World Applied Sciences Journal 32 (3): 472-477, 2014
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2014.32.03.261

ASIC Implementation of Low Power Universal Asynchronous Receiver Transmitter

Badam Suresh, P. Teja Reddy, V.S.V. Srihari and S. Sivanantham

School of Electronics Engineering, VIT University, Vellore - 632014, Tamilnandu, India

Abstract: To Design a Universal Asynchronous Receiver/Transmitter (UART) module using ASIC design flow. UART is a main module in many applications like USB communication, Bluetooth communication and in many Network Applications [1]. Here, the design in this paper is implemented with Verilog HDL and ASIC flow of synthesis is done by Cadence RTL synthesizer for Front-End and Cadence Encounter for Back-End. In this paper, we propose a new interrupt handler block to reduce the number of hand shaking signals between CPU and UART by self servicing the interrupts. Therefore, the load on CPU is reduced and it can perform faster. When the design is being Synthesized using Cadence, it is optimized for low area and lesser delay in the critical path. The design thus created can be instantiated any number of times.

Key words: Interrupt Handler · Hand shaking · ASIC Design · UART implementation · Verilog HDL

INTRODUCTION

UART is abbreviated as Universal Asynchronous Receiver/Transmitter. It is a device which is used for transmitting or receiving data serially between two systems or in-between the same system according to a pre-defined protocol and the rate of transmission depends on the baud rate [2]. The data is transmitted through the device after receiving the THRE (Transmitter Hold Register Empty) interrupt so that the previous data sent is not interfered with the present data to be sent. Similarly, the data is received by the CPU immediately when the DR (Data Ready) interrupt is received because if the data is not taken immediately, there is a possibility that the next data packet may overwrite the current data in Hold Register. This process is well explained the next sections. We can observe a Hand shaking kind of protocol exists between CPU and UART. Therefore it is termed as Asynchronous Transmission [3].

The design of Lattice Semiconductor Corporation [4] is taken as reference in this work and it is implemented using ASIC (Application Specific IC) flow of synthesis. ASIC Design comes under the hierarchy of semi-custom IC design in which there exists some predefined cells upon which the required design is built. All these predefined cells are defined in the library and this library is loaded into the RTL Compiler Tool before the HDL Design files are loaded [5]. The design is synthesized by using appropriate constraints so that a reasonable slack value is obtained after synthesis. Low power constraints are added to the design to achieve low power using techniques like Clock-Gating, Multi V_t etc.. DFT (Design for Testability) constraints are added to the design to achieve a Testable design. Once the Gate-level Netlist is obtained after synthesizing and mapping the design to a technology library, the design is placed and routed using Cadence Encounter. By using Cadence Encounter, the GDSII file is generated. It can be passed to Process Engineer for further process of manufacturing.

The Paper Is Organized as Follows: In the next two sections, we discuss the Block Diagram and Operation of the device along with protocols that are included in the design. The fourth section deals with implementation of each block. The algorithms adopted for implementation of each block are clearly specified. The fifth section deals with addition of new block to the original design called as Interrupt service module. The sixth section deals with procedure of implementation of design using ASIC flow of synthesis. It includes adding low power constraints and DFT constraints [6]. The seventh section involves placing and routing using Encounter (Back-end flow of synthesis). The next section deals with results and discussion and providing conclusion.

Corresponding Author: Badam Suresh, School of Electronics Engineering, VIT University, Vellore - 632014, Tamilnandu, India.

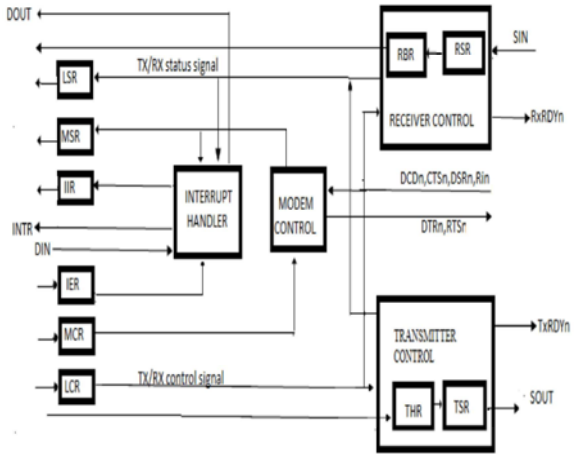

Fig. 1: Block diagram of UART

Block Diagram of Uart: The overall block diagram for the implementation of UART is shown in the Fig (1). It consists of 4 basic blocks. They are

- Transmitter Block
- Receiver Block
- Modem Control Block
- Interrupt Handler Block

The in-detail explanation of Transmitter, Receiver and Modem control blocks are given in section (IV). In the reference paper provided by the Lattice Semiconductors Corporation [4], the interrupt block in the design only recognizes the interrupts that occurred in the UART device and requests the CPU for service of interrupts. This reduces the speed of operation of the CPU and thereby increases the complexity in the handshaking signals between CPU and UART. In order to avoid this, we propose a model for Interrupt handling by providing an interrupt service block along with the interrupt block. These two blocks are shown combined in the block diagram. Detail description of the proposed Interrupt Handler block is shown in the section (V).

Functionality of Uart: The values that are loaded into the input registers controls the overall operation of the device. In total, there are 10 registers (4 input + 4 output + 1 internal to Transmitter + 1 internal to Receiver) that control the operation. All the registers are of 8bit length. The input registers are LCR (Line Control Register), MCR (Modem Control Register), IER (Interrupt Enable Register) and THR (Transmitter Hold Register). The output Registers are LSR (Line Status Register), MSR (Modem Status Register), IIR (Interrupt Identification Register) and RBR (Receiver Buffer Register). The register that is internal to the transmitter is TSR (Transmitter Shift Register) and the register that is internal to the receiver is RSR (Receiver Shift Receiver) [7].

Each register has its own purpose and format. The purpose and format for each register is specified in [2] and some of them are shown below

Line Control Register (LCR): It is used for controlling the word length of the transmitted bit.

Table 1: Description of Each bit:

B7	B6	B5	B4	B3	B2	B1	B0
0	SB	SP	EPS	PEN	STB	WLS[1]	WLS[0]

SB - Set break - Causes break condition while transmitting
SP-Stick parity - Parity bit is stick to either logic '1' or logic '0'
EPS - Even Parity stick
STB -Stop Bit length
WLS [1:0]-word length select bits

The stop bit, word length and the parity bit are selected according to following tables.

Stop Bit Length:

Table 2 :Table for determining Stop bit length

LCR [2] - STB	Word length	Stop bit length (Bit Times)
0	5,6,7,8	1
1	6,7,8	2

Word Length:

Table 3 :Table for determining length of the transmitted word

1LCR [1]	LCR [0]	Word length
0	0	5
0	1	6
1	0	7
1	1	8

Parity Selection:

Table 4:Table for determining length of the Parity Bit

LCR [5]	LCR [4]	LCR [3]	Parity Selection
X	X	0	No parity
0	0	1	Odd parity
0	1	1	Even Parity
1	0	1	Stick Parity 1
1	1	1	Stick Parity 0

Modem control Register (MCR): It is used to control the modem output signals.

Table 5 :Description of Each bit:

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	0	RTS	DTR

RTS - Controls the Request to send (RTSn) output
DTR - Controls the Data Terminal Ready (DTRn) output

Interrupt Enable Register (IER): It is used to enable the required interrupts.

Table 6 :Description of Each bit:

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	MSI	RLSI	THRI	RBRI

MSI - Modem Status Interrupt
RLSI - Receiver Line Status Interrupt
THRI - Transmitter Hold Register Interrupt
RBRI - Receiver Buffer Register Interrupt

Line Status Register (LSR): It is used to indicate the status of the received word and generate the interrupts correspondingly.

Table 7 :Description of Each bit:

B7	B6	B5	B4	B3	B2	B1	B0
0	0	THRE	BI	FE	PE	OE	DR

THRE - Transmitter Hold Register Empty
BI - Break Interrupt
FE - Framing Error
PE - Parity Error
OE - Overrun Error
DR - Receiver Data Ready

Modem Status Register (MSR): It is used to indicate the status of the modem and generate the modem status interrupts correspondingly.

Table 8 :Description of Each bit

B7	B6	B5	B4	B3	B2	B1	B0
DCD	RI	DSR	CTS	DDCD	TERI	DDSR	DCTS

DCD - Data Carrier Detect - compliment of DCDn input.
RI - Ring Indicator - compliment of RIn input.
DSR - Data Set Ready -compliment of DSRn input.
CTS - Clear to Sent -compliment of CTS input
DDCD - Delta DCD - change in DCDn input
TERI - Trailing Edge of RIn - falling edge of RIn
DDSR - Delta DSR - change in DSRn
DCTS - Delta CTS - change in CTSn

Interrupt Identification Register (IIR): It is used to identify the order of interrupts to be serviced when more than one interrupt occurs a t the same time.

Table 9: Description of Each bit

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	ID2	ID1	ID0	STAT

The ID2, ID1, ID0 bits are set according to the priority of interrupts occurred. The input registers and output registers are clearly specified in the block diagram. The D_{in} and D_{out}holds values in CPU.

Implementation of Induvidual Blocks: The clock and reset signals are provided to all blocks in the design. The in-detail explanation of each block in the block diagram is discussed below. The global clock is divided according to baud rate of the transmitter and then it is supplied to all the blocks in UART. Thus no confusion of sampling of bits at the receiver occurs.

Transmitter Block: The main objective of the Transmitter block is to transmit the given parallel data in serial format. The data is transmitted according to specified format given below.

Table 10: Format for transmission of bits

Stop Bit	Parity Bit	DATA BITS	Start Bit

When the transmitter is provided input data to transmit, it first transmits the START bit (logic '0') followed by the data bits and then followed by a parity bit and STOP bit (logic '1'). The length of data bits and stop bits to be transmitted is decided by the values loaded in LCR. Data bits can reach a maximum of 8 in this design. STOP bits can reach a maximum of 2. The parity and START bits are of unit length.

The inputs to transmitter block are LCR and THR whereas outputs are TXRDYn, LSR and Sout. The THR register holds the data to be transmitted whereas LCR register is used to select the format of the transmitted bits. The output is transmitted through Sout. The flag bits during operation are stored in LSR. It is used to activate interrupts in the Interrupt handler block. TXRDYn is an active low transmitter ready signal that is used to indicate data can be sent. The TSR (Transmitter Shift Register) is internal to the transmitter and performs shifting operation.

Implementation is done using FSM. It's state transition diagram is shown in Fig (2) It consists of 6 states. Functionality of each state is shown in table below.

Table 11: States of Transmitter

State	Function
S0 (start)	Transmission of START bit
S1 (Shift)	Transmission of Data Bits
S2 (Parity)	Transmission of Parity Bit
S3 (Stop_1_bit)	Transmission of 1st Stop Bit
S4 (Stop_2_bit)	Transmission of 2nd Stop Bit

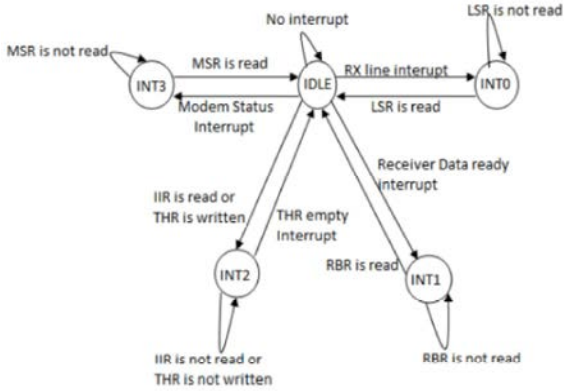

Fig. 2: Transmitter State Daigram

Receiver Block: The main objective of Receiver block is to receive serially transmitted data from transmitter and convert to original parallel data.

The inputs to the receiver are LCR, S_{in} and outputs are LSR, RXRDYn and RBR. The LCR is provided as input to the receiver in order to calculate the length of bits to be transmitted. S_{in} is the input data obtained from transmitter. Corresponding flag bits during the operation like overrun Error, parity error, framing error etc..are stored in LSR. It is used to generate the corresponding Interrupts in the Interrupt Handler Block. RSR is a 8 bit register internal to the receiver block that performs the serial to parallel shift operation of received bits. The output data obtained finally is stored in RBR. RXRDYn is an active low signal that is used to indicate the data is ready to be received.

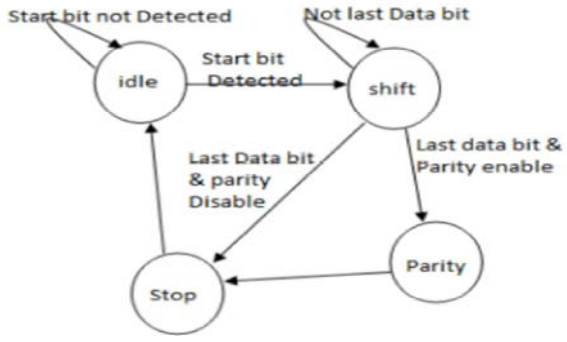

Fig. 3: Receiver State Diagram

Implementation is done using FSM. Its state transition diagram is shown in Fig (3). It consists of 4 states. Functionality of each state is shown in table below.

Table12: States of Receiver

State	Function
S0 (Idle)	Detection of Start Bit
S1 (Shift)	Perform shift operation with input bit until last data bit is received
S2 (Parity)	Receive the parity bit and check for errors during transmission
S3 (Stop)	Transfer received data to output register.

Modem Control Block: The purpose of modem control block is to indicate the status of modem to CPU. The input signals to this block are MCR, DCDn, CTSn, DSRn and RIn. The outputs of the block are DTRn, RTSn and MSR. The MCR is used to set the outputs DTRn (active low Data Terminal Ready) and RTSn (active low Request to Send). The DCDn, CTSn, DSRn and RIn are active-low Data-Carrier-Detect, Carrier-to-Send, Data-Set-Ready and Ring-indicator signals. If there is any change in these signals, the corresponding bit is set in MSR and required interrupts are generated in the interrupt handler.

Proposed Interrupt Handler: In the reference design of Lattice Semiconductor Corporation [2], the Interrupt handler block is designed only to generate the interrupts during the process. The generated interrupts are serviced by CPU. Here, in this paper, we propose a new interrupt handler that has self servicing capability.

It Consists of Two Blocks:

- Interrupter
- Interrupt Service

Interrupter: The main objective of interrupter is to generate interrupts during the normal operation of UART. The inputs to this block are IER, LSR and MSR. The outputs of this block are INTR and IIR. Apart from these, it also receives the interrupt reset signals from the interrupt service block. The reset signals are generated when the interrupts are serviced by the interrupt service block. IER is used to enable the 4 interrupts. LSR and MSR are used to identify the interrupt sources. According to the received interrupts, interrupt priority is done and is stored in IIR. INTR is an active high signal that indicates the occurrence of an interrupt.

The following table indicates the priorities of the interrupts

It is implemented using an FSM. The state transition diagram is shown in Figure (4)

It consists of 5 states. Functionality of each state is shown in table below.

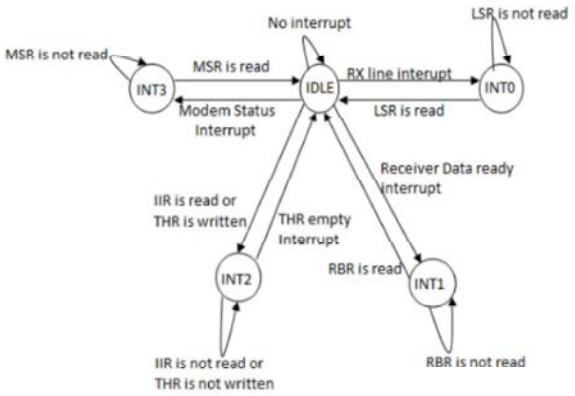

Fig. 4: Interrupter State Diagram

Table 13: Priorities of Interrupts

Priority	ISR[3:0]	Source of Interrupt	Reset Control
None	xxx1	None	None
Highest	0110	LSR Error flags	
		(OE/PE/FE/PI)	Read LSR
Second	0100	LSR DR flag	Read RBR
Third	0010	LSR THRE flag	Read IIR or write THR
Fourth	0000	MSR [3:0] flags	Read MSR

Table 14: States of Interrupter

State	Function
S0 (IDLE)	No interrupt received
S1 (int 0)	RLSI interrupt
S2 (int 1)	Receiver Data Ready Interrupt
S3 (int 2)	THR Empty interrupt
S4 (int 3)	Modem status interrupt

The main objective of this block is to service the interrupts that are generated from the interrupter. The inputs of this block are LSR,MSR,IIR,DIN and RBR. The outputs of this block are Doutand THR. Apart from these outputs, it also generates the interrupt reset signals that are used to reset the interrupts in the interrupter block.

When RLSI interrupt is occurred, the previous data is transmitted again before it is reset. When receiver data ready interrupt is occurred, the data in RBR is transferred to Dout before it is reset. When THR Empty interrupt is occurred, next data is written to THR from Din before it is reset. When Modem Status interrupt is occurred, the process is to be halted temporarily until the corresponding signals are available. The MSI interrupt can be reset before the process is halted.

The interfacing between the interrupter and interrupt handler is shown in Figure 5

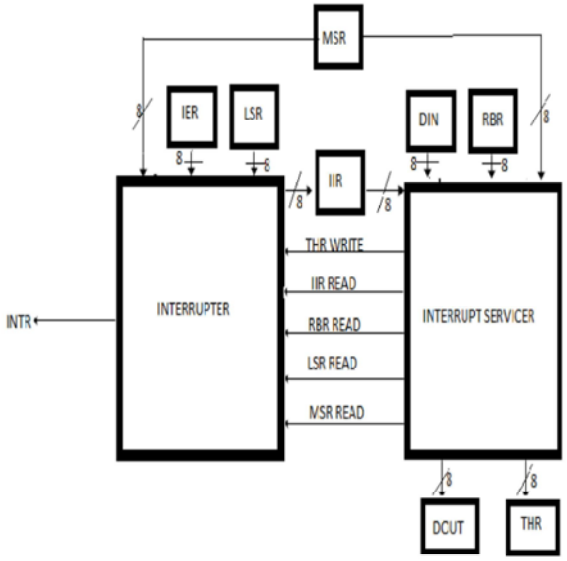

Fig. 5: Internal block diagram of Interrupt Handler

Implementation Using Asic Synthesis

Front-end Design: The designed Verilog code is implemented by Cadence RTL compiler and is synthesized using ASIC flow. The technology library files and.lef files are loaded into RTL compiler before reading HDL design files. The paths are appropriately set for library and current working directory. After elaborating the design, add constraints for clock, clock uncertainty and clock skew. Generate.VCD (Value Change Dump) file while simulating the design with NCSIM simulator. The generated VCD file is imported into Cadence RTL compiler for estimation of dynamic power. Synthesize the design into generic and observe for data path violations. After the design is synthesized to generic, it is mapped to specified technology library [1]. After, mapping the design to technology library, area, power and timing reports can be observed. If there are any violations in observed reports, constraints are appropriately adjusted in order to solve the violations.

To achieve low power design, low power constraints for clock gating and multiVt are set. For making the design testable, DFT constraints are also included. Now, the gate level netlist obtained is saved in a file and it is processed for back end design.

Back-End Design: In the gate level netlist obtained from front-end design, for all the I-O ports, Input pads and Output pads are attached. The resultant netlist obtained is imported in Cadence SOC (System on Chip) Encounter. Specify appropriate floor planning and size for the die and

pads. After the floor planning is complete, the core die is surrounded by power rings (Vdd and Vss). Also give horizontal and vertical power stripes across the die. Now, Place the design macros across the die so that optimum design is achieved. Once the Placement of Macro cells is done, make a trail route before making a permanent routing across the die. If the die is routed in optimum fashion, make the routing permanent [1]. In Cadence SOC Encounter, permanent routing is done by NANO-ROUTE. The obtained GDSII file is passed to process Engineer for manufacturing. The GDSII file is shown in the results section.

RESULTS AND DISCUSSION

We used Cadence NCSIM simulator for verifying the design code, Cadence RTL Compiler for synthesizing the design (Front-End) and Cadence SOC Encounter for obtaining the GDSII file (Back-End) The power of the design is reduced after applying low power constraints of clock gating and multi Vt.. The area, power and timing reports of the design are specified in table shown below.

It is found that the design can operate without any errors up to 400MHZ. The frequency of operation is found by calculating the delay in the critical path that is observed along with timing report.

The GDSII file obtained after all the typical steps of back end like floor planning, placement and routing is shown below in Figure 6.

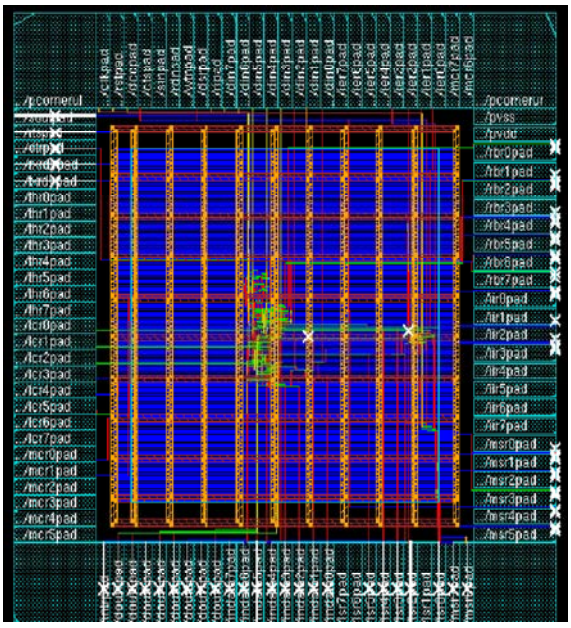

Fig. 6: GDSII file of the design

Table 15: Reports of design

Technology Node	180nm
Area	10259 μm^2
Power	1.4928 mW
Gates	506
Core Size	11.080 mm^2
Operating Frequency	400 MHz

CONCLUSION

The design of UART is done with Verilog HDL and implemented in ASIC synthesis flow. The power of the design is reduced by applying low power constraints and the design is also testable. The testable design is achieved by applying DFT (Design for Testability) constraints. A new approach for handling interrupts in UART is proposed. In this approach, the interrupts are serviced itself without the interference of CPU by the help of interrupt service block. The proposed interrupt handler is included in the design and the functionality UART is successfully verified. Therefore, we conclude that with this proposed design, the number of hand shaking signals between CPU and UART decreases and CPU can perform faster.

REFERENCES

1. https::www.datasheetarchive.com/4--Uart+ applications - datasheet.html (online)
2. “Programming DSPIC (Digital signal Controllers) in BASIC”, Zoran Milivojeviæ, Djordje Šaponjiæ published by mikroelektronika http://www.mikroe.com/products/view/265/programming-dspic-mcu-in-basic/
3. He Chun-zhi, Xia Yin-shui and Wang Lum-yao, 2011. "A Universal Asynchronous Receiver Transmitter Design," Natural Science Foundation of china (No.61041001) pp: 691-694.
4. Fang Yi-yuan and Chen Xue-jun, 2011. “Design and Simulation of UART Serial Communication Module Based on VHDL”, 3rd International Workshop on Intelligent Systems and Applications (ISA), May 2011.
5. Yongcheng Wang and Kefei Song, 2011. "A New Approach to Realize UART", International Conference on Electronic &Mechanical Engineering and Information Technology, pp: 2749-2752.
6. Dipanjan Bhadra, Vikas S. Vij and Kenneth S. Stevens, 2013. “A Low Power UART Design Based on Asynchronous Techniques” Circuits and Systems (MWSCAS), 2013 IEEE 56th International Midwest Symposium on , 21(24): 4-7.
7. He Chun-zhi, Xia Yin-shui and Wang Lum-yao, 2011. "A Universal Asynchronous Receiver Transmitter Design," Natural Science Foundation of china (No.61041001) pp: 691-694.