APB Based 16550 Minimal Controller Design

1 Contents

LIST OF FIGURES

LIST OF TABLES

1 Architecture

APB Based 16550 UART Controller is a simple asynchronous serial interface for data transmission and reception. It uses RX and TX lines for data transmission and reception along with APB Completer interface for easy integration with processor. It performs parallel to serial conversion of received data through APB interface and transmits using TX line. Similarly, it performs serial to parallel conversion of data received through RX line and provides it to the processor using APB interface. The block diagram of the APB Based 16550 UART Controller is as shown in Figure 2.

APB Interface block exposes APB interface for interconnection with processor and processes the read/ write transfers and interconnects with register set. Register set holds the configuration information and data which can be easily accessed by the processor through APB interface. Register set implemented is similar to 16550D UART Controller register set, address and format remains same but all the features specified in 16550 Specification are not supported, instead it implements a subset of most commonly used and necessary features. Register set also consists TX and RX FIFOs for storing the data, so that processor can process the data all at once instead of discrete transactions. UART TX is responsible for parallel to serial conversion and transmission of data. UART RX is responsible for serial to parallel conversion of data. Baud rate generator will generate the required clock based on baud rate selected using the reference clock frequency.

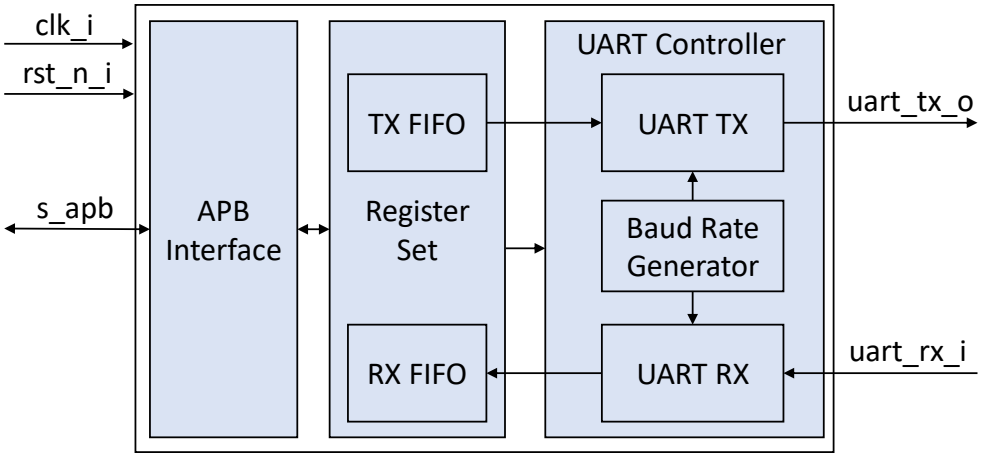

Figure 1:APB Based 16550 UART Controller

Port list of the top-level module is descried in Table 1. Description of each modules of the proposed implementation is included in following sections.

Table 1: APB Based 16550 UART Controller Port List

Port Name	Direction	Width	Description
clk_i	Input	1	Input Reference Clock, coming from the clock source
rst_n_i	Input	1	Active low reset, synchronous to reference clock
UART Serial Interface			
uart_tx_o	Output	1	UART Serial Output Line
uart_rx_i	Input	1	UART Serial Input Line
APB Interface			
s_apb_psel_i	Output	1	APB Select, Initiator selects the particular completer using this line whenever transfer is required
s_apb_penable_i	Input	1	APB Enable, Initiator enables the completer using this line during second and subsequent cycles of transfer
s_apb_paddr_i	Input	8	APB Address, address within initiators address space for which data transfer needs to be initiated
s_apb_pwrite_i	Input	1	APB Write Enable, indicates write transfer when it is high and indicates read transfer when it is low
s_apb_pwdata_i	Input	8	APB Write Data, write data is provided by initiator using this bus during write transfer
s_apb_prdata_o	Output	8	APB Read Data, read data is returned by completer during read transfer using this bus
s_apb_pready_o	Output	1	APB Ready, used by completer to extend the transfer. Until completer provides ready transfer will not be terminated

3.1.Baud Rate Generator

Baud rate generator is responsible to generate clock enable pulses with time period which is equal to 1-bit period i.e., baud rate for UART transmitter. It also generates clock pulses 16 times faster than baud rate for UART receiver. By generating clock enable pulses instead of actual clock the entire system will be operating in single clock domain that is the input reference clock domain.

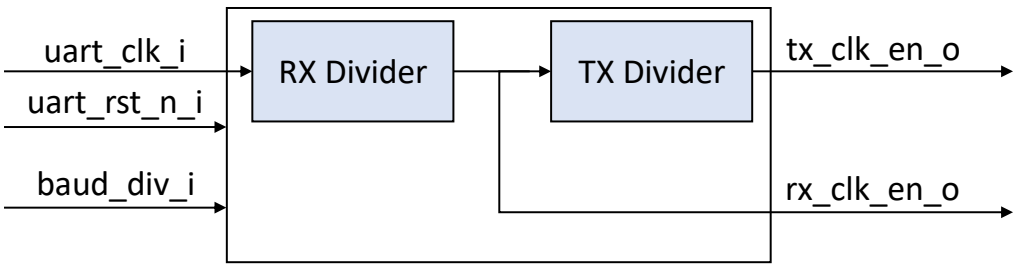

Figure 2: UART Baud Rate Generator Block Diagram

RX Divider is 16-bit counter which runs from 0 to programmed value and produces clock enable pulse 16 times baud rate for UART RX. Tx divider is 4-bit counter which counts from 0 to 15 and generates clock enable pulse at required baud rate for UART TX. The clock enable pulses are used by receiver and transmitter in order to transmit data and sample the received serial data. The clock divisor value calculated needs to be calculated using below formula and is programmed into DLL and DLM registers through APB interface.

$$RX\ CLOCK\ DIVISOR = \frac{\text{Reference Clock Frequency}}{\text{Baud Rate} * 16}$$

3.2.FIFO

FIFO is used to temporarily store data when there is data rate mismatch between the data producer and consumer. Synchronous FIFO is simplest form of FIFO which uses same clock at both read and write side. FIFO mainly consists of write interface which is used to write data into FIFO and similarly it consists of Read interface which is used to read data from FIFO.

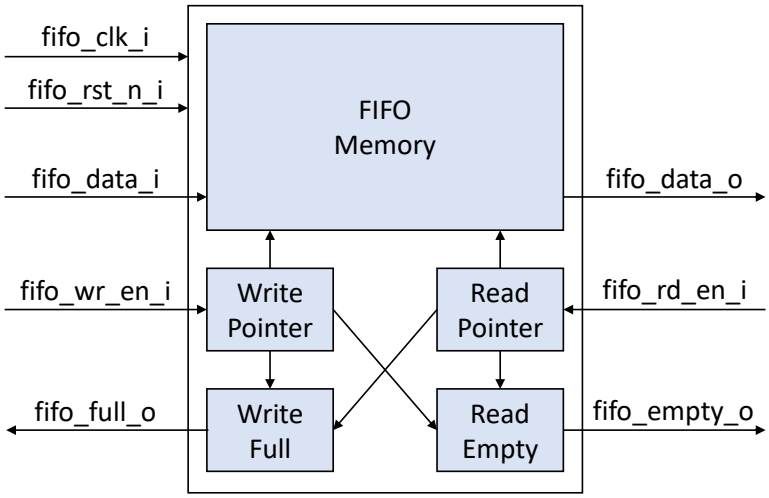

Figure 3: Synchronous FIFO Block Diagram

To perform write operation data along with write enable must be provided, before writing into FIFO it must be made sure that FIFO is not full. Full signal is used to indicate that FIFO is full and cannot accept any more data.

To perform read operation read enable must be provided, before providing read enable it must be made sure that FIFO is not empty. Empty signal is used to indicate that FIFO does not contain any data.

FIFO uses internal write and read pointers to write and read data. Write pointer points the next memory location within FIFO where data needs to be written and gets incremented

when data is written to FIFO. Read pointer points the memory location within FIFO from where data needs to be read and gets incremented when data is read from FIFO.

Read pointer and write pointer are kept 1 extra bit wide than required to easily detect the empty and full condition so that whenever the pointers increment to maximum permittable value rollover takes place and this sets the MSB bit. FIFO full logic compares write pointer and read pointer, except MSB if all other bits of write pointer and read pointer are equal then it indicates that write pointer is rolled over after performing maximum permittable writes. Similarly, when all bits of write pointer are equal to read pointer then it indicates all the data present is read out.

In normal FIFO's read data will be available in next clock cycle after providing read enable. This implementation makes use of Synchronous First Word Fall Through FIFO. In case of first word fall through FIFO read data will be available on read data port before application of read enable. In this case read enable acts like read valid and data on read data port changes when read enable is applied. So FWFT FIFO is also called as show ahead FIFO, which shows the data available within FIFO before application of read enable, this overcomes 1 clock cycle of latency in reading data from FIFO after application of read enable.

TX FIFO

Transmit FIFO stores the data written by the user through APB interface until it is removed by UART TX module and loaded in transmit shift register. Data width of TX FIFO is 8 bit and the depth of the FIFO is 16 words, data is written into TX FIFO by writing THR register.

RX FIFO

Receive FIFO stores the data received by receiver shift register, along with data RX FIFO also stores parity and frame error associated with the received data. Data width of RX FIFO is 10 bits and depth is 16 words.

3.3. UART TX

UART TX Block is responsible to accept parallel data from Transmit Holding Register (TX FIFO) and load it to transmit shift register and shift the data serially through the UART TX line. Data is transmitted out serially in LSB first manner along with start bit, stop bit and optional parity. The configurations can be selected through Line Control Register.

Transmit Shift Register shifts out each data bit serially based on FSM state. Bit counter keeps track of number of bits shifted out. Transmit FIFO read logic is responsible to generate read enable pulse to read data from TX FIFO whenever required. Parity Generator is responsible to generate parity based on selection.

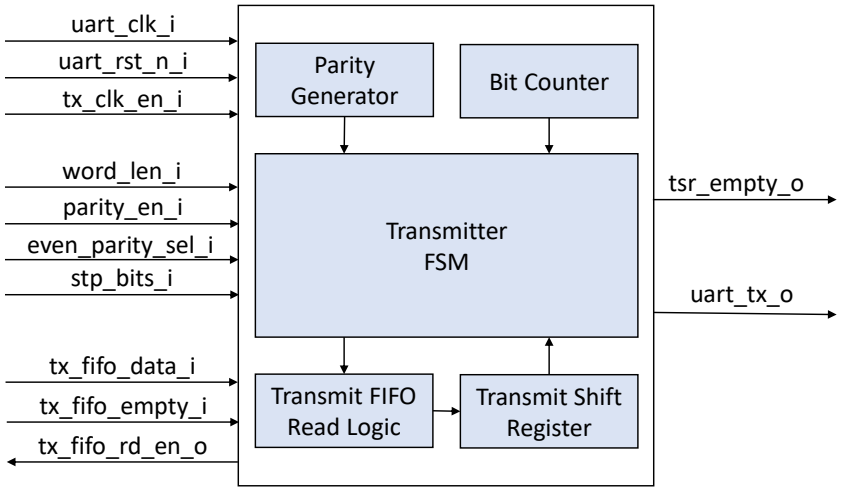

Figure 4: UART Transmitter Block Diagram

UART TX FSM is responsible for complete operation of UART TX, FSM will change its states only on when TX Clock Enable pulse is received.

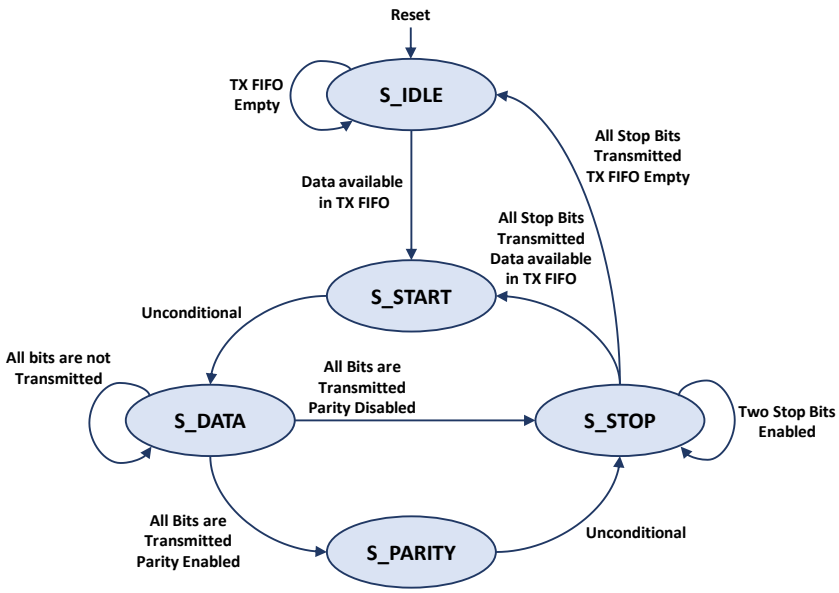

Figure 5: UART Transmitter FSM

S_IDLE: This state is entered upon reset, indicating that transmitter is in Idle condition. UART TX line will be held high in this state and TSR Empty output will be high. Whenever there is data present in TX FIFO FSM transits to S_START.

S_START: Start state is entered either from Idle state or Stop state whenever there is data available in TX FIFO. Start bit is transmitted over UART TX line and read enable signal will be generated to TX FIFO to fetch data to be transmitted. FSM stays in this state until next TX Clock enable pulse is received, on receiving TX Clock enable unconditionally FSM transits to Data State.

S_DATA: This state is entered from Start state after complete start bit is transmitted out. In this state FSM checks the Word Length and shifts out each bit at a time on receiving TX Clock Enable pulse and increments bit count. Once bit count reaches word length FSM transits to Parity state if parity is enabled else it will be moving to Stop state.

S_PARITY: This is entered from Data state if parity is enabled, FSM stays in this state until it receives next TX Clock Enable and unconditionally it moves to Stop state. This state checks whether even or odd parity needs to be transmitted and accordingly sends out the parity bit on UART TX line.

S_STOP: This state is entered either from Data state or Parity state. In this State FSM checks for number of stop bits to be transmitted and accordingly transmits the stop bits over UART TX line. After sending out stop bits it checks whether data is available in TX FIFO, if so, it enters Start state else it moves to Idle state.

3.4.UART RX

UART RX line is synchronized to the reference clock in order to sample it using the RX Clock Enable pulse. UART RX module continuously samples RX line with 16 times oversampling rate. When UART RX detects a high to low transition it detects it as start bit and waits for 8 intervals of oversampling rate if line is still low then it detects it as valid start. Similarly, it samples all incoming data bits, parity if enabled and the stop bit.

Receive shift register shifts in each received data bit and finally on receiving all data bits data is written to RX FIFO through Receive FIFO Write Logic along with the parity error and frame error if the received data was erroneous. Bit counter keeps track of number of received bits similarly tick counter keeps track of number of elapsed RX Clock Enable pulses in order to sample data at centre. Parity Checker checks the incoming parity bit with the calculated parity based on type of parity enabled. FSM is responsible to keep all the activities in synchronization within UART RX Block and is also responsible to detect and generate frame error.

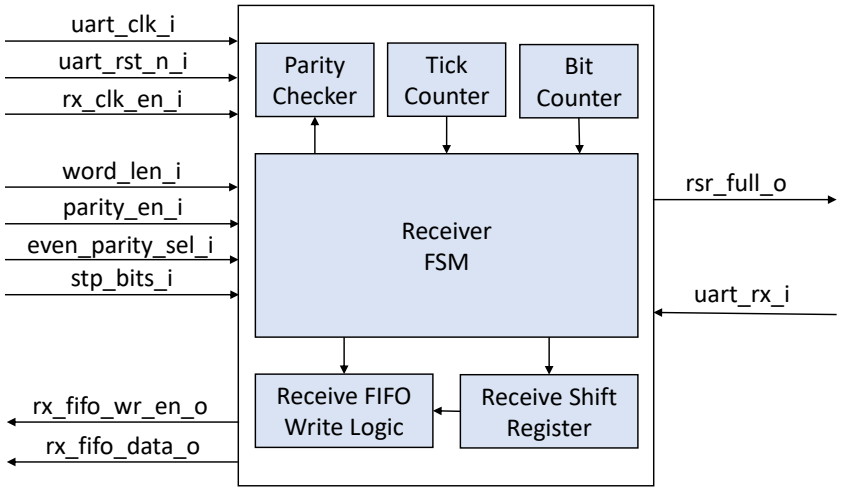

Figure 6: UART Receiver Block Diagram

UART RX FSM will be triggered only on reception of RX Clock enable pulse and is responsible to keep the operation of UART Receiver in synchronization.

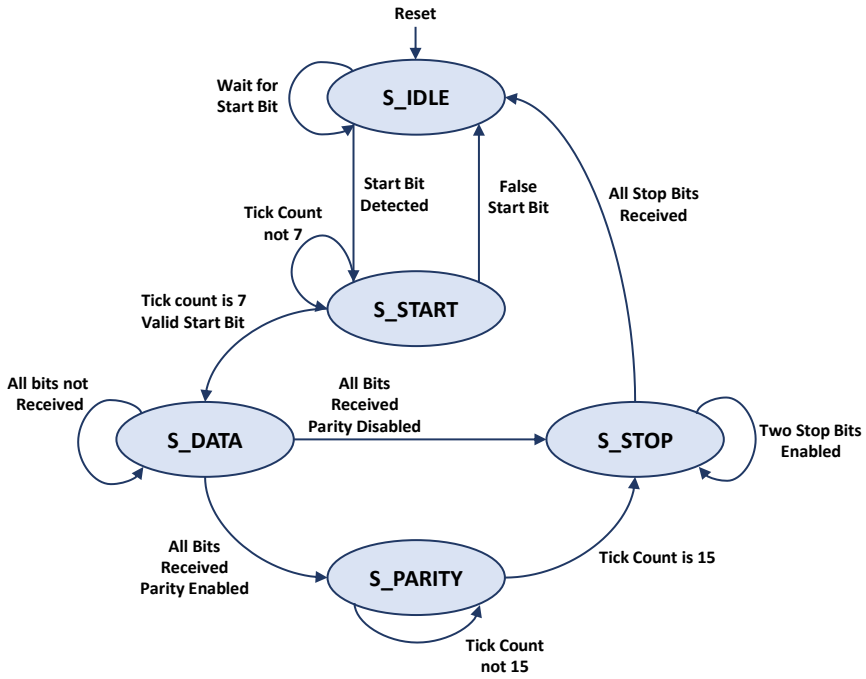

Figure 7: UART Receiver FSM

S_IDLE: Idle state is entered upon reset or after completion of receiving complete data from Stop state. This state indicates that receiver is not performing any operation and is in Idle condition. Whenever High to low transition is detected on UART RX line state is changed to Start state.

S_START: Start State is entered from Idle state when FSM detects start bit, in this state Tick counter starts incrementing on receiving RX Clock Enable pulse and once Tick Count reaches 7 UART RX line is sampled at centre of baud interval to verify whether valid start bit is

received. If valid start bit is received then FSM transits to Data state by clearing Tick counter else false start condition is detected and FSM moves back to Idle state.

S_DATA: Data state is entered from Start state on receiving valid start bit Tick counter keeps on incrementing on receiving RX Clock Enable pulse. Once Tick count reaches 15 incoming data on UART RX line is sampled and shifted in to Receiver Shift Register and Bit Counter is incremented. This state compares word length with the bit count to check whether all data bits are received, on reception of all the data bits if parity is enabled FSM moves to Parity state, else FSM moves to Stop state.

S_PARITY: Parity state is entered from Data state if parity is enabled, Tick counter increments until it reaches 15 and the incoming parity bit is selected. Received parity bit and forwards it to parity checker. Once Tick count reaches 15 FSM moves to Stop state.

S_STOP: Stop state can be entered either from Data state or Parity State. Tick counter increments until it reaches 15 and samples stop bit, if stop bit has gone low in between then frame error is detected. In Stop state number of stop bits enabled will be checked and on receiving all stop bits FSM moves to IDLE state.

3.5. APB Interface

The APB protocol is a low-cost interface, optimized for minimal power consumption and reduced interface complexity. UART controller acts as APB completer which services read and write requests received from APB requestor. Every transfer takes at least two cycles to complete. The APB interface is designed for accessing the programmable control registers of UART Controller through simple read and write interface.

Write Transfer

1. The Setup phase of the write transfer occurs at T1. The select signal, PSEL, is asserted, which means that PADDR and PWDATA must be valid. PWRITE should be held high.
2. The Access phase of the write transfer is shown at T2. where PENABLE is asserted.
3. PREADY is asserted by the Completer at the rising edge of PCLK to indicate that the write data will be accepted. PADDR, PWDATA, and any other control signals, must be stable until the transfer completes.
4. During an Access phase, when PENABLE is HIGH, the Completer extends the transfer by driving PREADY LOW.

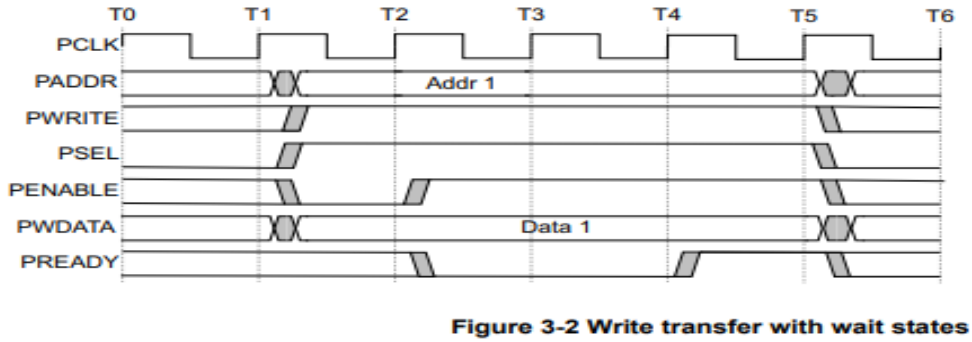

Figure 3-2 Write transfer with wait states

Figure 8: APB Write Transfer with wait timing diagram

Read Transfer

1. The Setup phase of the read transfer occurs at T1. The select signal, PSEL, is asserted, which means that PADDR, PWRITE and PWDATA must be valid. PWRITE should be held low.
2. The Access phase of the read transfer is shown at T2. where PENABLE is asserted.
3. The Completer must provide the data before the end of the read transfer. The transfer is extended if PREADY is driven LOW during an Access phase. Data will be provided by completer along with PREADY.

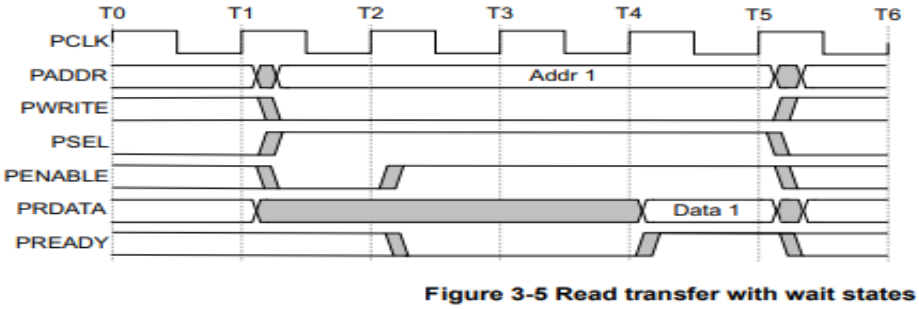

Figure 3-5 Read transfer with wait states

Figure 9: APB Read Transfer with wait timing diagram

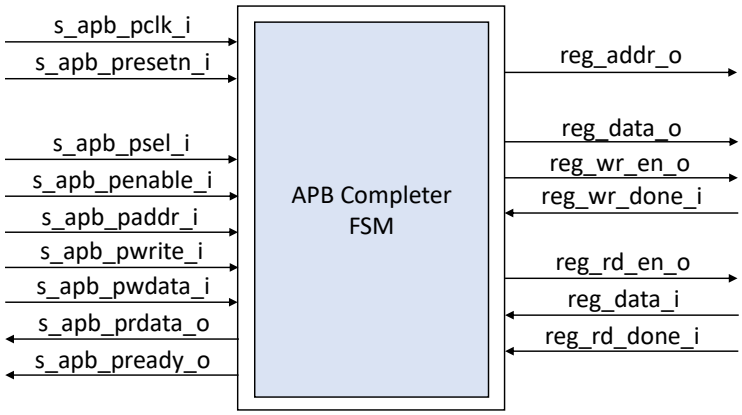

Figure 10: APB Interface Block Diagram

High level block diagram of APB completer is shown in Figure 12, and APB completer FSM is shown in Figure 13.

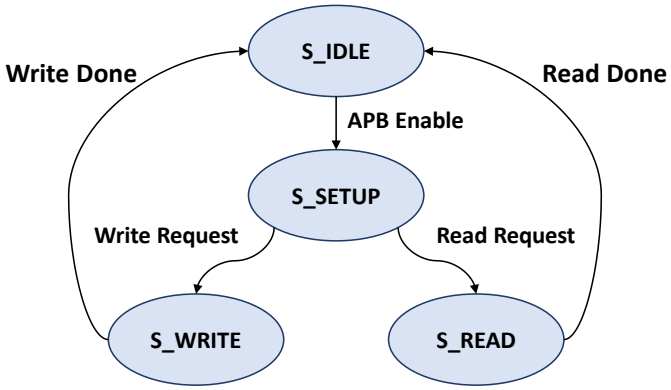

Figure 11: APB Completer FSM

IDLE State: Slave will be waiting in IDLE state until PSEL is received from master. Once PSEL becomes high FSM state moves to SETUP state.

SETUP State: Slave will stay in this state until it receives PENABLE from master. Once PENABLE is high WRITE signal will be checked to determine the next state. If WRITE is high next state will be WRITE else if WRITE is low next state will be READ.

WRITE State: In this state slave waits until the completion of write operation, once it receives write done from register set it moves to IDLE state by making PREADY high.

READ State: In this state slave waits until the completion of read operation, once it receives read done from register set it moves to IDLE state by making PREADY high.

3.6.Register Set

Register set provides simple read and write interface with the APB completer in order to provide access to UART Controller Status, Control and Data to the processor. Processor can set the configuration of UART Controller by performing APB Write to the control registers. It can get the status of UART Controller by performing APB read by providing address of status registers. In similar way it can write data to be transmitted into THR and access the received data by reading RBR.

Registers are kept similar to traditional 16550D UART Controller for simplicity only minimal and necessary features are supported. Apart from the registers this block also holds TX and RX FIFOs.

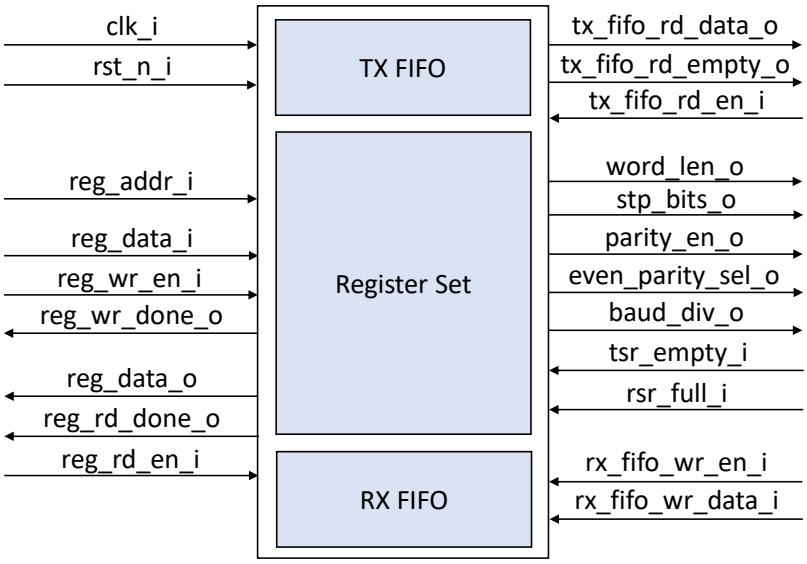

Figure 12: UART Register Set

The register map and the registers implemented in APB Based 16550 UART Controller is shown in Table 2. Registers shaded in grey are unimplemented.

Table 2: APB Based 16550 UART Controller Register Map

Address Offset	Register Name	DLAB = 0		DLAB = 1	
		Write	Read	Write	Read
0x00	THR_RBR_DLL	THR	RBR	DLL	
0x01	IER_DLM	IER		DLM	
0x02	IIR_FCR	FCR	IIR	FCR	IIR
0x03	LCR	LCR			
0x04	MCR	MCR			
0x05	LSR		LSR		LSR
0x06	MSR	MSR			
0x07	SCR	SCR			

DLL: Divisor Latch LSB: 0x00

DLL is read write register which can be accessed only when divisor latch access register bit-7 (DLAB) of LCR is high. Divisor value least significant byte needs to be loaded into the DLL register, this value is calculated based on the reference clock input frequency and the required baud rate.

Table 3: Divisor Latch Register LSB

7							0
DLL							

DLM: Divisor Lath MSB: 0x01

DLM is read write register which can be accessed only when divisor latch access register bit-7 (DLAB) of LCR is high. Divisor value most significant byte needs to be loaded into the DLM register, this value is calculated based on the reference clock input frequency and required baud rate.

Table 4: Divisor Latch Access Register MSB

7							0
DLM							

THR: Transmit Holding Register: 0x00

THR is used to load the data to be transmitted. This is a write only register and can be accessed only when divisor latch access register bit-7 (DLAB) of LCR is low. This register is nothing but head of the transmitter FIFO. From transmit FIFO data will be loaded to transmit shift register and data will be transmitted serially. Before writing new data into THR it is necessary to check whether there is sufficient space in transmit FIFO to hold new data. When line status register bit-5 is high it indicates that transmit FIFO is empty and new data to be transmitted can be loaded. However, there is no indication about the number of data characters present in transmit FIFO.

Table 5: Transmit Holding Register

7							0
THR							

RBR: Receive Buffer Register: 0x00

RBR holds the data received by the UART controller. This is read only register and can be accessed only when divisor latch access register bit-7 (DLAB) of LCR is low. This register is nothing but head of receiver FIFO. line status register bit-0 indicates that there is valid data in RBR and can be read out.

Table 6: Receive Buffer Register

7							0
RBR							

FCR: FIFO Control Register: 0x02

FCR controls the behaviour of transmitter and receiver FIFO's. This is a write only register. This register is used to enable the FIFO's, clear the FIFO's, set the receiver FIFO trigger level, and select type of DMA signalling.

Table 7: FIFO Control Register

Bits	Description
[0]	FIFO Enable bit, writing 1 to this bit will enable receiver and transmitter FIFO's This bit is always set to 1 as this implementation supports only FIFO mode
[1]	Clear receiver FIFO, writing 1 to this bit will clear receiver FIFO This bit will become low automatically after clearing the receiver FIFO
[2]	Clear transmitter FIFO, writing 1 to this bit will clear transmitter FIFO This bit will become low automatically after clearing the transmitter FIFO
[3]	DMA mode selection This is not implemented as DMA mode is not supported
[5:4]	Reserved
[7:6]	Receiver FIFO Trigger Level These bits are not used as interrupt feature is not implemented

LCR: Line Control Register: 0x03

Line control register is used to set various communication parameters.

Table 8: Line Control Register

Bits	Description
[1:0]	Data word length, this field specifies the number of bits in each character [1] [0] Data Word Length 0 0 5 Bits 0 1 6 Bits 1 0 7 Bits 1 1 8 Bits
[2]	Number of stop bits to be transmitted [2] Stop Bits 0 1 Stop bit 1 2 Stop bits when data word length is 6, 7 or 8 1.5 Stop bit when data word length is 5, this feature is not implemented
[3]	Parity enable bit, When set to 1 parity will be generated and transmitted, and receiver will be checking for received parity. When set to 0 parity generation and checking is disabled.
[4]	Even parity select, this bit will be effective only when parity is enabled When set to 1 even parity will be generated and checked When set to 0 odd parity will be generated and checked
[5]	Stick parity bit, This feature is not implemented
[6]	Break control bit,

	This feature is not implemented
[7]	Divisor Latch Access Bit (DLAB), When set to 1, access to DLL and DLM will be enabled When set to 0, access to THR, RBR and IER will be enabled

LSR: Line Status Register: 0x05

LSR provides current status of the controller.

Table 9: Line Status Register

Bits	Description
[0]	Data Ready, When there is valid received data available in RBR this bit will be set to 1 When there is no valid data available this bit will be set to 0
[1]	Overrun Error, When this bit is set to 1 it indicates that receiver FIFO was full and a new character is received by the receiver shift register. In this case data present in the receiver shift register will be overwritten and data present in receiver FIFO will not be affected. This bit is cleared as soon as the content of LSR is read out.
[2]	Parity Error, This bit indicates that there was a mismatch between the received parity and calculated parity. This bit is set to 1 when data word present at top of FIFO is associated with parity error.
[3]	Framing Error, This bit indicates that the received character did not have a valid stop bit. This bit is set to 1 when data word present at top of FIFO is associated with framing error.
[4]	Break Interrupt Indicator, This feature is not implemented
[5]	Transmit FIFO Empty, This bit is set to 1 when there is no data in transmit FIFO and it is empty When the transmit FIFO is occupied this bit is set to 0
[6]	Transmitter Idle, This bit is set to 1 when both transmit FIFO and transmit shift register are empty When any one among transmit FIFO or transmit shift register are occupied this bit is 0
[7]	Error in Receiver FIFO data, This bit is set to 1 if there are any errors in the data words present in receiver FIFO This bit will be 0 if there are no errors in the data words present in receiver FIFO

SCR: Scratch Pad Register: 0x07

SCR does not control UART operation in any way it is user register which can be read or written in anyway and is intend to hold temporary data.

Table 10: Scratch Pad Register

7							0
SCR							

2 Simulation Results

All the modules of APB Based 16550 UART controller are individually verified by simulating before integration. The baud rate generator is simulated by providing 100MHz clock frequency by selecting 115200 as required baud rate. The behaviour of TX and RX counters is captured in below Figures and the RX and TX clock enable pulses are generated as expected.

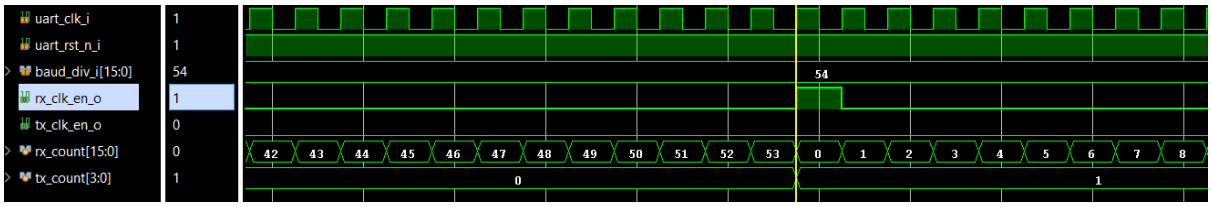

Figure 13: Baud Rate Generator RX Clock Enable

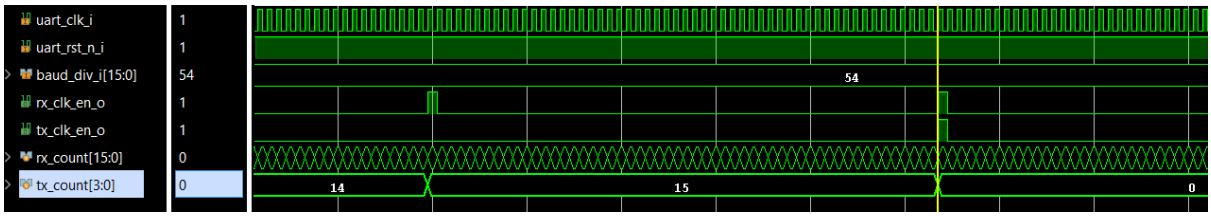

Figure 14: Baud Rate Generator TX Clock Enable

Synchronous FIFO is verified for different test cases and the behaviour of FIFO for continuous writes without read is shown in below Figure, FIFO full signal goes high as expected.

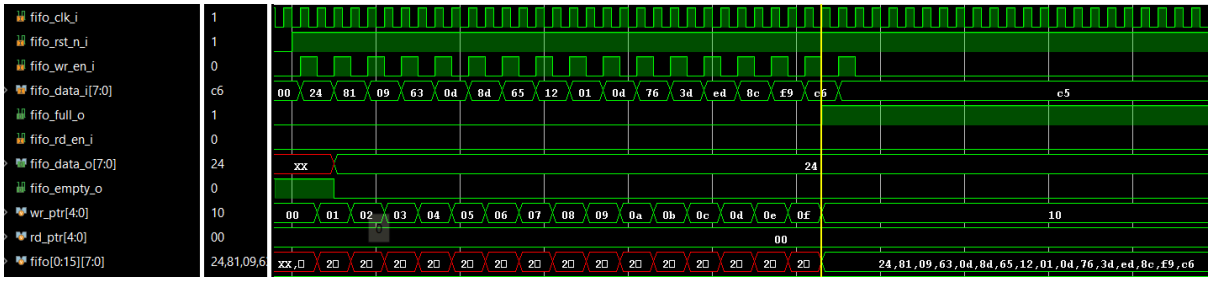

Figure 15: Synchronous FIFO Simulation Waveform

UART TX module is simulated with the help of baud rate generator. UART TX transmits 0x23, and by selecting word length 8, with single stop bit and parity disabled. As soon as TX FIFO Empty signal goes low FSM enters START state on receiving TX Clock Enable and when next TX Clock enable is received Read enable is generated to read data from FIFO. As the actual read enable pulse generated by FSM is high for 1 entire baud period FIFO Read logic converts it to single read enable pulse which will be high only for 1 cycle of reference clock to avoid multiple reads, this is achieved by detecting positive edge of the read enable generated by FSM.

Data is loaded in to shift register (data). As the UART TX Output is registered the output is reflected when next TX Clock Enable is received hence output looks as if it has shifted. Initially start bit, i.e. 0 gets transmitted followed by the data bits and stop bit that is high and Finally FSM enters IDLE state as there is no data available and TX FIFO Empty is high. Bit count is incremented as expected on transmitting each bit.

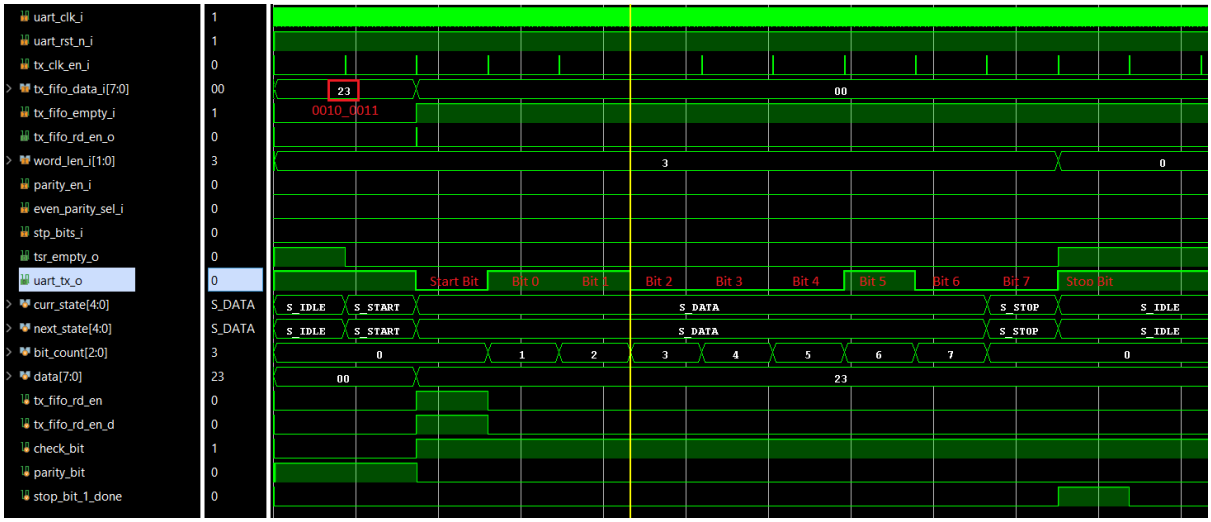

Figure 16: UART Transmitter Simulation Waveform

UART RX module is simulated with the help of UART TX and Baud Rate Generator. UART TX transmits 0x23, and by selecting word length 8, with single stop bit and parity disabled. The data transmitted by UART TX line is looped back to UART RX line in the testbench. UART RX line is synchronized to reference clock domain by using synchronizer Flip Flops d0, d1 and d2. Data from d2 will be sampled further. As soon as start bit high to low is detected Start state will be entered Tick counter starts incrementing and at the centre of start bit data will be sampled and checked whether it is valid start bit. Tick count is cleared and Data State will be entered. Each incoming bit will be sampled at centre and loaded to receiver shift register. After receiving all bits write enable pulse will be generated to write data to RX FIFO. As the write enable pulse will be high for 1/16 baud interval it is converted to single clock cycle pulse by detecting the edge.

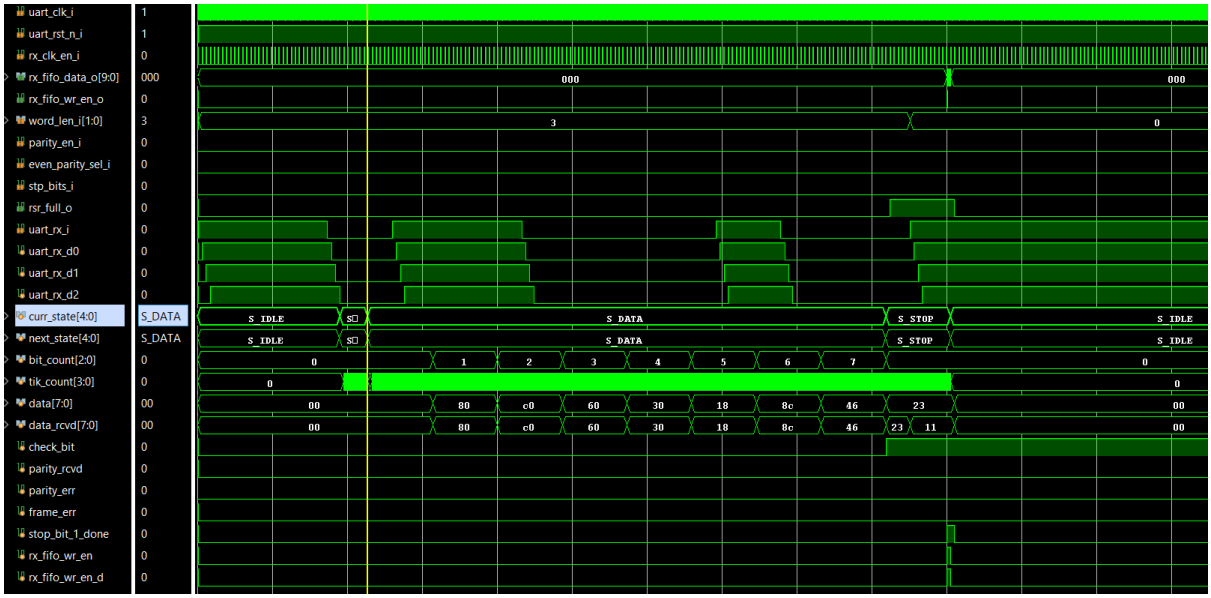

Figure 17: UART Receiver Simulation Waveform

APB Interface module is verified with the help of simple register set with 4 registers and the output is verified using waveforms. Data 0x32 is written to register 0 and verified by reading back register 0. Slave is selected by initiator by putting data, address on the bus. Write signal is made high whenever initiator has valid data on the bus. APB Interface process the incoming transaction as write transaction and provides write enable for register set along with data and address. After slave register set stores the data into register 0 it acknowledges the APB Interface module by making write done high. Complete write transfer takes 5 clock cycles. Similarly, when initiator wants to read the data, it puts address, data by keeping write signal as low and makes enable signal high. APB Interface block recognizes it as read transfer and provides read enable for register set, register set acknowledges the APB Interface Block by making read done high and provides the data from addressed register.

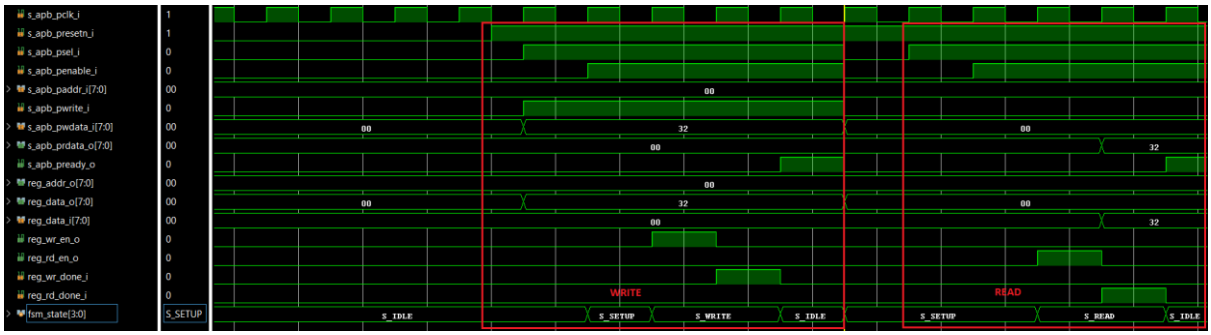

Figure 18: APB Interface Simulation Waveform

After integration all possible test cases are covered and the results are obtained as expected. The functionality of Line Control register and Line Status are also covered by injecting parity and frame errors, the errors are detected successfully. Functionality of FIFO is

also verified. Baud rate configuration functionality is verified by programming different divisor values to DLL and DLM registers.

3 References

[1] Liakot Ali, Roslina Sidek, Ishak Aris, Alauddin Mohd. Ali, Bambang Sunaryo Suparjo, “Design of micro-UART for SoC Application”, Computers & Electrical Engineering, Volume 30, Issue 4, 2004, Pages 257-268, ISSN 0045-7906,
https://www.sciencedirect.com/science/article/abs/pii/S004579060400014X

[2] M. Roopa, R. M. Vani and P. V. Hunagund, "UART controller as AMBA APB slave," *National Conference on Challenges in Research & Technology in the Coming Decades (CRT 2013)*, 2013, pp. 1-6, doi: 10.1049/cp.2013.2507.
https://ieeexplore.ieee.org/document/6851554

[3] Xilinx, “AXI 16550 UART v2.0 Product Guide (PG143)”
https://docs.xilinx.com/v/u/en-US/pg143-axi-uart16550

[4] Texas Instruments, “PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs”, Datasheet
https://media.digikey.com/pdf/Data%20Sheets/Texas%20Instruments%20PDFs/PC16550D.pdf

[5] ARM AMBA APB Specification, “AMBA APB Protocol Specification Version C”
https://developer.arm.com/documentation/ihi0024/c/

[6] Analog Devices Wiki, “UART: A Hardware Communication Protocol, Understanding Universal Asynchronous Receiver/Transmitter”
https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html