

IMPLEMENTATION OF USB TRANSCEIVER MACROCELL INTERFACE

A. Vamshidhar Reddy¹, A.Laxman²,Prakash³L, T.Satyanarayana⁴

¹Assoc.Prof. ECE Department, RRS COLLEGE OF ENG. & TECH.,AP,India,avamshireddy@gmail.com

²Asst.Prof. ECE Department, MAHAVEER INST.OF SCIENCE & TECH., AP, India, amgothlaxman@gmail.com

³Asst.Prof., ECE Department, DVR COLLEGE OF ENG. & TECH., AP, India, laudiya.prakash@gmail.com

⁴Asst.Prof., ECE Department, DVR COLLEGE OF ENG. & TECH., AP, India , satyant234@gmail.com

Abstract

The universal serial bus (USB) transceiver macro cell interface (UTMI) is a two wire, bi-directional serial bus interface. UTMI consists of transmitting and receiving sections, in which the transmitter of the UTMI sends data to different USB devices through D+ and D- lines whereas the receiver gets data on the same lines.

UTMI is one of the important functional blocks of USB controller, which can transmit and receive data to or from USB devices. There are three functional blocks present in USB controller; those are Serial Interface Engine (SIE), UTMI and Device Specific Logic (DSL). The parallel data from SIE is taken into the transmit hold register and this data is sent to transmit shift register from where the data is converted serially. This serial data is bit stuffed to perform data transitions for clock recovery and NRZI (1) encoding. Then the encoded data is sent on to the serial bus. When the data is received on the serial bus, it is decoded, bit unstuffed and is sent to receive shift register. After the shift register is full, the Data is sent to receive hold register.

This presentation reveals the FPGA implementation of UTMI transmission rate providing with USB 2.0 specifications. Further UTMI has been designed by using VHDL code and simulated, synthesized and programmed to the targeted Spartan family of FPGA in the Xilinx environment.

Index Terms: INTRODUCTION, USB TRANCIEVER MACROCELL INTERFACE AND DESIGN OF HARDWARE MODEL

1. INTRODUCTION

1.1 Introduction

The fig-1 shows the block diagram of USB controller, which is present in every USB device. There are three major functional blocks in a USB 2.0controller.The USB 2.0 Transceiver Macrocell Interface (UTMI), the Serial Interface Engine (SIE), and the device specific logic.

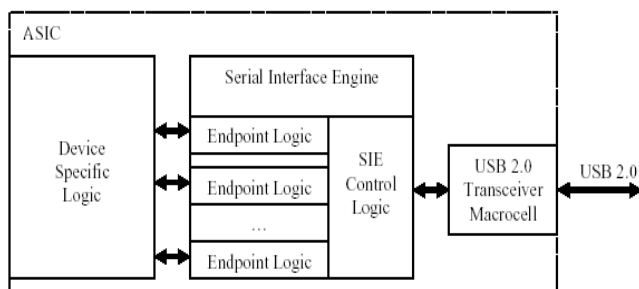


Fig1-Block diagram of USB controller.

UTMI

This block handles the low level USB protocol and signaling. This includes features such as; data serialization and deserialization, bit stuffing and clock recovery and synchronization. The primary focus of this block is to shift the clock domain of the data from the USB 2.0 rate to one that is compatible with the general logic in the ASIC.

The UTMI is designed to support HS/FS, FS Only and LS Only UTM implementations. The three options allow a single SIE implementation to be used with any speed USB transceiver. A vendor can choose the transceiver performance that best meets their needs.

A HS/FS implementation of the transceiver can operate at either a 480 Mb/s or a 12 Mb/s rate. Two modes of operation are required to properly emulate High-speed device connection and suspend/resume features of USB 2.0, as well as Full-speed connections if implementing a Dual-Mode device.

FS Only and LS Only UTM implementations do not require the speed selection signals since there is no alternate speed to switch to.

Serial Interface Engine

This block can be further sub-divided into 2 types of sub-blocks; the SIE Control Logic and the Endpoint logic. The SIE Control Logic contains the USB PID and address recognition logic, and other sequencing and state machine logic to handle USB packets and transactions. The Endpoint Logic contains the endpoint specific logic: endpoint number recognition, FIFOs and FIFO control, etc. Generally the SIE Control Logic is required for any USB implementation while the number and types of endpoints will vary as function of application and performance requirements.

SIE logic module can be developed by peripheral vendors or purchased from IP vendors. The standardization of the UTMI allows compatible SIE VHDL to drop into an ASIC that provides the macro cell.

Device Specific Logic:

This is the glue that ties the USB interface to the specific application of the device.

Applications

The UTMI has been developed into a common code (Generalized USB Transceiver) which can be used for developing the complete USB device stack.

Some of the Low speed and High speed USB devices, which are presently in the market, are:

1. Optical Mouse
2. Key Board
3. Printer
4. Scanner
5. Joy Stick
6. Memory Stick
7. Flash Memory
8. Mobiles
9. Video cameras

2. USB TRANCIEVER MACROCELL INTERFACE (UTMI) Introduction

Universal Serial Bus (USB) Transceiver Macrocell Interface (UTMI) is one of the most important blocks of USB Controller. This block handles the low level USB protocol and signaling. This includes features such as data serialization, de serialization, bit stuffing, bit de stuffing, Non Return to Zero Invert on '1'(NRZI) encoding, decoding, clock recovery and synchronization. The primary focus of this block is to shift the clock domain of the data from the USB 2.0 rate to one that is compatible with the general logic in the ASIC.

Key features of the USB 2.0 Transceiver

- Supports 480 Mbit/s "High Speed" (HS)/ 12 Mbit/s "Full Speed" (FS), FS Only and "Low Speed" (LS) Only 1.5 Mbit/s serial data transmission rates.
- Utilizes 8-bit parallel interface to transmit and receive USB 2.0 cable data
- SYNC/EOP generation and checking
- High Speed and Full Speed operation to support the development of "Dual Mode" devices
- Data and clock recovery from serial stream on the USB
- Bit-stuffing/unstuffing; bit stuff error detection
- Holding registers to stage transmit and receive data
- Logic to facilitate Wake Up and Suspend detection
- Ability to switch between FS and HS terminations/signaling
- Single parallel data clock output with on-chip PLL to generate higher speed serial data clocks.

2.2 Functional Block Diagram and Description

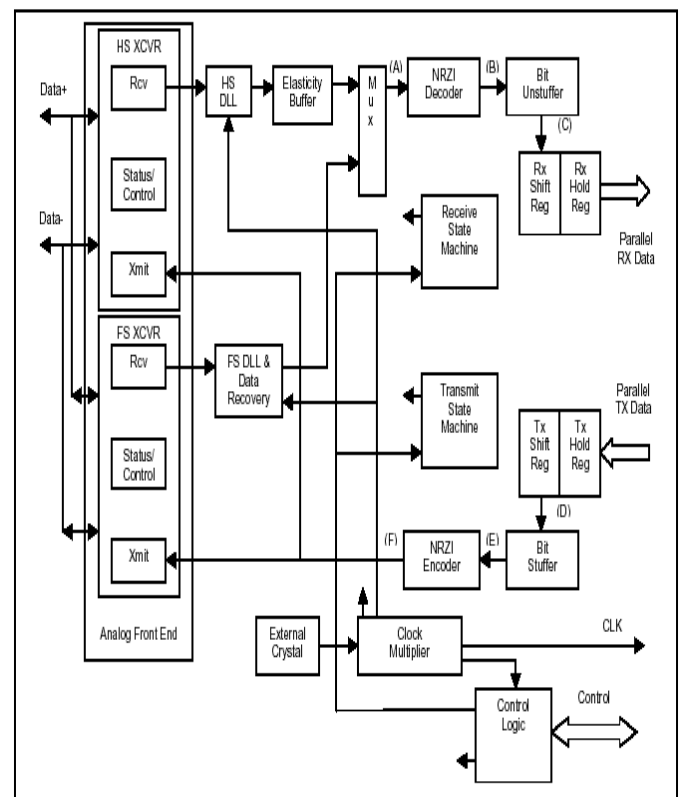


Fig2- UTMI Functional Block Diagram.

2.3 UTMI Signal Descriptions

The Table-1 gives the summary of system interface signals those are referred by the SIE and the UTMI. Each of the signals given in the table 1 is clearly described below.

2.3.1 System Interface Signals

Table-1 System Interface Signals

| Name | Direction | Active Level | Description |
|-----------------|-----------|--------------|---|
| CLK | Output | Rising-Edge | Clock. This output is used for clocking receive and transmit parallel data. 60 MHz HS/FS, with 8-bit interface 30 MHz HS/FS, with 16-bit interface 48 MHz FS Only, with 8-bit interface 6 MHz LS Only, with 8-bit interface See section 4.1.1 for more information on CLK . |
| Reset | Input | High | Reset. Reset all state machines in the UTM. |
| Xcvr Select | Input | N/A | Transceiver Select. This signal selects between the FS and HS transceivers: 0: HS transceiver enabled 1: FS transceiver enabled This signal is not provided in FS Only and LS Only transceiver implementations. See section 4.1.2 for more information on XcvrSelect . |
| Term Select | Input | N/A | Termination Select. This signal selects between the FS and HS terminations: 0: HS termination enabled 1: FS termination enabled This signal is not provided in FS Only and LS Only transceiver implementations. See section 4.1.3 for more information on TermSelect . |
| SuspendM | Input | Low | Suspend. Places the Macrocell in a mode that draws minimal power from supplies. Shuts down all blocks not necessary for Suspend/Resume operation. While suspended, TermSelect must always be in FS mode to ensure that the 1.5K pull-up on DP remains powered. 0: Macrocell circuitry drawing suspend current 1: Macrocell circuitry drawing normal current |
| LineState (0-1) | Output | N/A | Line State. These signals reflect the current state of the single ended receivers. They are combinatorial until a "usable" CLK is available then they are synchronized to CLK . They directly reflect the current state of the DP (LineState[0]) and DM (LineState[1]) signals: DM DP Description 0 0 0: SE0 0 1 1: 'J' State 1 0 2: 'K' State 1 1 3: SE1 See section 4.1.4 for more information on LineState . |

2.3.2 USB Interface Signals

Table-2 USB Interface Signals

| Name | Direction | Active Level | Description |
|------|-----------|--------------|--------------------|
| DP | Bidir | N/A | USB data pin Data+ |
| DM | Bidir | N/A | USB data pin Data- |

Table-2 gives the summary of USB interface signals. These are bi-directional signals and they are used to connect the USB device to the outer world. The data transmission and reception occur on these two lines. If bit '1' is present on DP line, bit '0' will present on DM line in the same clock duration. This is required because if bit '1' i.e., +5V is present on DP line, a magnetic field will be created around the line. This magnetic field may affect the nearby lines. To compensate the affect of magnetic field, a bit '0' i.e., -5V is present on DM line. The net effect of the magnetic field will be zero. Thus the noise is reduced.

2.4 Operational Modes

The OpMode signals are capable of inhibiting normal operation of the transceiver and evoking special test modes. The table 3 gives the summary of operational modes. Each of the signals given in the table 3 is clearly described below.

There are 3 test modes:

- Normal Operation (0)
- Non-Driving (1)
- Disable Bit Stuffing and NRZI encoding (2)

Table-3 Operational Modes

| OpMode (0-1) | Input | N/A | Operational Mode. These signals select between various operational modes: |
|--------------|-------|-----|---|
| | | | [1] [0] Description |
| | | | 0 0 0: Normal Operation |
| | | | 0 1 1: Non-Driving |
| | | | 1 0 2: Disable Bit Stuffing and NRZI encoding |
| | | | 1 1 3: Reserved |

Mode 0 allows the transceiver to operate with normal USB data decoding and encoding.

Mode 1 allows the transceiver logic to support a soft disconnect feature which tri-states both the HS and FS transmitters, and removes any termination from the USB making it appear to an upstream port that the device has been disconnected from the bus.

Mode 2 disables Bit Stuff and NRZI encoding logic so 1's loaded from the Data In bus becomes 'J's on the DP/DM lines and 0's become 'K's. Note that this mode affects the automatic SYNC Pattern and EOP generation by TX Valid. It is disabled so that Chirps can be generated on the USB. The operation of the receiver is undefined.

Changing the OP Mode signals under all other conditions, while the transceiver is receiving or transmitting data will generate undefined results.

2.5 Bi-directional 8-bit Interface

An option for the UTMI is to attach the block described in the fig-3 below to the 8-bit interface. This option provides an 8-bit bi-directional bus, minimizing the connection count. When this option is applied, 8 data lines will be presented by the transceiver, where Data0-7 is a bi-directional data bus.

If TX Valid is asserted (1) then the signals Data0-7 accept transmit data from the SIE. If TX Valid is negated (0) then the signals Data0-7 present received data to the SIE.

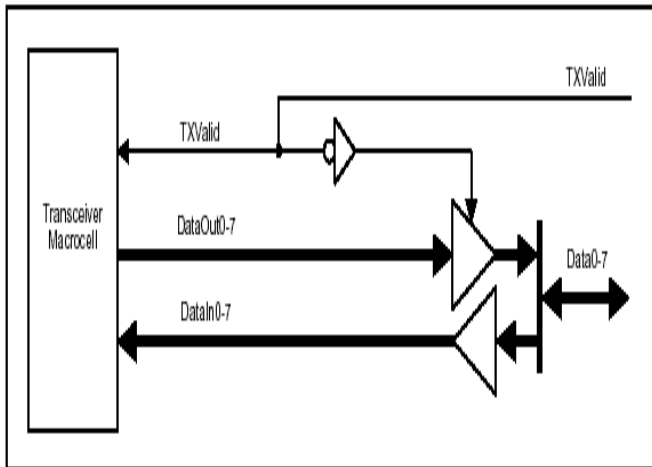


Fig-3 8-Bit Bi-directional Data Bus Interface

2.6 UTM Entity Diagram

The entity diagram of UTMI is shown in the fig- 4. The figure shows all the signals of UTMI and their direction. All the input signals to the UTMI are from SIE and all the output signals are to the SIE.

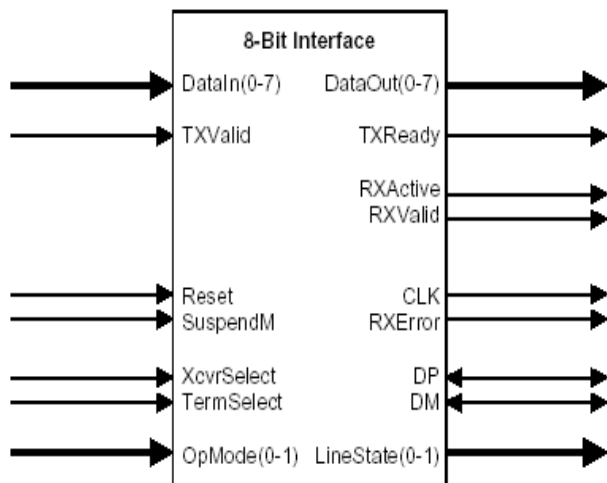


Fig-4 UTM Entity Diagram

3. DESIGN OF HARDWARE MODEL

The UTMI is divided into two modules, which are the transmitter module and the receiver module.

3.1 The Transmitter module Specifications

The transmitter module of UTMI has been implemented by considering the following specifications.

- The SYNC pattern ‘01111110’ has to be transmitted immediately after the transmitter is initiated by the SIE.
- After six consecutive ‘1’s occur in the data stream, a zero to be inserted.
- The data should be encoded using Non Return to Zero Invert on 1(NRZI -1) encoding technique.
- The EOP pattern two single ended zeros (D+ and D- lines are carrying zero for two clock cycles) and a bit 1 have to be transmitted after each packet or after SIE suspends the transmitter.

The behavior of the Transmit State Machine given by USB 2.0 specifications is described and illustrated in Fig.5

The Reset signal forces the state machine into the Reset state which negates TX Ready. When Reset is negated the transmit state machine will enter the TX Wait state.

In the TX Wait state, the transmit state machine looks for the assertion of TX Valid. When TX Valid is detected, the state machine will enter the Send SYNC state and begin transmission of the SYNC pattern.

When the transmitter is ready for the first byte of the packet (PID), it will enter the TX Data Load state, assert TX Ready and load the TX Holding Register. The state machine may enter the TX Data Wait state while the SYNC pattern transmission is completed.

TXReady is used to throttle transmit data. The state machine will remain in the TX Data Wait state until the TX Data Holding register is available for more data. In the TX Data Load state, the state machine loads the Transmit Holding register. The state machine will remain in the TX Data Load state as long as the transmit state machine can empty the TX Holding Register before the next rising edge of CLK.

When TXValid is negated the transmit state machine enters the Send EOP state where it sends the EOP. While the EOP is being transmitted TXReady is negated and the state machine will remain in the Send EOP state. After the EOP is transmitted the Transmit State Machine returns to the TX Wait state.

The summary of the above description is given below.

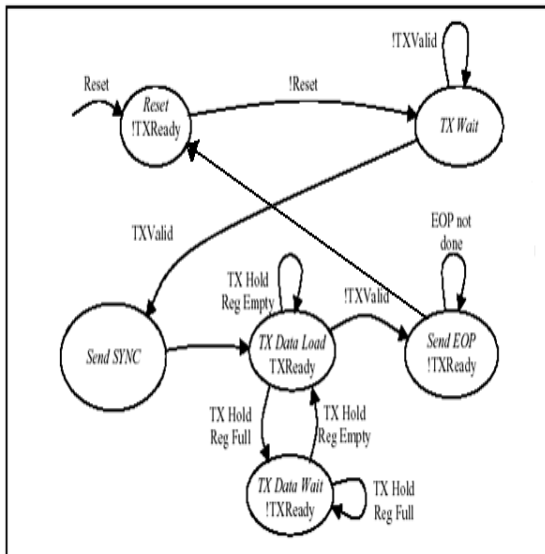


Fig-5 Transmit state machine

- Transmit must be asserted to enable any transmissions.
- The SIE asserts TXValid to begin a transmission.
- The SIE negates TXValid to end a transmission.
- After the SIE asserts TXValid it can assume that the transmission has started when it detects TXReady asserted.
- The SIE assumes that the UTMI has consumed a data byte if TXReady and TXValid are asserted.
- TXValid and TXReady are sampled on the rising edge of CLK.
- The SIE must use LineState to verify a bus Idle condition before asserting TXValid in the TX Wait state.

Fig-6 shows the timing relationship between TXValid, DataIn, TXReady and the transmitted data (DP/DM).

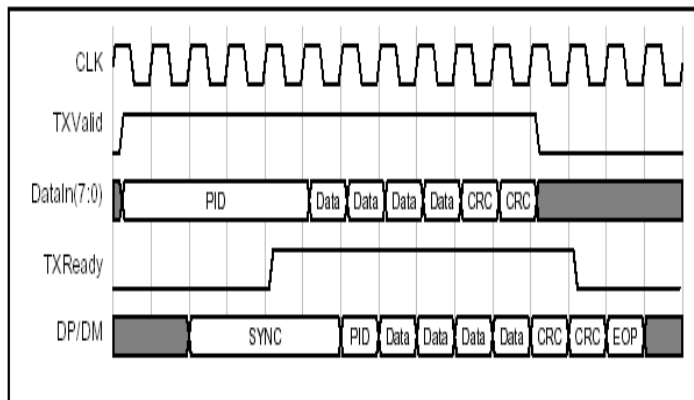


Fig-6 Transmit Timing for a Data packet

The SIE negates TXValid to complete a packet. Once negated, the Transmit State Machine will never re-assert TXReady until after the EOP has been loaded into the Transmit Shift Register.

3.2 The Transmitter Module Design

The transmitter module is designed by considering all the above specifications. VHDL is used to design the transmitter module. The transmitter module of the UTMI consists of various blocks such as SYNC generator, transmit hold and shift register, bit stuffer, NRZI encoder and EOP generator. A transmit state machine is developed by considering all the states given by USB 2.0 transmit state machine. Initially the transmitter is at Reset state where the reset signal is high. If reset signal goes low state the state of the transmitter is changed to TX wait state where it is waiting for assertion of TX valid signal by the SIE.

3.2.1 SYNC generator

When the TX valid signal is asserted by the SIE, transmit state machine enters into send Sync state where a signal called of the clock outside the state machine. When this signal is enabled sync enable is asserted. This signal is checked at every rising edge, a sync pattern “01111110” is sent to the NRZI encoder.

3.2.2 Transmit hold and shift register

The data byte placed on the data lines by the SIE sampled by the UTMI at the rising edge of the clock. For this purpose, an 8-bit vector is declared in the entity declaration of the transmitter module. This 8-bit vector is considered as transmit hold and shift register.

The transmit hold and shift register is loaded with 8-bit parallel data from SIE at the rising edge of the clock. At this movement the transmit state machine is in Data load state. After the register is loaded, the data is sent to the other modules serially. Each bit of the register is sent to the Bit stuff module. After all the bits are serially sent to the Bit stuff module, Tx ready signal is asserted by the transmit state machine. During parallel to serial conversion data, the transmit state machine is in Data wait state.

3.2.3 EOP Generator

When TX valid signal is negated by the SIE, the transmit state machine enters into send EOP state where it enables a signal called eop_enable. This signal is checked outside the state machine for every clock. If this signal is high then the EOP pattern: two single ended zeroes (i.e., DP, DM lines contain zeroes) and a ‘J’ (i.e., a ‘1’ on DP line and a ‘0’ on DM line) is transmitted on to DP, DM lines.

3.3 The Receiver Module Specification

The receiver module has been implemented by considering the following specifications.

- When SYNC pattern is detected that should be intimated to the SIE.
- If a zero is not detected after six consecutive ‘1’s an error should be reported to the SIE.
- When EOP pattern is detected that should be intimated to the SIE.

The behavior of the Receive State Machine is illustrated in Fig-7

The assertion of Reset will force the Receive State Machine into the Reset state. The Reset state negates RXActive and RXValid. When the Reset signal is negated the Receive State Machine enters the RX Wait state and starts looking for a SYNC pattern on the USB. When a SYNC pattern is detected the state machine will enter the Strip SYNC state and assert RXActive. The length of the received SYNC pattern varies and can be up to 32 bits long. As a result, the state machine may remain in the Strip SYNC state for several byte times before capturing the first byte of data and entering the RX Data state.

After 8 bits of valid serial data is received the state machine enters the RX Data state, where the data is loaded into the RX Holding Register on the rising edge of CLK and RXValid is asserted. The SIE must clock the data off the DataOut bus on the next rising edge of CLK.

Stuffed bits are stripped from the data stream. Each time 8 stuffed bits are accumulated the state machine will enter the RX Data Wait state, negating RXValid thus skipping a byte time.

When the EOP is detected the state machine will enter the Strip EOP state and negate RXActive and RXValid. After the EOP has been stripped the Receive State Machine will reenter the RX Wait state and begin looking for the next packet.

If a Receive Error is detected, the Error State is entered and RXError is asserted. Then either the Abort 1 State is entered where RXActive, RXValid, and RXError are negated, or the Abort 2 State is entered where only RXValid, and RXError are negated. The Abort 1 State proceeds directly to the RX Wait State, while Abort 2 State proceeds to the Terminate State after an Idle bus state is detected on DP and DM. The Terminate State proceeds directly to the RX Wait State.

When the last data byte is clocked off the DataOut bus the SIE must also capture the state of the RXError signal. The description of the receiver given above is summarized below.

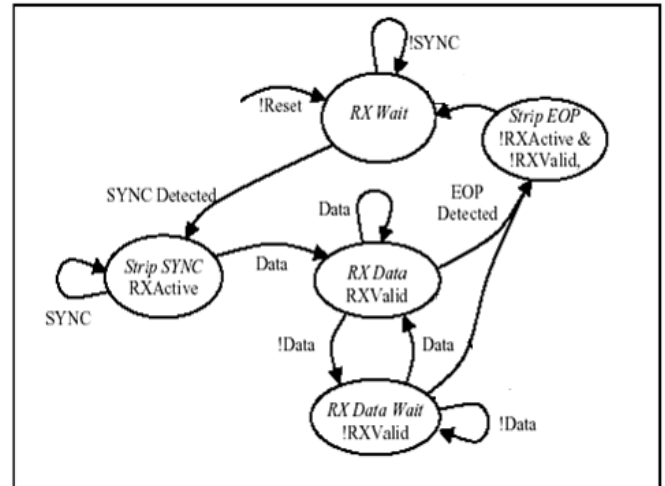


Fig7 Receive state machine.

- RXActive and RXValid are sampled on the rising edge of CLK.
- In the RX Wait state the receiver is always looking for SYNC.
- The Macrocell asserts RXActive when SYNC is detected (Strip SYNC state).
- The Macrocell negates RXActive when an EOP is detected (Strip EOP state).
- When RxActive is asserted, RXValid will be asserted if the RX Holding Register is full.
- RXValid will be negated if the RX Holding Register was not loaded during the previous byte time. This will occur if 8 stuffed bits have been accumulated.
- The SIE must be ready to consume a data byte if RXActive and RXValid are asserted (RX Data state).
- In FS mode, if a bit stuff error is detected then the Receive State Machine will negate RXActive and RXValid, and return to the RXWait state.

Fig-7shows the timing relationship between the received data (DP/DM) , RXValid, RXActive, RXError and DataOut signals.

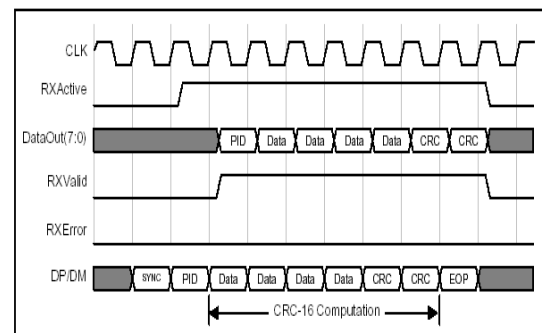


Fig-8 Receive Timing for Data Packet

3.4 The Receiver Design

The receiver module is designed by considering all the above specifications. VHDL is used to design the receiver module. The receiver module of the UTMI consists of various blocks such as SYNC detector, NRZI decoder, bit unstuff, receive shift and hold Register and EOP detector.

A receive state machine is developed by considering all the states given by USB 2.0 receive state machine. Initially the receiver is at Reset state where the reset signal is high and RX active and RX valid signals are low. If reset signal goes low the state of the receiver is changed to RX wait state where it is waiting for SYNC pattern.

3.4.1 SYNC Detector

When the receiver detects encoded SYNC pattern "01010100", the Receive state machine will enter into strip sync state where the SYNC pattern is stripped off. To detect the SYNC pattern a state machine is developed. It checks every bit for every rising edge of the clock. If the pattern is detected, a signal called sync detected is enabled. This signal is checked by the Receive state machine. If the signal is high, the Receive state machine will enter into strip sync state where RX active signal is asserted and the state machine will enter into RX data state.

3.4.2 NRZI Decoder

The received data on DP, DM lines are NRZI decoded. The NRZI Decoder simply XOR the present bit with the provisionally received bit. During NRZI decoding, the receive state machine is in RX wait state.

3.4.3 Receive Shift and Hold Register

The serial data received from the bit Unstuffer is shifted into the receive shift register. After the shift Register is full, it is held there for one clock duration and then the data is placed on to the data out bus. This 8-bit data is sampled by the SIE at the next rising edge of the clock during shifting, the receive state machine is in RX data wait state. During holding, the receive state machine is in RX data state where it asserts RX valid signal.

3.4.4 EOP Detector

A state machine is developed for EOP detection, which is invoked at every rising edge of the clock. When two single ended zeroes followed by a 'J' state is detected, it asserts a signal called eop_detect which is checked by the Receive state machine at every rising edge of the clock. When this signal is high, the receive state machine will enter in to Strip eop state where the EOP pattern is stripped off and RX active, RX valid signals are negated. At the next rising edge of the clock. The Receive state machine will enter into the RX wait state.

3.5 GENERAL IMPLEMENTATION FLOW

The generalized implementation flow diagram of the project is represented as follows.

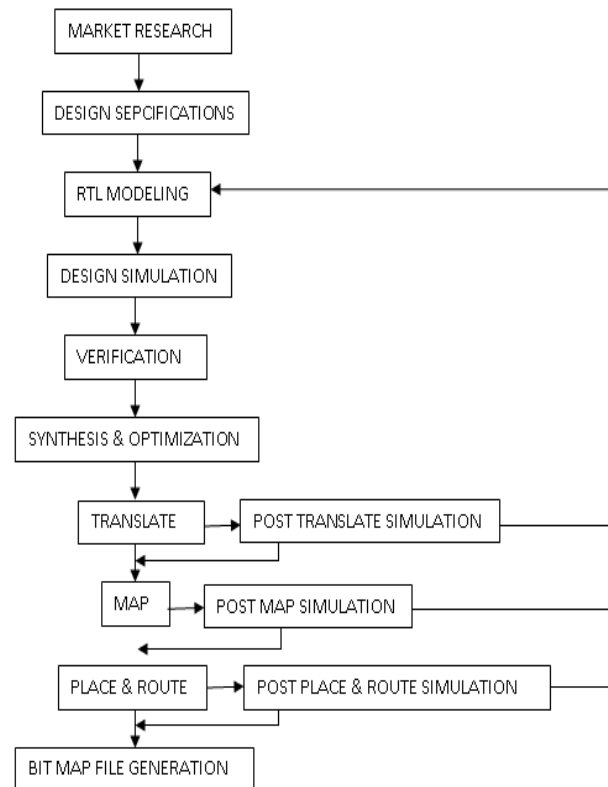


Fig-9 General Implementation Flow Diagram

Initially the market research should be carried out which covers the previous version of the design and the current requirements on the design. Based on this survey, the specification and the architecture must be identified. Then the RTL modeling should be carried out in VHDL with respect to the identified architecture. Once the RTL modeling is done, it should be simulated and verified for all the cases. The functional verification should meet the intended architecture and should pass all the test cases.

Once the functional verification is clear, the RTL model will be taken to the synthesis process. Three operations will be carried out in the synthesis process such as

- Translate
- Map
- Place and Route

The developed RTL model will be translated to the mathematical equation format which will be in the understandable format of the tool. These translated equations will be then mapped to the library that is, mapped to the hardware. Once the mapping is done, the gates were placed and routed. Before these processes, the constraints can be

given in order to optimize the design. Finally the BIT MAP file will be generated that has the design information in the binary format which will be dumped in the FPGA board.

CONCLUSIONS

The individual modules of UTMI have been designed, verified functionally using VHDL simulator.

The UTMI Transmitter is capable of converting parallel data into serial bits, performing bit stuffing and NRZI encoding.

The UTMI Receiver is capable of performing NRZI decoding bitunstuffing and converting serial bits into parallel data. The functional simulation has been successfully carried out. The design has been synthesized using FPGA technology from Xilinx. This design is targeted to the device family→spartan2, device→xc2s30, package→cs144 and speed grade→ -5. The device belongs to the Vertex-E group of FPGAs from Xilinx.

REFERENCES:

- [1]. USB 2.0 Specification, April 27, 2000
- [2]. USB 2.0 Transceiver Macrocell Interface (UTMI) Specification, version 1.05, March 29, 2001
- [3]. On-The-Go Supplement to the USB 2.0 Specification, revision 1.0, Dec 18, 2001
- [4]. UTMI+ Specification, revision 0.9, February 21, 2003
- [5]. VHDL With Example Douglas .L .Perry
- [6]. Data and Computer Communications by William Stallings
- [7]. Computer Networks by Andrew S.Tannenbaum
- [8]. www.opencore.org
- [9]. www.digitalcoredesign.org

BIOGRAPHIES:



A.VAMSHIDHAR REDDY received the M.TECH degree from Bandari Srinivas Institute Of Tech.,Chevella, Hyderabad. Currently he is working as an assoc. prof. in RRS College of Eng. And Tech., Hyderabad



A.LAXMAN received the M.TECH degree from Srinidhi Institute Of Technology, Ghatkesar, Hyderabad. Currently he is working as an asst. Prof. in MAHAVEER Inst. Of Science and Technology, Hyderabad



L.PRAKASH received the M.TECH degree from TRR College of Eng. And Tech.,Hyderabad. Currently he is working as an assoc. prof. in DVR College of Eng. And Tech.,Hyderabad



T.SATYANARAYANA received the M.TECH degree from JBIT,Moinabad, Hyderabad. Currently he is working as an assoc. prof. in DVR College of Eng. And Tech.,Hyderabad