDA= Change SCL=0	1 Cycle SCL=0 SDA 0 to 1 for read	1 cycle SCL=0 SDA = 0 (PAck)	8 Cycle SDA= Change SCL=0	1 cycle SCL=0 SDA = 1 (NAck)	1 cycle SCL=1 SDA 0 to 1

(b)

Fig. 4 (a) TI2C write operation (b) TI2C read operation.

IV. METHODOLOGY

$$\text{Host Processor} \overset{\text{AHB}}{\longleftrightarrow} \text{HI2C Master} \overset{\text{I2C}}{\longleftrightarrow} \text{HI2C Slave} \overset{\text{APB}}{\longleftrightarrow} \text{CCS Memory Controller} \overset{\text{AHB}}{\longleftrightarrow} \text{Camera Device} \quad (A)$$

$$\text{Host Processor} \overset{\text{AHB}}{\longleftrightarrow} \text{HI2C Master} \overset{\text{I2C}}{\longleftrightarrow} \text{TI2C Slave} \overset{\text{CCBI}}{\longleftrightarrow} \text{New CCS Memory Controller} \overset{\text{AHB}}{\longleftrightarrow} \text{Camera Device} \quad (B)$$

Flow (A) above shows the Camera control interfacing in available methods [3-7] and Flow (B) above shows the Camera control interfacing used in proposed work. This work did three major changes (i) Replace HI2C slave with TI2C Slave (ii) Develop new Camera control Bridge Interface (CCBI) signal Set on top of TI2C slave (iii) Design new FSM based CCS controller with two DPRAM memory modules for CCS control registers.

A Modified SLIM bus protocol design for the MIPI alliance presented in 2013 was an application-specific I2C two-wire multimedia communication protocol design, Similarly, the proposed work is an application-specific design of Tiny I2C (TI2C) protocol for command exchange between state of art camera device and mobile phone baseband processor. TI2C eliminates undesired registers of HI2C and also DMA interfacing modules of HI2C. Removal of DMA allows to reduce the size of FIFO in HI2C, these changes make TI2C more area optimized then HI2C.

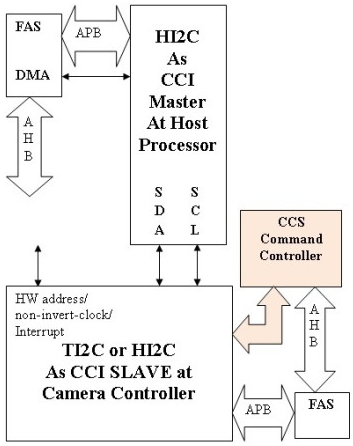

Fig. 5. CCI IP Top module Interfacing.

Fig. 5 shows the state of art Camera device CSI receiver's sub-module CCI core and its signal interfacing at the top-level [3]. CCI Signals are as follow:-

SDA/SCL: Send and collects data with HI2C master (CSI receiver/host processor) on SDA and SCL lines.
DMA: Direct Memory Access from the processor (ARM versatile board), not useful when only camera command needs to exchange.
APB: Advance Peripheral Bus Interface for configuration of TI2C/HI2C slave control or configuration registers.
AHB: Advance High-speed Bus for direct read/write into CCS Registers.
HW address/non-invert-clock/Interrupt: Signals direct from or to the processor, HW address used only when the general call for a processor, in case of TI2C slave, HI2C master should not produce a general call.

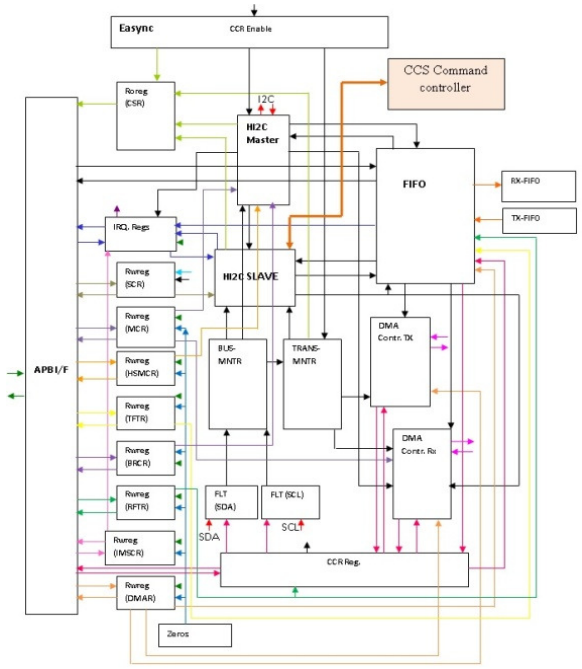

Fig. 6. HI2C internal modules.

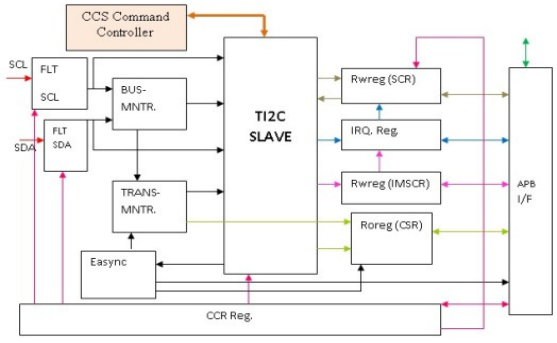

Fig. 7. TI2C internal modules.

HI2C master needed in State of art mobile phone processors, because it has to interface multiple other sensors along with camera device, but the same HI2C module at camera controller is not needed hence a Tiny version of I2C (TI2C) can be used at Slave side. Fig. 6 shows all internal modules and control registers of HI2C RTL. The Registers in HI2C can be con Fig. d by APB to activate only those modules which are required to make it work as CCI slave. This work is a Tiny design of HI2C which kept only required modules and control registers of HI2C. Fig. 7 shows the proposed work TI2C RTL which is a Tiny version design of HI2C for Camera command exchange only. DMA interfacing and related control registers are excluded in TI2C, TI2C does not support multi-master I2C hence no need to keep master module and related control registers. 256 byte FIFO is replaced with an 8-byte temporary register. FPGA AMBA Subsystem (FAS)[2] is used for AHB to APB and vice versa protocol conversion hence no need of bigger size FIFO for commands read-write. An 8-byte temporary register is used for atomic operations.

A. TI2C Design

TI2C design is as shown in Fig. 7 and its internal modules are as below:

(i) FLT: FLT block is for filtering of spikes coming on the scl_in and sda_in, it activates when requested from the APB via CCR.

(ii) BUS-Mntr: This block takes the scl_in and the sda_in signal forms the FLT and generates the status signals. It finds out the start, stops, and busy status of data after monitoring the SDA and SCL lines.

(iii) Trans-Mntr: This block takes the signals from the Bus-mntr and generates required status signals for the Slave physical block (SPHY). It monitors the transmission of the data bytes.

(iv) APB Interface: It has one decoder which decodes the APB address and generates the select signals for different TI2C Registers. When any Register gets selected then data transfer takes place. Transfer direction depends upon the PWRITE signal of APB, if write operation then con figure the TI2C Registers placing PWDATA into them, else if read then read operation TI2C Registers on the PRDATA.

(v) Easync: CCR Register place enable into this block for enabling the Trans-mntr. But Trans-MNTR works on I2C clock and CR on the System clock so this block is used for synchronization of Enable signal.

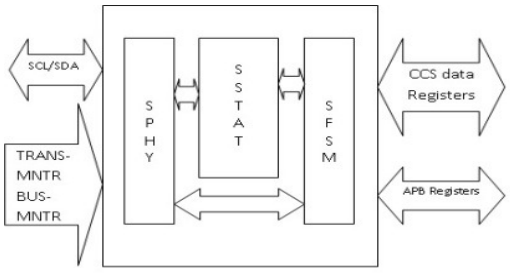

Fig. 8. TI2C Slave control.

(vi) TI2C SLAVE control: Fig. 8 shows TI2C slave control. This block takes the signals from the trans-mntr, Bus-mntr, and in-out I2C line. TI2C slave module also works as a bridge for interfacing CCS controller and HI2C master. This module operates according to control registers which con figure by camera device using APB. It also checks whether the slave address is right or not and if right then It identifies the address and operation mode, in a write operation, it extracts the data from I2C lines and write in on CCS memory, in reading operation it fetches the data from the CCS memory and put the data on SDA line.

— Slave Physical(SPHY): Receive and Transmit data on the SDA lines. In reading operation it starts putting data from SDA-in to the Shift register (serial in parallel out) and in the 9th clock, it put the ack. on the SDA-out line. In reading mode it starts transmitting the data on the SDA-out line, from the Shift register (parallel in and serial out), and on the 9th clock, it read the SDA-in line for acknowledgment. Transmission control signals are provided by trans-mntr and Bus-mntr.

— Slave State(SSTAT): It generates the status signals like reading, writes, EOT (end of transmission), sleep, etc. for SFSM and SPHY and status register.

— Slave Finite State Machine (SFSM): This block generates push and pop signals at every push transfer data into the CCS Register set and at pop fetched data from the Register set, it also isolates the index address from the received data and generates interrupts and status signals. This work con Fig.d HI2C as a TI2C, and two additional control signals added for the SFSM module, these signals are for recognizing index addresses form the incoming data during write oration. New added signals are 'indexls' and 'indexms'.

(vii) TI2C control Registers: Table 2 shows the registers required for configuring the TI2C slave through APB.

Table 1: TI2C Configuration Registers.

Addr	Register	Description
0 × 00	CCR	Control Register: APB loads the configuration information into it. It has FLT controls, General Call control for the SCR Register, Speed control (High speed, fast, and standard), Slave address control (7 bit, 10 bit), and Operating mode controls (Slave mode, master mode (multi-master mode), master/Slave mode (multi-master mode) & Enable. To con Fig. I2C Slave as TI2C Slave we must select Fast mode speed, no general call, 7-bit slave address, and slave mode of operation.
0 × 04	SCR	Slave Control Register: APB loads the Slave data setup time and 7 bit or 10-bit slave address. HW can also load the Slave address when a General call from the APB side. But CCI2C master never makes the general call.
0 × 14	SR	Status Register: Slave logic block write the status information into this register and APB read it on prdata line.
0 × 2C	IMSCR	Interrupt mask set/clear register, APB loads the Mask and Clear for the interrupts.
0 × 30	RISR	Register values read on the PRDATA as raw interrupts.
0 × 34	MISR	Register values read on the PRDATA as mask interrupts.
0 × 38	ICR	APB loads this register, each data bit that is set to '1' causes the corresponding bit in the status registers to be cleared.

TI2C Register values to con figure as CCI slave are mention below:

TI2C CCR Control Register

31-15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	FON		LM	DMA_SLE	DMA_RX_EN	DMA_TX_EN	FRX	FTX	SGCM	SM		SAM	OM		PE

I2C SCR			
31-16	15-10	10-7	6-0
SLSU	Reserve	ESA10	SA7

Set value TI2C_CCR = 0x00000011 to con figure TI2C as CCI slave and Set value I2C_SCR = 0x000A0010 for initial slave address 0x10 and SLSU (slave data setup time) is 0 × A. Refer I2C specification for further details about I2C control registers [17].

B. CCS Controller

This module is developed for interfacing TI2C Slave and CCS memory. The CCS controller defines an additional data protocol layer on top of TI2C slave it has index addressing feature which is not presented in I2C. CCS controller differentiates index address and data coming via I2C lines. An additional set of signals named Camera control Bridge Interface (CCBI) have been developed for interfacing TI2C slave with CCS controller. Table 2 below shows the signals developed for interfacing TI2C and CCS controller.

Table 2: CCBI signals between TI2C SLAVE and CCS controller.

Signal	Type	Function
rx_drdy	IN	Stretching I2C lines during a write operation.
rx_push	OUT	Write data into the DPRAM of CCS registers.
index_ls	OUT	Buffer MSB of index address.
index_ms	OUT	Buffer LSB of index address.
tx_pop	OUT	Read data from the DPRAM of the CCS register.
tx_drdy	IN	Stretching i2c lines during a read operation.
rx_data_in	OUT[7:0]	Data byte coming from CCS controller.
tx_data_out	IN [7:0]	Data byte for CCS controller.

CCS controller contains CCS standard registers (CCS memory) which can be accessed by Camera device via AHB directly when it requires controlling information. CCS memory can be con figure (d) from the baseband processor using I2C lines or Camera devices using AHB. It is completely flexible with the Camera device interface so once it interfaces with the camera module it should stop (stretch) interfacing with the baseband processor. AHB is an independent interface of CCS Registers forms the Camera controller. Fig. 9 mentions CCS Registers RTL Top and its interfaces explained below:

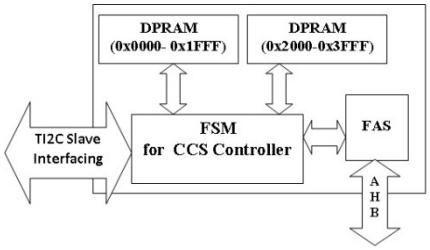

Fig. 9. CCS controller module and interface.

(i) Dual-Port RAM: This work uses DPRAM as CCS memory to store camera control commands. DPRAM is a type of Random Access Memory it can perform two reads or writes at the approximate same time. CCS memory address range is 0 × 0000 to 0 × 3FFF hence, two individual DPRAMs are in use first addresses from 0 × 0000 to 0 × 1FFF and second address from 0 × 2000 to 0 × 3FFF to support memory banking and parallel access of 2 bytes. This DPRAM stores CCS command information from the baseband processor and camera device can read or write into it via AHB.

(ii) CCS Controller FSM: This module controls read/write data in DPARM either from TI2C through CCBI or from the Camera device through AHB. An RTL entry is done for state machine based CCS controller to handles the following task:

- To maintain Data from or to I2C is synchronized with the I2C clock and data from the Camera module is synchronized with the System clock.
- During reading/write with HI2C master if Camera device needs to perform read/write operation with DPRAM then send stretching signals to TI2C slave so it can stop trans-receiving of data from Master.
- Extract index address from incoming data, buffer it and increment index address after every push or pop.
- Write data into DPARM on push at corresponding index address or write data into DPRAM when AHB selected and camera device requires to write operation at corresponding AHB address.

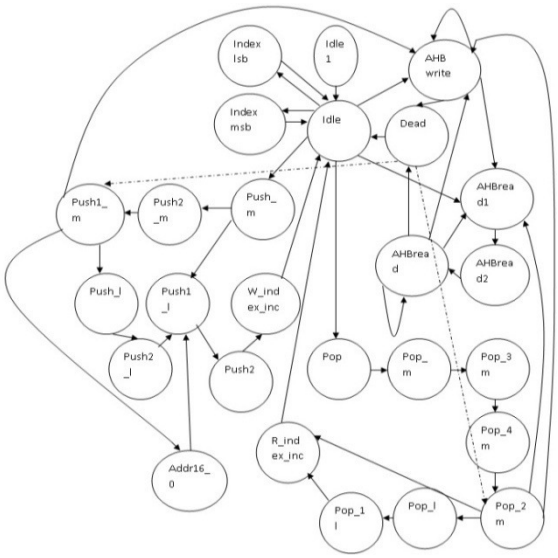

Fig. 10. FSM bridge Control for bridging Proposed signals set TI2C and AHB and DP RAM.

- Read data from the corresponding index address of DPARM on pop or read data from DPRAM when AHB selected and camera device requires to read data of corresponding AHB address.
- Avoid corruption in multi-byte registers during write operation buffer data until the last byte of multi-byte received then write data from the buffer to DPRAM in one atomic cycle after, The Same goes for a read operation.
- Generation of AHB ready-out signal during AHB read operation.
- No, write permission when the index address from the HI2C master is wrong.
- Generation of start pulse when write operation, DPRAM address is 0 × 0010 and write data is 0 × 02.
- Generation of stop pulse when writing operation, DPRAM address is 0 × 0010 and write data is 0 × 01.

Table 3: CCS controller State transition.

Current State	INPUT	Next State
idle1	NA	Idle
Idle	AHB-write &ahb-enb	AHBwrite
	AHB-enb	AHBread1
	Index-ms	Indexmsb
	Index-ls	Indexlsb
	Push	Push_m
	Pop	Pop
	Else	Idle
AHBwrite	not ahb-enb	Dead
	not ahb-write	AHBread1
	Else	AHBwrite
AHBread1	NA	AHBread2
AHBread2	NA	AHBread
AHBread	not ahb-enb	Dead
	AHB-write	AHBwrite
	Else	AHBread
Dead	temp-push	Pushdata1_m
	temp-pop	Popdata2_m
	Else	Idle
Indexlsb	Index-ls	Indexlsb
	Else	Idle
Indexmsb	Index-ms	Indexmsb
	Else	Idle
Push_m	Push	Push_m
	Addr16 & addr[0]='0'	Push2_m
	Else	Push1_l
Push2_m	NA	Push1_m
Push1_m	ahb-write &ahb-enb	AHBwrite
	AHB-enb	AHBread1
	Push	Push_l
	Stop	Addr16_0
	Else	Push1_m
Addr16_0	NA	Push1_l
Push_l	Push	Push_l
	else	Push2_l
Push2_l	NA	Push1_l
Push1_l	NA	Push2
Push2	NA	W_index_inc
W_index_inc	NA	Idle
Pop	NA	Pop_m

Fig. 10 and Table 4 explains CCS controller FSM state transition. The FSM is developed to maintain the synchronization of read/write operation into DPRAM with TI2C and Camera module without any data loss.

V. RESULTS

Xilinx Vivado 2018 used for RTL entry and verification. the simulation has been performed using TLM 2.0 testing environment. Transaction Level Modeling (TLM) [29-30] is a generalized test suit it is a software written in a high-level language that can run along with RTL developed in any HDL and it generates interface on Top level RTL. Using TLM we can force any value with predefined protocol to check our designed protocol for

verification. I2C must support 2.4 MHz speed hence In test RTL, System clock has set at 100 MHz and SCL set at 2.4 MHz Total 76 possible TLM test cases have been verified on simulator few of them explained below:

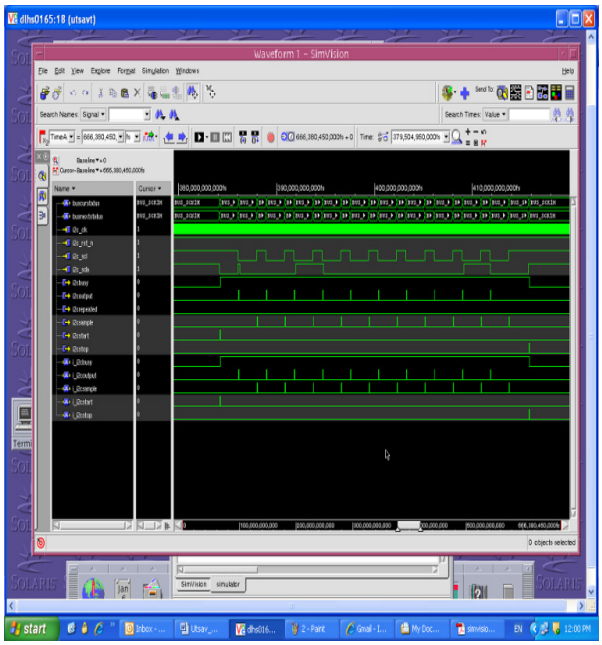

Fig. 11. I2C bus monitoring state simulation.

I2C states like Start, Repeated Start, stop, and Change is correctly identified in at TI2C receiver module shown in Fig. 11 above.

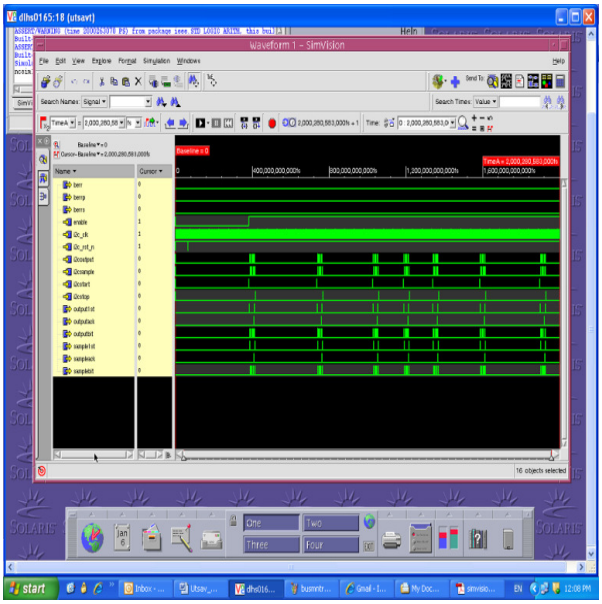

Fig. 12. Transmission control simulation.

Fig. 12 shows transmission control simulation, Data bytes through I2C are properly synchronized and it is correctly responding on the different clock and reset signals.

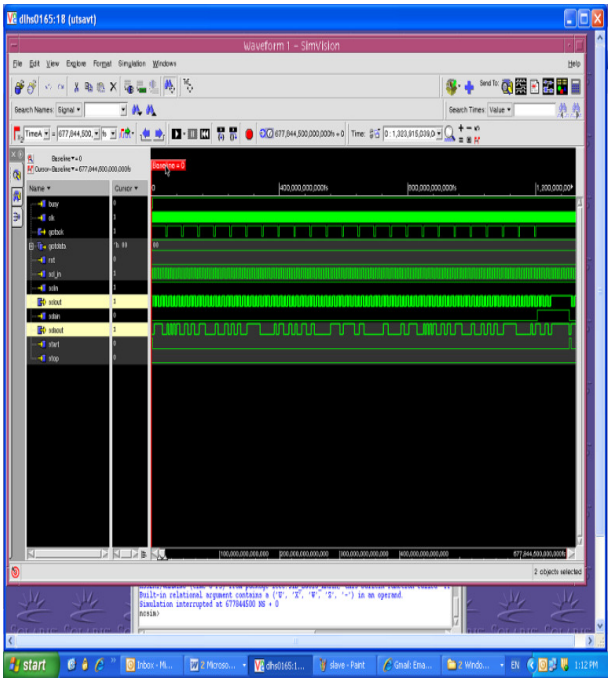

Fig. 13. Single read data from a random location.

In Fig. 13 it may observe that after the correct Slave device addressing and corrects 16-bit address and correct repeated start state, single-byte data is correctly read from CCS memory to I2C lines via TI2C slave.

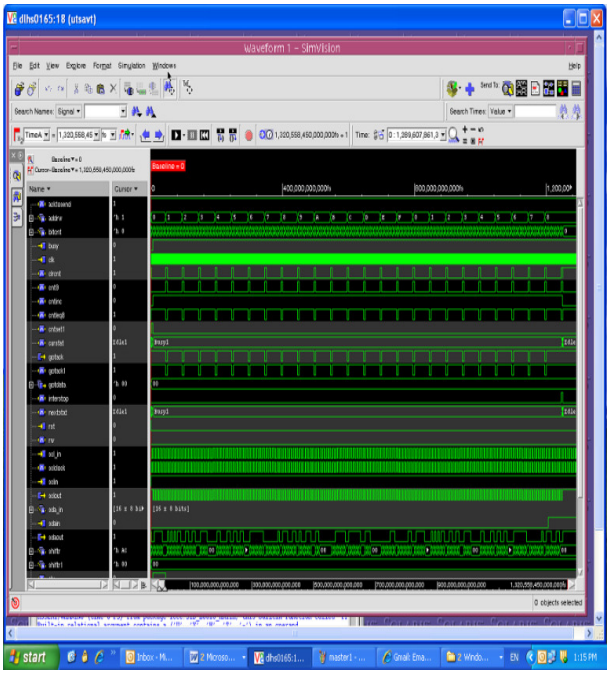

Fig. 14. Repeated data write on TI2C.

In Fig. 14 after correct salve selection and correct addressing two consecutive data bytes correctly write on CCS memory from I2C signals via TI2C slave.

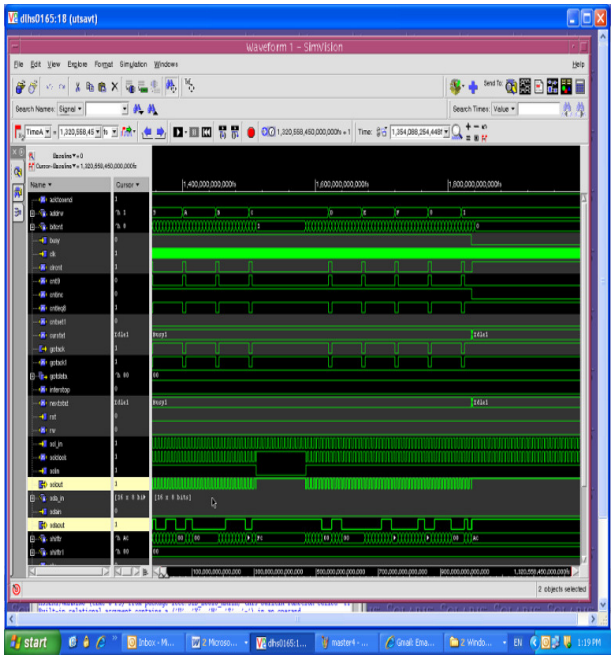

Fig. 15. Negative acknowledgment and data stretching.

From Fig. 15 above it may observe that after the wrong slave addressing TI2C sends a negative acknowledgment and after negative acknowledgment, I2C signals switch to stretching state (SCL=1, SDA=1) till next start condition.

To validate the Design Zybo Z7-20 Zynq-7000 ARM/FPGA SoC Development Board has been used. The Xilinx's I2C core IP is used as CCI Master and has been implemented on one FPGA board and TI2C based CCI slave with CCS controller has been implemented on second FPGA board also CMOS camera (OV5647) have been used for testing. The test data generated using industry-standard (JTAG) protocol to offer start-to-finish testing and validation. Chip-scope pro [31] has been used to verify the results in the emulation environment. Fig. 16 shows the experimental setup for the validation of the design.

Fig. 16. Experimental setup for validation of TI2C and CCS controller.

With Balance design goal and proper floor planning the RTL entry of the design observed fully synthesizable and to evaluate the area utilization of the TI2C slave module it's RTL entry has been synthesized on Xilinx Vivado and Target device selected is Zynq-7000 FPGA with 35nm technology. Table 5 below shows the synthesis summary report for the design of the TI2C Slave. The number of Zynq-7000 slices observe is 160 and 299 LUT's for the TI2C slave module. The maximum speed of RTL design obtain is 156.333 Mhz.

Table 4: Device Utilization summary

HDL Synthesis Report	
Selected Device	Zynq 7000
Number of Slices	160
Number of Slice Flip Flops	133
Number of 4 input LUTs	299
Number of IOs	93
Number of bonded IOBs	90
Number of GCLKs	2
Speed Grade	-5
Minimum period	6.397ns
Maximum Frequency:	156.333MHz
Minimum input arrival time before the clock	6.770ns
Maximum output required time after the clock	7.435ns
Maximum combinational path delay	7.066ns
16-bit subtractor	1
4-bit adder	2
11-bit up counter	1
Registers	122
11-bit comparator great or equal	1
7-bit comparator equal	2
8-bit comparator equal	1
8-bit comparator not equal	1
Power consumption X-power Analyzer	11 mw at 25 degree

The table below shows the optimization obtained in this work in comparison with reference works. This work supports CCS camera command standard, it makes the work superior then [3,4] where SMIA camera command standard was used. Also this work use TI2C for CCI which makes the work more area optimized then [7] where HI2C was used for CCI. HI2C IP core [7] also synthesized on the same platform the number of slices in HI2C found is 279 and 157.21 MHz maximum frequency obtained, clearly observe that TI2C will require less chip area on the same platform.

Table 5: Comparative analysis.

Ref	This work	[3]	[7]
Design	Standard CSI with CCI using TI2C	CSI with multilane and CCI using HI2C	standard CSI and CCI using HI2C
Platform	7th gen. FPGA Semi-custom	CMOS Full custom	7th gen. FPGA Semi-custom
Camera Command Standard	CCS	SMIA	CCS
Maximum Speed	156.33 Mhz	125 Mhz	157.21 Mhz
Numbers of slices	160		279
Minimum time	6.397 ns		11.235 ns

VI. CONCLUSION

This work is an implementation of an ASIC design in which RLT entries are done using a mix modeling style and VHDL programming language. Xilinx-Vivado EDA tool has been used for Synthesis and simulation of the design. The motivation of the work was SLIMbus protocol standard which is a MIPI's application-specific protocol for audio device and it was developed on I2C core. CCI protocol is a part of CSI protocol for command interchange between the camera device and baseband processor in the MIPI standard. This work developed TI2C based CCI in place of HI2C based CCI for reducing the on-chip area in state of art camera devices of mobile phones. Because of TI2C a new FSM based CCS controller has been developed for synchronizing CCS memory access from camera devices through AHB and from baseband processor through I2C, also a new set of signals CCBI have been defined for interfacing TI2C slave and CCS controller. This work has been verified using a dummy ROM based HI2C master and TLM 2.0 test bench. This work also validated the experimental setup of two Zynq-7000 ARM/FPGA SoC Development Boards one master and one slave. The chip area observes for the TI2C module is found less in comparison with the chip area of the inbuilt HI2C IP core module. This work is a semi-custom design near future the work can be developed on the full-custom EDA tool.

ACKNOWLEDGMENTS

This work is a part of the research project assign by the All India Council for Technical Education (AICTE), New Delhi, under the Collaborative Research Scheme(CRS).

REFERENCES

[1]. Allied Vision, (2017). MIPI CSI-2: A new camera interface for developers of embedded machine vision systems. *Stemmer imaging technology forum*. https://www.stemmer-imaging.com/media/*uploads*.

[2]. CSI-2 Specification, (2005). MIPI Alliance Standard for Camera Serial Interface. *MIPI Alliance*, Version 1.00s.

[3]. Lim, K., Kim, G., Kim, S., and Baek, K. H., (2010). A multi-lane MIPI CSI receiver for mobile camera applications. *IEEE Transactions on Consumer Electronics*, *56*(3): 1185–1190.

[4]. Lu, Y.-C., Chen, Z.-Y., and Chang, P.C. (2013). Low power multi-lane MIPI CSI-2 receiver design and hardware implementations. *IEEE International Symposium on Consumer Electronics (ISCE)*, 199–200. DOI: 10.1109/isce.2013.6570183.

[5]. Kaushik, R., and Singh, P. (2015). Study and Implementation of CSI-2 Receiver and Lane Merging. *International Journal Of Advanced Research In Engineering Technology & Sciences*, *2*(3): 56-64.

[6] Lee, P. H., Lee, H. Y., Kim, Y. W., Hong, H. Y., and Jang, Y. C. (2017). A 10-Gbps receiver bridge chip with deserializer for FPGA-based frame grabber supporting MIPI CSI-2. *IEEE Transactions on Consumer Electronics*, *63*(3): 209–215.

[7]. LogiCORE IP Product Guide of Vivado Design Suite, (2018). MIPI CSI-2 Receiver Subsystem. *Xilinx inc.*, v3.0 PG232. www.xilinx.com/support.

[8]. Anusha, P., and Ravinder, K. (2014). CMOS Camera Implementation with CSI based on ARM Processor. *International Journal of Ethics in Engineering and Management Education*, *1*(9): 58-61.

[9]. Pagliari, D. J., Macii, E., and Poncino, M., (2017). Zero-Transition Serial Encoding for Image Sensors. *IEEE Sensors Journal*, *17*(8): 2563–2571.

[10]. Feng, A., Knieser, M., Rizkalla, M., King, B., Salama, P., and Bowen, F., (2012). Embedded system for sensor communication and security. *IET Information Security*, *6*(2), 111–118. DOI: 10.1049/iet-ifs.2010.0073.

[11]. Kushwaha, M., Kapse, V., (2012). A New subset I2C protocol for interfacing Camera module with a baseband processor. *International Journal of Advancements in Research & Technology*, *1*(7):1-8.

[12]. I2C-bus specification and user manual, (2014). *NXP Semiconductors,* UM10204, Rev. 6- 4.

[13]. Pareshbhai, S. R., and Jagtap, S. (2016). Design and verification of serial low-power inter-chip media bus (SLIMbus) device controller supporting the MIPI standard. *IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology*, 932–936. DOI: 10.1109/rteict.2016.7807965

[14]. Chen, B., Penugonda, S., Sun, J., and Fan, J. (2019). Fast Transmitter and Receiver Eye Diagrams Acquisition in the MIPI D-PHY Interface. *IEEE International Symposium on Electromagnetic Compatibility, Signal & Power Integrity (EMC+SIPI)*, 570-574, DOI: 10.1109/ISEMC.2019.8825304.

[15]. McKeone, P., Iepure, B., Agili, S., and Morales, A., (2019). Time Domain Reflectometer Measurements on MIPI D-PHY Protocol for Signal Integrity Analysis. *IEEE International Conference on Consumer Electronics*,1-6, DOI: 10.1109/ICCE.2019.8662018.

[16]. Muukki, M., (2018). Introduction to MIPI Camera Command Set(CCS). *MIPI Alliance*, version 1.0, https://www.mipi.org/sites/default/files.

[17]. Valdez, J., and Becker J., (2015). Application Report on Understanding the I2C Bus. *Texas Instruments Incorporated*, SLVA704.

[18]. Xilinx SoC architecture, (2018). Product Specification of Zynq-7000 SoC Datasheet. *Xilinx inc.,* DS190 v1.11.1.

[19]. Vivado Design Suite User Guide, (2018). *Xilinx inc.*, UG910, v2018.1.

[20]. Bharath, K. B., Kumaraswamy, K. V., and Swamy, R. K., (2016). Design of arbitrated I2C protocol with DO-254 compliance. *International Conference on Emerging Technological Trends (ICETT)*, 332–338. DOI: 10.1109/icett.2016.7873672.

[21]. Lokapure, A. N., Satyanarayana, B., & Raees, P. M. (2016). Interfacing of digital TPH sensors with FPGA using the I2C interface. *IEEE Bombay Section Symposium (IBSS)*, 1021–1025. DOI: 10.1109/ibss.2016.7940205.

[22]. VB6955CM 5.0 megapixel auto-focus camera module Datasheet (2015). *ST Microelectronics*, DocID028544 Rev. 1, https://www.st.com/resource-/en/datasheet .

[23]. Application Note Document Number: AN5305, (2016). MIPI–CSI2 Peripheral on i.MX6 MPUs. *NXP Semiconductors*, Rev. 0, 07/2016, https://www.nxp.com-/docs/en.

[24]. Specification for Camera Command Set (CCS), (2017). Public Release Edition. *MIPI Alliance*, Version 1.0.
[25]. Padmanabhan, T. R., Bala B., and Sundari T., (2000). Design through Verilog HDL. *IEEE Press*
[26]. Bhasker, J., (2008). A VHDL Primer. *Prentice-Hall India*.
[27]. Ghenassia, F., (2005). Transaction-level modeling with SystemC: TLM concepts and applications for embedded systems. *Springer*.
[28]. Moondra, A., (2015). simulation using transaction-level modeling: implementation for ARA modules, Ph.D. Thesis, *School of Vanderbilt University*, Nashville, Tennessee.
[29]. ChipScope Pro Software and Cores, (2012). User Guide-UG029. *Xilinx inc.*, version 14.3.
[30]. Malviya, U. -K., Swain, A., and Kumar, G., (2020). Tiny I2C Protocol for Camera Command Exchange in CSI-2: A Review, *International Conference on Inventive Computation Technologies (ICICT-2020),* 177-182, DOI: 10.1109/ICICT48043.2020.9112536.
[31]. Total MIPI Camera IP Solution Datasheet, (2018). *Arasan Chip Systems*, https://www.arasan.com/wp-content/uploads/2016/05.
[32]. Xilinx Development System *CORE Generator Guide,* (2000). *Xilinx Inc*, version *3.1i.*

How to cite this article: Malviya, U.K. and Swain, A. (2020). Area Optimized TI2C Design for CSI's Camera Control Interface Protocol. *International Journal on Emerging Technologies*, *11*(4): 130–139.