2022 2nd Asian Conference on Innovation in Technology (ASIANCON)
Pune, India. Aug 26-28, 2022

Design & Development of micro-controlled system using VHDL with the help of UART Tx & Rx

[1] Chennagiri Rajarao Padma, [2] S. Hema Priyadarshini, [2] Nanditha H.G., [3] Dr. Pavithra G., [4] Dr. Manjunath T.C.
[1]Assistant Professor & Head of the Dept., Dept. of Medical Electronics, Dayananda Sagar College of Engg., Bangalore
[2]Assistant Prof., Dept. of Medical Electronics Engg., Dayananda Sagar College of Engineering, Bangalore
[3]Associate Professor, ECE Dept., DSCE, Bengaluru
[4] Professor & Head of the Dept., Electronics & Communication Engg., DSCE, Bengaluru
tcmanju@iitbombay.org

2022 2nd Asian Conference on Innovation in Technology (ASIANCON) | 978-1-6654-6851-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASIANCON55314.2022.9909032

***Abstract*—With respect to the work done in this research article, the designing & development of micro controlled system using VHDL with the help of UART Tx & Rx In modern day systems, it is essential for a system to be multifunctional, compact as well as price worthy. Technically speaking, a system should be optimal in its use of resources and be programmable so that the functionalities can be altered dynamically as per the requirements of the system. A software-based design can give such flexibility to the designer. This is the reason that software-based design systems are becoming increasingly popular. In today's world when new innovations come up at a rapid rate it is essential that you have tools that can provide precise modeling capabilities which can help in reducing design time and give information about all the static and dynamic parameters of your design. Software like V.H.D.L, Verilog provide a strong platform for design simulation and testing which is the basic building block in development of ASIC/FPGA based design. VHDL is called as the VHSIC based Hardware Description Language. The acronym 'VHSIC' stands for Very High-Speed Integrated Circuits, which could be utilized for explaining the structures & the behaviours of the different types of electronic systems. In general, but it is especially well-suited to describing the structure and behaviour of digital electronic hardware designs behavior & the structure of the different digital electronic h/w design like the FPGAs & the ASIC's as well as to design the traditional digital logic circuits. Verilog is another popular tool, which is available, but we have used V.H.D.L, as we are more conversant and comfortable with it. Field based programmable type of gate arrays (FPGAs) are alternative to programmable logical device (PLDs) and ASICs. FPGAs, as their name implies, have the advantage of being easily programmable. FPGAs, unlike their PLD forerunners, can (in most circumstances) be programmed numerous times, providing designers multiple opportunity to tune their circuits. To implement the project V.H.D.L. was selected as H.D.L, Xilinx ISE 6.1 as the tool for synthesis and Modelsim for simulating the VHDL Codes.**

Keywords—Microcontroller, VHDL, UART, Tx, Rx

I. Introduction to the Micro-Controllers

A micro-controller is a highly integrated chip that has all or most of the components required for a controller on a single chip. A "one-chip solution" could be used to describe the microcontroller. It usually consists of [1]:

- CPU's (central processing units)
- RAM's (Random Access Memories)
- EPROM-PROM-ROM (Erasable type of Programmable Read Only Memories)
- I-O (input & output) : serial & parallel types
- Timer
- Interrupt Controllers

Cost is kept to a minimum by only including elements relevant to the task (control). A typical microcontroller provides instructions for bit manipulation, direct, easy access to I/O, and quick, effective interrupt handling. Microcontrollers offer a "one-chip solution" that significantly lowers the number of parts and design expenses [2].

CPU : The 'computer' element of the Microcontroller is the CPU core. Its function is to execute the software that the designer has provided. It accomplishes this by utilising memory, a few registers, programme memory, and the ALU's (arithmetical logical units)

RAM's : Random Access Memories are the abbreviation for random access memories. It's a sort of a general type of memory which could hold both the datas & the programmes. The RAMs are ductile in nature, which is to convey the message that the contents of the memory will be lost if the power supplies are turned off to the RAM device. The RAMs on majority of the PCs are of several MBs or GBs or TB's. Many micro controller's have got inbuilt RAM, but it is usually not a large quantity; 256 byte is a popular size.

ROM:- ROM stands for Read-Only Memory. This is generally factory-programmed memory with predetermined value. It could be altered, but; it may be surfed majority of the times as desired. Programs and data that do not change over time are often stored in ROM. Many microcontrollers feature a large amount of read-only memory (ROMs). Likely, un-less you are going to order tens of thousands of part, the ROM will be pointless & wastes address space. The majority of people avoid controllers with a lot of the ROMs.

EPROMs : EPROMs are Erasable type of Programmable Read Only Memories. ROM stands for Read-Only Memory. This is generally factory-programmed memory with predetermined value. It can not be altered, on the other hand it could be read many a times as one can desire. Programs and data that do not change over time are often stored in ROM. Many microcontrollers feature a large amount of read-only memory (ROMs). Un-likely, unless you are going to order tens of thousands of part/s, the ROMs are pointless & wastes address space. The majority of people avoid controllers with a lot of ROM.

EEPROM:- Electrically Erasable type of Programmable Read Only Memories (EEPROMs) is a sort of ROM which can be erased in toto. EPROMs are particularly useful for a

lot of applications in computing systems & apps. For instance, EEPROMs are frequently used to store setup data on a range of household appliances. EEPROMs can be "programmed" from software, so you normally don't have to take them out of your circuit to make changes to them. This could be an significant benefit. Each byte of EEPROM is typically written in roughly 10 milliseconds.

Flash EEPROM:- Flash memory, a quicker version of the EEPROM, is another option. A byte can be programmed in a few hundred microseconds. This is a benefit over EEPROM. However, before you can programme Flash memory, you must usually delete the entire contents. Flash EEPROM is typically available in huge quantities.

Batteries type of back up static RAMs : To see that if a big amount of non volatile programme & requires more amount of data spaces, static RAM comes in handy. Static RAM has the advantage of being substantially acts fast in comparision to different sorts of non volatile type of memories, making it ideal for ab high-performance applications. It can also be written to an unlimited number of times, making it ideal for apps which save & handle vast quantities of local datas.

I/O:- These are the input output device.

UARTs : An UART (Universal type of Asynchronous Receiver Transmitters) are serial type of port adaptors that allows you to send and receive asynchronous serial data.

Analog - Digital Convertion (A-D) : It will convert a external analogue signals to digital signals (usually in relation to voltage). Microcontrollers with this capability could be utilized for environmental, instrumentation type of datas loggers, or other type of analog application. Successive type of approximations A to D converter and flash A to D converter are two forms of A/D converters that can be found.

This is the most popular type of A/D, and it's found in almost every microcontroller. The analogue to digital converter determines whether the next step is higher or lower by processing each bit individually (most significant bit comes first).

This technique has some benefits, including the fact that each conversion takes the same amount of time and that it is straightforward and frequently used (and therefore very cheap). Since it takes so long compared to other A/Ds, it is a power hog. The foundational design for the quickest A/Ds is as stated above. The flash converter functions by examining all possible voltage levels and displaying the voltage level in real time.

Utilizing comparator threshold detectors, each of which is set to a voltage of precisely '1', this is accomplished. This is greater than the detector beneath it in LSB. The benefit of this architecture, which explains why it is so quick, is that you can determine precisely what the input voltage is with just one clock cycle. The downside is that 8-bit precision requires 256 comparators, while 10-bit accuracy necessitates 1024 comparators.

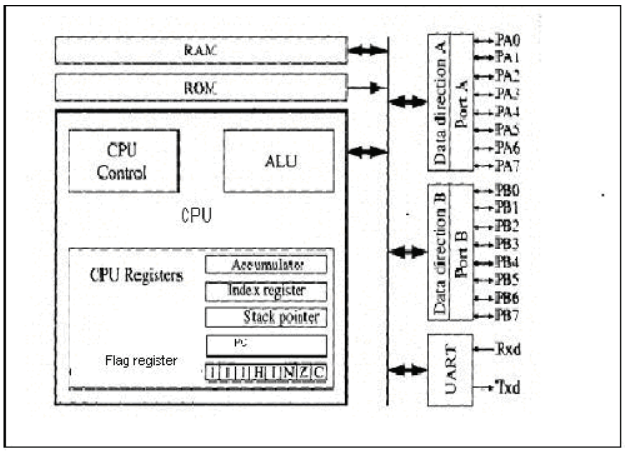


Fig. 1. Simplified block diagarm of a micro controller

The D to A (Digital to the Analog) Converters function convert a digital number to an analogue o/p. On an 8-bit / 5 Volt system, the no. '50' can be transformed to an analogue o/p of (50 / 256 * 5 Volt) = 0.9765625 V.

Pulse width modulator (PWM) is a digital-to-analog conversion technology that is frequently employed. A low-pass filter is used to create and regulate a pulse train in order to produce a voltage proportional to the duty cycle.

Pulse accumulation device : An event counter is a pulse accumulator. The pulse accumulator register, which keeps track of how many times this event has happened, is incremented with each pulse.

Watch-dog timers : A watch dog type of timer allows for a smooth recoveries after a system's failure. It could be a software that runs indefinitely or a hardware issue that prevents the programme from working properly. A hardware reset will be conducted if the application fail for resetting the watch-dog at a predefined intervals. Although the problem may still persist, the system now offers a means to recover. This is especially important for systems that are left unattended.

II. Architecture of the Micro-Controllers

A microcontroller typically contains a CPU, RAM, and ROM memory and various serial and parallel input-output interfaces, all on a single IC chip. An simple model of the microcontroller is displayed in the Fig. No. 3-1. The block diagram shows the CPU core, RAM, ROM, 2 8 bits of parallel type of input outs ports called as the port A &t he port B along with a UART, which provides a serial communications interface. In the sections that follow, we design a CPU & then integrate this CPU into system shown in Fig. 1.

Next, we describe the operation of the CPU considering the programming view points. The data bus 8 bits wide, and has an 8-bit address bus, so it is capable of addressing 256 bytes of memory. Figure below shows the register structure for the microcontroller-programming model. The accumulator (A) and the flag register are 8 bits long. The left 3 bit of the flag register are finally settled to 111 & the remaining bits are used as follows:

C (carry flag): Stores the carry or borrow that result from an arithmetic operation.

H (half carry flag): Used for BCD arithmetic

N (negative flag): Put to ‘1’ if the end result of any operation will be turned out to be negative.

Z (Zero type of flag): fixed to 1 if any operational result will become zero.

I (interrupt flag): When set to 1, prevents hardware interrupts from interrupting the processor.

III. PROGRAMMING MODEL OF MICROCONTROLLER

7	6	5	4	3	2	1	0

Accumulators (A)

7	5	6	4	3	1	2	0

Indexed Registers (X)

7	5	6	4	3	1	2	0

Stack pointer (SP)

7	5	6	4	3	1	2	0

Programmed counters (PC)

1	C	AC	O	P	N	Z	I

Programed Status Words (PSW’s)

The programmed counters (PC), which is 8 bits long, addresses the instructions in memory as they are executed. In the microcontroller, a portion of the RAM memory is reserved for the stack. This stack is used for storing subroutine return addresses, and the PC, A and flag registers made to enter the stack pointer when an interrupts is processed. The program person has no access directly to the stack. In memory, the stack decreases in height. The maximum stack size is 256 bytes because the stack pointer (SP) is 8 bits wide. SP always identifies the top-most vacant space on the stack and it is decremented after a byte is pushed onto the stack. Therefore, SP must be incremented before a byte is popped off the stack. The above table gives such infos [3].

IV. APPLICATIONS

This type of microcontroller can be widely used in simple control applications such as thermostats, appliance controllers, keyless entry systems, and so forth. These applications typically require much I/O capability and relatively little computational capability. Here low cost is much more important than high speed [4].

V. UART DESIGN

Serial data ports are found on most computers and microcontrollers, and they are used for communicating with the i-o devices, for ex., key boards & the printers. Using a modem (modulator-demodulator) connected to a serial port, serial data can be sent to and received from a distant location over phone lines (Refer Fig. No. 2). A serial communication interface known as a UART (Universal Asynchronous Receiver/Transmitter) sends and receives serial data (Universal Asynchronous Receiver-Transmitter). RxD stands for received serial data signal and TxD for transmitted serial data signal.

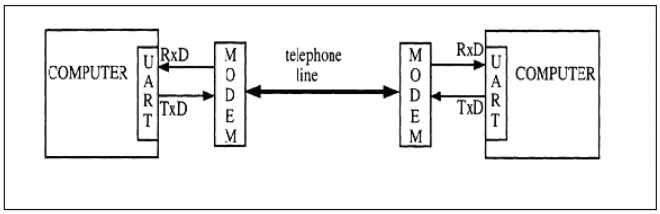


Fig. 2. Serial data transmission block diagram

The standard format for serial data transfer is shown in Figure 3. Since there is no clock line, the data (D) is sent asynchronously, one byte at a time. While no data is being sent, D stays high. The start bit, which denotes the beginning of transmission, causes D to go low for one bit. Eight data bits are transferred after the least important bit. To convey text, ASCII code is typically used. In ASCII coding, each alphanumeric character is represented by a 7-bit code. To verify for parity, utilise the eighth bit. In this illustration, the letter U is broadcast followed by a parity bit of 0, producing a total of 10101 01 1s. The BAUD rate is a common term used to describe the amount of bits sent per second [5].

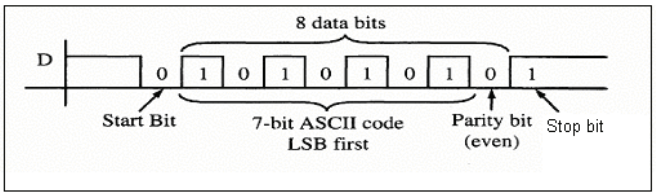


Fig. 3. Standard serial data format

When sending, the UART translates eight bits of parallel data into bit-stream which is serial in nature which will include an ‘start bit’ called as the logic '0', Eight (8) data bits—the least important bit coming first—and one or more stop bits (logic '1) The UART detects the start bits while receiving, collects the eight data bits, and, upon detecting the stop bit, turns the data into parallel form. Since no clock is transmitted, the UART must synchronise the incoming bit stream with the local clock. Right now, we're going to create a UART that is less complex. Fig. No. 3 shows the UART's connection to the 8-bit data bus. There are six 8-bit registers in use:

R-S-R Receive the shift registers

R-D-R Receive the data registers

T-D-R Transmit the data registers

T-S-R Transmit the shift registers

S-C-C-R Serial communications type of control registers

S-C-S-R Serial communication type of status registers

TABLE I. UART REGISTERS CHART

RSR	Receive the shift registers
RDR	Receive the data registers
TDR	Transmit the data registers
TSR	Transmit the shift registers
SCCR	Serial communications-controlled registers
SCSR	Serial communication type of status registers

The UART is assumed to be connected to the microcontroller's data and address bus so that the CPU can read and write to the registers in the discussion that follows. Each register in the memory-mapped RDR, TDR, SCCR & SCSR has its own address spaces. RDR, SCSR, and SCCRs can all drive the data bus through tristate buffers;

TDR and SCCR can also be loaded from the data bus. Fig. No. 4 displays a block diagram of the UART.

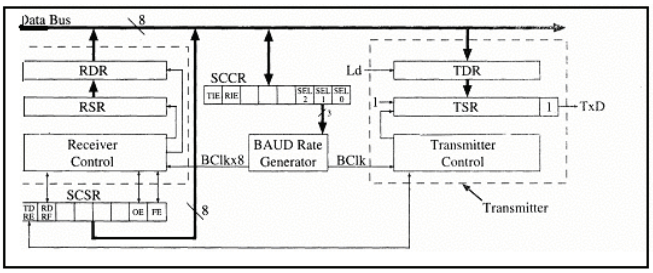


Fig. 4. UART Block Diagram

VI. UART TRANSMITTER

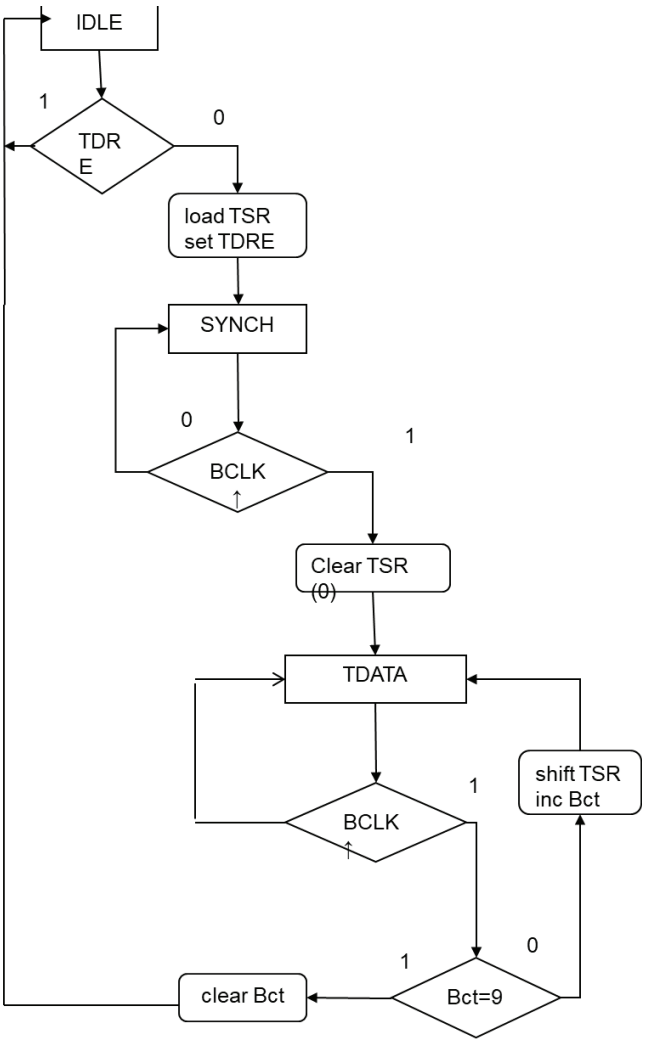


Fig. 5. State machine chart for the UART Transmitter

The BAUD type of rate generators, receiver's control, & transmitter controls are the 3 basic components of the UART, in addition to the registers. The system clock is split into two bits clocks (BC1k and BC1kX8) by the BAUD rate generator, each of which has a period that is equal to one bit of time and a frequency that is eight times the BCIk frequency. The SCSR's TDRE (transmit data register empty) bit is set when the TDR is empty. The following happens when the microcontroller is ready to transfer data [6]:

- The micro-controller loads a byte of data into 'TDR' & will clear TDRE after waiting until TDRE = "1".
- The UART sets TDRE and moves data from TDR to TSR.
- The UART transmits an initial start bit ('0') for one bit, shifts TSR to the right to communicate the eight data bits, and then produces an initial stop bit ('1').

The micro-controller system clock will be used for clock to the matching sequential machine (SM) (CLK). Until TDR is loaded and TDRE is cleared, the SM waits in the IDLE state. The low-order bit of the TSR is cleared by the SM before transmitting a "0" for one bit time on the rising edge of the bit clock (BCIk) in the SYNCH state. Each time BCIk is found to be in the TDATA state, TSR is shifted to the right to transmit the subsequent data bit, and the bit counter (Bct) is increased. When Bct = 9, eight data bits plus a stop bit have been transmitted. Bct is then cleared, and the SM goes back to IDLE. The SM chart in Fig. No. 5 is the foundation for the VHDL code for the UART transmitter.

Then, the TDR and TSR registers, as well as the transmit control, are all found in the transmitter. It connects to the data bus and TDRE (DBUS). The first process is the combinational network, which generates the nextstate and control signals. The second process updates the registers as the clock rises. The signal Bclk rising is '1' for one system clock time after the rising edge of Bclk. To cause Bclk to rise, it is held in a flip-flop known as Bclk Dlayed. If the value of Bclk is "1" right now and "0" was the prior value (stored in Bclk Delayed), then "1" is the value of Bclk rising. As a result, Bclk rising = Bclk, not Bclk Dlayed [7].

VII. UART RECEIVER

The following is how the UART receiver works:

- When a start bit is detected, the UART serially will read the bits remained & will shift all those in to the B S R.
- Then, RSR is loaded into the RDR once the bits of datas are stopped using the stop bits that would be received & the Receives Data Registers Full (RDRF) flags in the SCSR's will beset.
- Note that in case the RDRF flag will be set, the micro-controller reads the RDR and clears the flag.

On RxD, the bit stream is not synced taking into consideration the logic bit block set clock called as the 'Bclk'. If RxD moved close to the clock edge, it would be difficult for us to read it at the rising edge of Bclk. It's possible that we'll encounter issues with setup and hold times. If the incoming signal's bit rate deviated by a little amount from Bclk, we might end up reading some bits at the time which is fully wrong. To circumvent these problems, we'll sample RxD eight times throughout each bit period. We'll sample the leading edge of BclkX8. The arrows in Fig. 6 represent the rising edge of BclkX8. The bit value should be read in the middle of each bit time for maximum accuracy. When RxD first reaches zero, we'll wait four BclkX8 periods and be very close to the start bit. We'll then wait for eight additional BclkX8 cycles, which should get us to the middle of the first data bit. Prior to reaching the stop bit, we continue reading every 8 BclkX8 clocks [8].

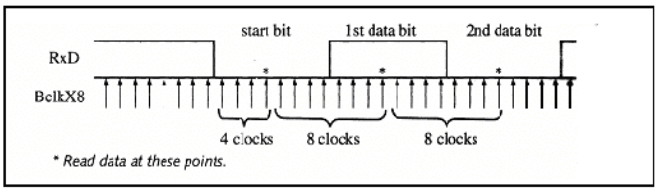
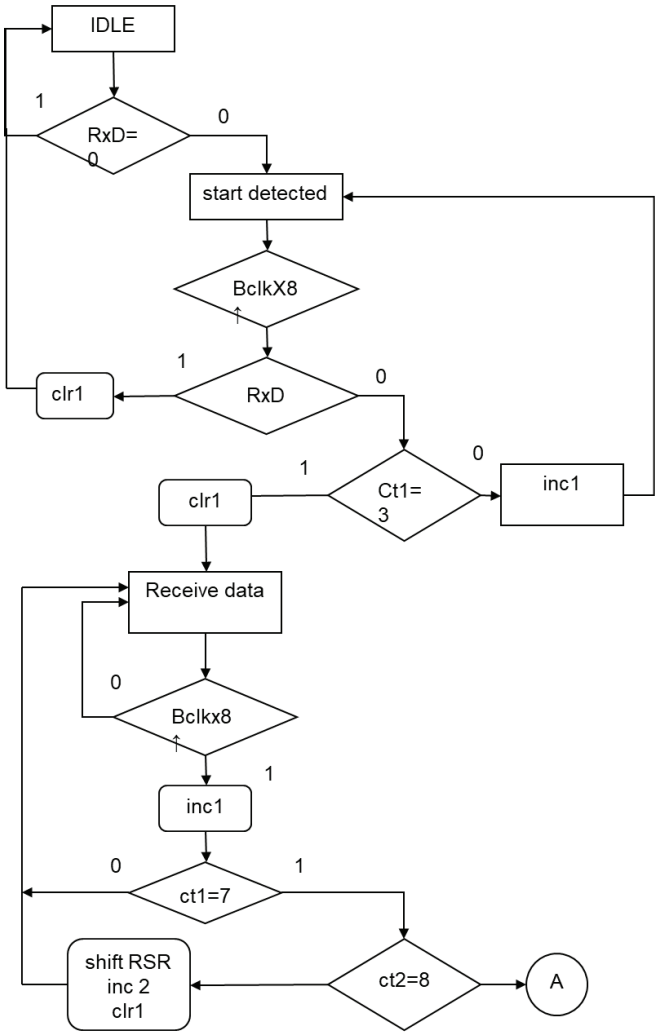


Fig. 6. Sampling RxD with BclkX8 block

An state machine chart for the UART based receivers are shown in Fig. 7. There are two counters in use. The number of BclkX8 clocks is counted by Ct1. After the start bit, 'Ct2' count the no. of received bits. The SM waits for the start bit (RxD = '0') in the IDLE state before moving to state of detection at the starting levels. The state machine waits for the rising edge of BclkX8 before sampling RxD once more (BclkX8). Since the start bit for eight BclkX8 clocks should be "0," we should read that value. The SM increases Ct1 because it is still zero and waits for BclkX8. If RxD = '1', the SM resets to the IDLE state and clears Ct1. which is an error situation. Otherwise, the SM continues to loop. When RxD is used [9].



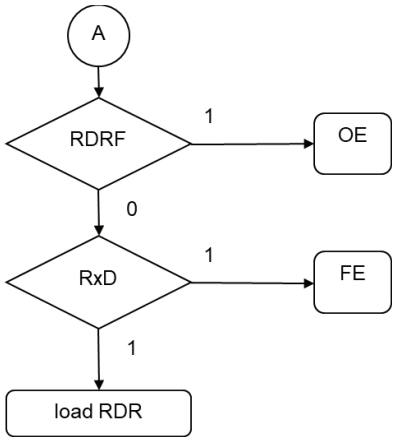


Fig. 7. Flow chart for design of the UART receiver

The state machine will increment the Ct1 following each edge which is rising w.r.t. the Bclk x 8 for the state considered. The eighth clock checks that Ct1 = 7 and Ct2. RxD is placed into RSR, Ct2 is increased, and Ct1 is cleared if the value is less than 8. We have read all eight bits and are currently in the midst of the stop bit if Ct2 = 8. An overflow error occurs if RDRF = 1 because the microcontroller has not yet read the last data byte it received. The new data is disregarded since the OE flag in the status register is set. If RxD='0,' the status register's frame error (FE) flag has been set and the stop bit has been improperly recognised.

TABLE II. BAUD RATES SELECTION CONCEPTS

Selected Bits	Baud Rate Type (Bclk)
000	38462
001	19231
010	9615
011	4808
100	2404
101	1202
110	601
111	300.5

In all circumstances, the RDRF is programmed to indicate that the receive operation is complete and the counters have been cleared. The SM chart in Figure 7 is the foundation for the VHDL code for the UART receiver. The receiver contains the RDR and RSR registers as well as the receiver control. Data can be driven onto the data bus by RDR, and SCSR serves as the control interface. The first process is the combinational network, which generates the nextstate and control signals. The second process updates the registers as the clock rises. The signal BclkX8 rising is '1' for one system clock period after the rising edge of BclkX8. Similar to how Bclk rising is generated, BclkX8 rising is also made [10].

VIII. BAUD RATE GENERATOR

We'll then create a programmable BAUD rate generator. The SCCR has three bits that are used to pick one of eight BAUD rates. We'll use an 8 MHz system clock and BAUD rates of 300,600, 1200, 2400, 4800,9600, 19200, and 38400. 38400 x 8 = 307200 is the maximum BclkX8 frequency required. We need to divide 8 MHz by 26.04 to get this frequency. Because we can only divided by the integer types so that we accept it either by a tiny Baud rated inaccuracy or to reduce the system's frequency of the clock up to 7.9 Mhz

for the compensation purposes. The frequencies generated, assuming an 8-MHz clock, are listed in the table below:

The BAUD rated generators are depicted in Figure 8 as a block diagram. An counter first divide the 8 MHz system clock by a value of 13. The o/p of the said counter will be transmitted to the 8 bit BC. The outputs of the flip-flops in this counter are divided by 2, divided by 4, and divided by 256. One of these outputs is picked by the multiplexer. The MUX select inputs are provided by the lower three bits of the SCCR. BclkX8 is then multiplied by 8 to produce Bclk so that the the MUX output sets in[12].

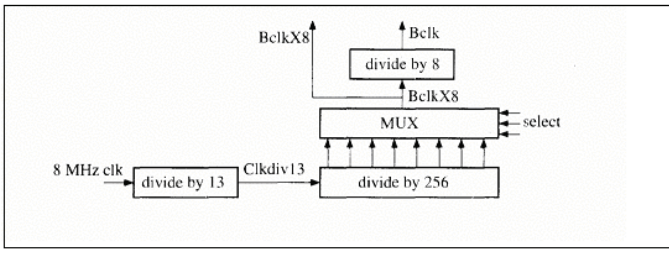


Fig. 8. Baud Rate Generator

The VHDL code for the BAUD rate generator is created and executed. The divide-by-13 counter is increased by the first process on the rising edge of the system clock. The second process increases the divide-by-256 counter at the rising edge of Clkdivl3. BclkX8, the output of the MUX, is produced by a concurrent statement. The third procedure increases the divide-by-8 counter to produce Bclk on the rising edge of BclkX8.

IX. Complete Design of UART

To complete the UART design, we must link the three designed components, connect them to the control and status registers, and add the interrupt generation logic and bus interface. The SCI IRQ interrupt signal is transmitted to the CPU whenever the UART receiver or transmitter needs attention. When the RIE (receive interrupt enable) is set in SCCR, SCI IRQ is triggered whenever RDRE or OE is '1'. Anytime TDRE is set to "1" and TIE (transmit interrupt enable) is set in SCCR, SCI IRQ is generated. The microcontroller's address and data buses are connected to the UART with SCIsel = '1', enabling the CPU to read and write to the UART registers. When the UART is not chosen for reading, the data bus is driven to high-Z. The R W signal and the final two bits of the address (ADDR2) are utilised to choose the registers as follows [11]:

TABLE III. Control Signal Generation Chart

ADDR2	R_W	ACTION
00	1	DBUS←RDR
00	0	TDR ← DBUS
01	1	DBUS←SCSR
01	0	DBUS←hi-Z
1-	1	DBUS←SCCR
1-	0	SCCR←DBUS

Then, microcontroller has two parallel ports A and B, each of 8 bits length. Each pin of the ports is bidirectional and can be individually configured as i/p or o/p port. The figure below is depicted the block diagrammatic representation of the parallel port A (Fig. 9)

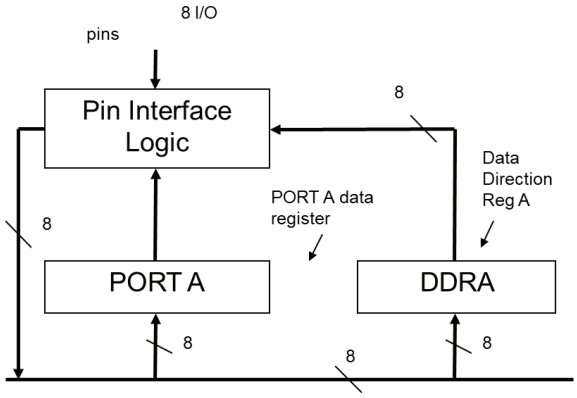


Fig. 9. Parallel port block diagaram

Description of parallel port: The Port_sel pin is used to select one of the two ports. When the value of Port_sel is '1', port A is selected; if the value of Port_sel is'0' then port B is selected. The signal ADDR0 is used to select the data direction register (DDRA) or the port A. If the value of ADDR0 is '1' then DDRA is selected; if the value of ADDR0 is '0' then port A is selected. The control signal R_W is used to read/write into the port A or the DDRA register. The flow chart for the working of parallel port is shown in Fig. 10.

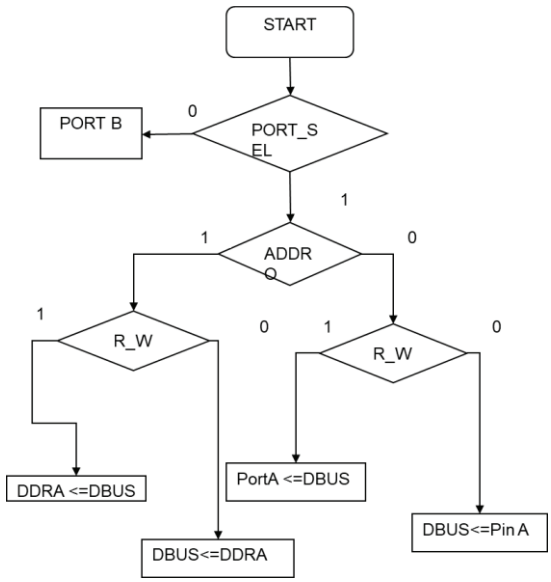


Fig. 10. Flow chart for the parallel ports

Condition 1(for configuring the pins of the port using the DDRA register): When a port's corresponding pin is set to "1" in the DDRA register, it is configured as an output pin, and when it is set to "0," it is configured as an input pin. In DBUS, the desired data is initially loaded. When the signal loadDDRA is set to "1," this data is transmitted to the DDRA register. This condition is achieved when:

1) Ports_sel = "1"

2) ADDR0 = "1"

3) R_W = "1"

The datas are then transfered starting from DATA type of bus to the DDRA. Based on the data in DDRA the pins of the port are configured as input/output.

Condition 2 (for sending data through the parallel port): Once the pins of the port are configured to be output, we need to send the data through the port. The type of datas or the informations which is to be transmitted is then loaded into the data bus. This data is loaded into the port when the following conditions are satisfied:

1) Port_sels = "1"
2) ADDR0 = "0"
3) R_W = "1"

The control signal load PORTA sets to '1' on achieving the above conditions and the data in the data bus is loaded onto the port.

Condition 3 (for reading from-to the DDRA register)s – for reading the data from the DDRA registers the control signal readDDRA should be '1'. This control signal is set to '1' when:

1) Port_sels = "1"
2) ADDRO = "0"
3) R_W = "1"

Thus, on achieving the following conditions, readDDRA is set to '1'and the data in the DDRA registers will be synched with the data bus for getting the information.

Condition 4 (for reading from the port): To read data from the port A the control signal readPORTA should be '1'. This control signal is set to '1' when:

1) Port_sels = "1:
2) ADDR0 = "0"
3) R_W = "0"

Thus, on achieving the following conditions, readPORTA is set to '1'and the data on the pins of the port A is transferred to the data bus. Appendix shows the VHDL code for the parallel port. Concurrent statements generate the control signals for reading and writing to the registers. Port_Sel is '1 when the port is selected for reading or writing. A single address bit, ADDRO, selects either the PORTA or DDRA register. The generate statement labeled Port-bits generates the logic associated with each bit in the port. Then, the processes will be updated onto the port so that it registers all the types of informations that occurs on the clock's rising edge.

X. DESIGNING OF THE MICROCONTROLLER CPU

Here, w.r.t. this section, a MC will be used along with the CPU. We have omitted several of the instructions given in Table 1-ito reduce the complexity. We determine the cycle-by-cycle operations necessary to implement different types of instructions and addressing modes. Then we write behavioral level VHDL code and verify that our design meets specifications. After constructing block diagrams for the CPU, we rewrite the VHDL code in terms of register transfers and control signals. After simulating the CPU, we synthesize it from the VHDL code to fit into an FPGA and a CPLD. The next step in designing the CPU is to determine what actions should take place during each clock cycle. The internal clock period is the same as the memory cycle time. That is, during one clock cycle we can read or write a byte to memory, or we can complete an internal CPU operation such as addition. Each instruction takes from two to ten clock cycles to execute, depending on its complexity. The first cycle of every instruction is used to fetch the opcode from memory. At the start of the cycle, the program counter is pointing to the first byte of an instruction in memory, which is the opcode. The memory returns the opcode on the data bus as the PC exits on the address bus. At the end of the cycle, the opcode is loaded into the opcode register and the program counter is incremented. We designate these actions as follows:

$$\text{Opcode} \leftarrow \text{mem(PC)};$$

$$\text{PC} \leftarrow \text{PC} + 1;$$

where, mem is an array of bytes that represents the memory.

XI. ARCHITECTURE DETAILS OF THE SPARTRAN – 3 TYPES OF ARRAYS

Figure 1 shows the Spartran-2E user enabled programmed gate based arrays, which is made up of five key adjustable elements:

- CLBs offer the necessary building blocks for most logic constructions.
- IOBs serve as the interface for the internal logic and packaging pins.
- A flexible multi-level interconnect system with dedicated block RAM memory of 4096 bits per unit; Clock DLLs for clock-distribution delay correction and clock domain management.

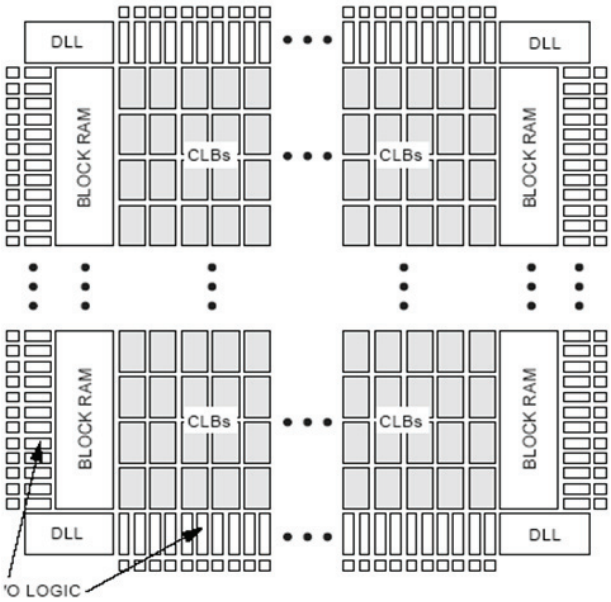


Fig. 11. Basic SPARTRAN 3 Block Diagram

The CLBs constitute the central logic structure, as shown in Figure 11, providing easy access to all routing and support structures. To facilitate quick and easy signal routing on and off the chip, the IOBs are positioned all around the logic and memory modules. Values kept in static

memory cells govern all movable logic components and interconnect resources. When the device turns on, these values are placed into the memory cells, and they can be reloaded if necessary to adjust the device's function. The next sections will go through each of these elements in depth.

XII. Input/Output Block

Figure 12 shows the Spartan-IIE IOB, which has inputs and outputs that support a wide range of I/O signalling standards. These high-speed inputs and outputs can handle a wide range of cutting-edge memory and bus interfaces. Table 1 shows which standards are supported, as well as the requisite reference, output, and termination voltages to meet the standard. The three IOB registers can be used as level-sensitive latches or as edge-triggered D-type flip flops. Each IOB contains a clock signal (CLK) that is shared by all three registers, as well as separate Clock Enable (CE) signals for each register..

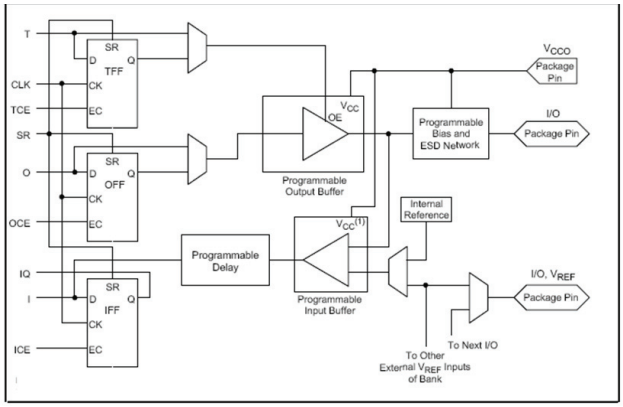


Fig. 12. Basic input – output block diagram

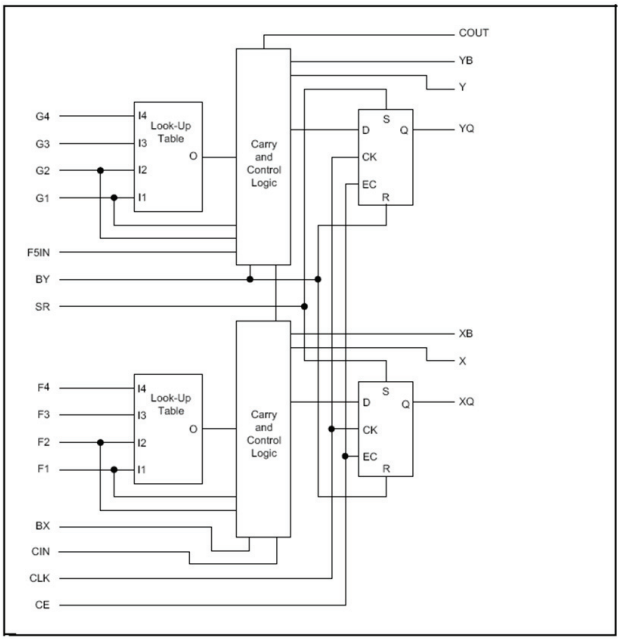


Fig. 13. CLB Slice block diagram

Figure 12 shows the Spartan-IIE IOB, which has input's & output's which will support a large range of the input and output signal standards, which will initiate the high speed handling of the inputs and outputs. These high-speed inputs and outputs can handle a wide range of cutting-edge memory and bus interfaces. The supported standards are listed in Table 1, together with the necessary reference, output, and termination voltages to comply with each standard. The three IOB registers can function as edge-triggered D-type flip flops or level-sensitive latches. Clock signals (CLK) are present in each IOB and are shared by the 3 types of registers, as well as separate Clock Enable (CE) signals for each register. Although the weak-keeper and pull-down resistors are disabled, inputs can still be pulled up if necessary. The worldwide activation of pull-up resistors before setup is controlled by the configuration mode pins. If the pull-up resistors are not triggered, all pins will be floating. Therefore, external pull-up or pull-down resistors are required on pins that must reach a particular logic level before configuration. All pads are shielded from damage by electrostatic discharge (ESD) and overvoltage transients. Following configuration for the LVTTL, PCI, HSTL, SSTL, CTT, and AGP standards, clamping diodes are connected to the VCCO.

XIII. Configuralbe in Logic Block-Sets

The logic ell is the Spartan-IIE CLB's fundamental structural component (LC). A storage element, carry logic, and a 4-input function generator make up an LC. The CLB output or the D input of the flip-flop is driven by the function generator output in each LC. For each Spartan-IIE CLB, four LCs are arranged in two slices; Figure 13 shows one slice. The Spartan-IIE CLB includes logic that combines function generators to produce functions with five or six inputs in addition to the conventional four LCs. The Fig. 14 gives the F5 F6 Multiplexers block diagram.

XIV. Looking Up With the Look-UP Table

Spartan-IIE functions generator could be implemented in look up tables using 4 inputs called as the LUTs. In addition to functioning as a function generator, each LUT can deliver a 16 x 1-bit synchronous RAM. By integrating the two LUTs within a slice, a 16 x 2-bit, 32 x 1-bit, or 16 x 1-bit dual-port synchronous RAM can be produced. The Spartan-IIE LUT may additionally offer a 16-bit shift register, making it appropriate for burst-mode or high-speed data recording. This mode can also be used to store data in applications like digital signal processing..

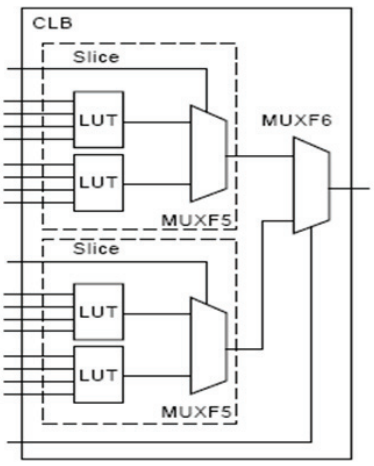


Fig. 14. F5 F6 Multiplexers block diagram

XV. Storage Elements

The Spartan-IIE slice's storage elements can be set as either level-sensitive latches or edge-triggered D-type flip-flops. Slice inputs, which omit function generators, or function generators inside the slice can both control the D inputs directly. In addition to the clock and clock enable signals, each slice also has synchronous set and reset signals (SR and BY). The startup state specified in the configuration is compelled into a storage element by SR. By shifting it to the opposing side. Additionally, these signals can be configured to operate asynchronously. The two flip-flops in the slice share all control signals and they can all be inverted separately..

XVI. Addition Based Logic Levels

The function generator outputs are combined by the F5 multiplexer in each slice (Fig. 14). Any 5-input function, a 4:1 multiplexer, or a combination of functions with up to 9 inputs can be utilised with this configuration to construct a function generator. To combine the outputs of all four function generators in the CLB, the F6 multiplexer chooses one of the two F5-multiplexer outputs. This allows any 6-input function, an 8:1 multiplexer, or a specified function with up to 19 inputs to be implemented..

XVII. Arithmetic Logic

Dedicated carry logic offers quick arithmetic carry capability for high-speed arithmetic operations. Two independent carry chains are supported by the Spartan-IIE CLB, one for each slice. Per CLB, the carry chains have a two-bit height. An LC can build a 1-bit complete adder using an XOR gate in the arithmetic logic. A separate AND gate also improves the effectiveness of multiplyer implementation. The dedicated type of carry paths can also be utilised to build extensive logic functions by cascading function generators..

XVIII. BUFTs

2 three-state's driver (BUFTs) could be driven on a chip type of bus in each Spartan-IIE CLB. The on-chip buses can also be driven by the IOBs on the left and right sides. Each Spartan-IIE BUFT has its own three-state control pin as well as an input pin. The active-Low enable on the three-state controlled pins will be actively low ('T'). The net is High when all BUFTs on it are disabled. Unless you want to simulate something, there's no need to create a pull-up. Contention will not occur if many BUFTs are driven onto the same network at the same time. When using both high and low speeds, the overall outcome will be poor..

XIX. Clock's Distribution Types

The Spartan-IIE series offers high-speed, low-skew clock distribution through the important global routing resources mentioned above. Figure 15 depicts a typical clock distribution net. Two global buffers are located in the device's top centre and two are located in the device's bottom centre. The four basic global nets are driven by them, and any clock pin is driven by them. Each of the four global buffers has its own dedicated clock pads..

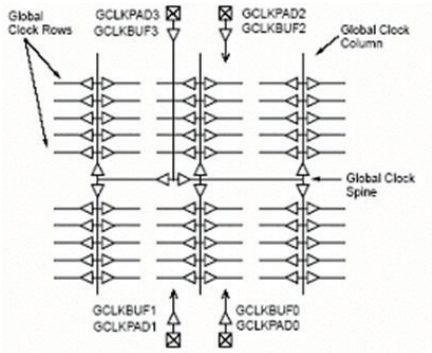


Fig. 15. Clock distributions

XX. Delay-Locked LOOPS (DLL)

Each global clock input buffer is connected to a completely digital Delay-Locked Loop (DLL), which might lessen skew between the device's internal clock input pins and the clock input pad. Two global clock networks can be driven by each DLL. The DLL automatically adjusts a clock delay element and monitors input and distributed clocks (Figure 16). Clock edges reach internal flip-flops one clock period after they reach the input thanks to an additional delay. This closed-loop gadget efficiently reduces clock-distribution delay by ensuring that clock edges arrive at internal flip-flops in synchrony with clock edges coming at the input...

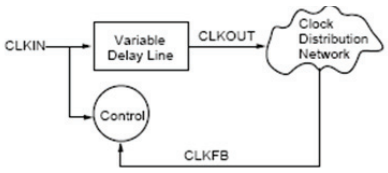


Fig. 16. Delay lock loops block diagaram

XXI. VHDL

A programming language called VHDL was developed expressly for defining how digital systems behave. There are several features in VHDL that are helpful for defining the behaviour of electronic components, from straightforward logic gates to complete microprocessors and specialised devices. Electrical characteristics of circuit behaviour (such as signal rise and fall times, delays through gates, and functional activity) can be precisely specified using VHDL's features. The VHDL simulation models that arise can then be used as building blocks in bigger circuits for simulation (using schematics, block diagrams, or system-level VHDL specifications). VHDL is also a general-purpose programming language, allowing complicated design concepts to be expressed as computer programmes in the same way that high-level programming languages do. The behaviour of complicated electronic circuits can be represented in VHDL and applied to automatic circuit synthesis or system simulation in a design system. VHDL, like Pascal, C, and C++, provides a rich set of control and data representation features that are beneficial for structured design methodologies. VHDL, unlike these other programming languages, has characteristics that allow for the description of concurrent events. This is significant since VHDL-based hardware is inherently concurrent in its functioning. VHDL is used to capture the performance specification for a circuit in the

form of a test bench, which is one of the most essential uses. Test benches are VHDL descriptions of circuit stimuli and predicted outputs that are used to check a circuit's behaviour. Any VHDL project should include test benches, which should be built in conjunction with other circuit specifications. A computer language called VHDL can be used to abstractly describe many types of electronic equipment. Algorithm, register transfer level (RTL), and gate level are the three layers of abstraction that must be recognised and understood when using VHDL to create FPGAs and ASICs. The synthesis input is RTL, the synthesis output is gate level, and algorithms cannot be synthesised. It is possible to comprehend the difference in temporality between different levels of abstraction...

A. Algorithms

A pure algorithm consists of a set of steps that must be followed in order to finish a task. A pure algorithm has no clock or precise delay. Some characteristics of timing can be inferred from the method's partial operation sequence. Algorithmic VHDL code is an input for some synthesis tools (behavioural synthesis). Even with such tools, the VHDL input could still need to be artificially constrained, perhaps by adding a "algorithm" clock, which may then be synced with operations in the VHDL code..

B. RTL

An explicit clock is included in an RTL description. All processes are timed to happen in particular clock's cycle and there will no precise delay. In this regard, commercially available synthesis tools provide some flexibility. It is not necessary, although it may be preferable, to have a single global clock. Retiming is another feature that permits processes to be rescheduled over clocks cycle, even though they are not upto the extent of the behavioural based synthesis designs tools.

C. Gate

A network of gates and registers that are instanced from a technology library and individually carry technology-specific delay data make up a gate level description...

D. Writing VHDL for Synthesis

The RTL abstraction level is highlighted. Given the state of the art of today's synthesis tools, this is the best level of abstraction for designing hardware. This is the best degree of abstraction for developing hardware given the state-of-the-art of today's synthesis tools. Keep in mind that we want to avoid dealing with implementation-related problems in hardware design, therefore the gate level is inadequate for defining hardware. We want to abstract down to the hardware's specifications, or what it does, not how it does it. The majority of commercially accessible synthesis tools, on the other hand, are unable to build hardware from a description at this level due to the algorithmic level being too high. The RTL level of abstraction will eventually be seen as the "dirty" method to write VHDL for hardware and algorithmic (sometimes referred to as behavioural) VHDL will become the standard as synthesis technology advances. The best outcomes will be achieved until then by using VHDL coding at RTL as an input to a synthesis tool..

E. Design Flow using VHDL

In a real-world design environment, each of the procedures described in the following sections can be broken down into a number of smaller phases, and different parts of the design flow can be iterated as errors are found...

F. Systems-levels Verification's Process

VHDL may be used to model and simulate a portion of the entire system, which includes one or more devices, in the first stage. This could be a completely functional description of the system that enables verification of the FPGA/ASIC specification before beginning detailed design. Alternately, this could be a partial description that abstracts away some system characteristics, such a performance model to identify bottlenecks in system performance..

G. R-T-L Designing Types & the Test Benches Creation

When the general system architecture and partitioning are established, each FPGA/detailed ASIC's design can start. The first step in this process is to capture the design in VHDL at the register transfer level as well as a collection of test cases. These two complimentary tasks are occasionally carried out separately by various design teams to guarantee that the specification is accurately interpreted. If automatic logic synthesis is to be utilised, the RTL VHDL needs to be synthesizable. The quality of the final FPGA/ASIC depends on the coverage of these test cases, which is a challenging undertaking that calls for disciplined thinking and a lot of engineering ingenuity..

H. RTL's verification process

The functionality of the RTL VHDL is then verified against the specification using simulation. Experience has shown that the best approach to take advantage of this speedup is to perform more simulation, not less, as RTL simulation is often one or two orders of magnitude faster than gate level simulation. Developing and simulating VHDL at and above the register transfer level typically takes up 70–80% of the design cycle, with the remaining 20–30% going into synthesising and testing the gates...

I. Look Ahead's Synthesizing Design Circuitry

The main synthesis production run won't start until the functional simulation is finished, despite some exploration composition being done early in the design process to provide accurate speed and area data to aid in the evaluation of architectural decisions and to make sure the engineer understands how the VHDL will be synthesised. Spending a lot of time and energy on synthesis before the design's usefulness has been established is wasteful...

XXII. Conclusions

A microcontroller was designed using VHDL concepts. VHDL Codes were written or UART transmitter, receivers, CPU, accumulators. It was simulated and the results were observed.

References

[1] Ayala Kenneth-The 8051 microcontroller: Architecture, Programming and Applications

[2] Bhasker J. -*VHDL Primer*

[3] Brown S. and Vranesic Z.Fundamentals of Digital Logic Design with VHDL Design
[4] Hayes-Computer Architecture and Organization, *Text Book.*
[5] Perry D., VHDL. *McGraw-Hill Publications* – 1999
[6] Dr. Suhasini V., Pavithra G., Dr. T.C.Manjunath, "Microcontroller based control of devices using a sophisticated control system", *Third International Conference on Recent Advances and Challenges in Engineering and Management* (*RACEM-2016*), Technically co-sponsored by ISTE, Govt. of Maharashtra, CSI, Bio-medical Engineering Society of India, Global Advanced Research Publication, Vidyalankar Institute of Technology (VIT), Wadala, Mumbai, Maharashtra, India, Session 4B, Embedded and Control Systems, 02.30-03.30 pm (Venue: M-101), paper no. 10/142, pp. 23, 22 Dec. 2016.
[7] Dr. T.C.Manjunath, Pavithra G., Dr. Dharmanna Lamani, "Microcontroller based control of devices using a sophisticated control system", *Proc. of the IFERP's Int. Conf. on Chip*, *Circuitry*, *Current*, *Coding*, *Combustion & Composites* (i7c-2016), paper id 81, pg. 87, ISBN 9788192958026, organized by IFERP & Shirdi Sai College of Engg., Bangalore, Karnataka, India, & associated with Technocrate Group (Technocrate Research & Development Association), Conference Alerts, ECA, IERD, ISER, IIAR, pp. 87 (abstract booklet), 10-11 Nov. 2016.
[8] Pavithra G., Dr. T.C.Manjunath, Dr. Dharmanna Lamani, "Hardware implementation of Glaucoma using A PIC Micro-controller – A novel concept for a normal case of the eye disease", *IEEE Int. Conf. on Current Trends in Computer, Electrical, Electronics & Communica*tion (*ICCTCEEC-2017*), IEEE Bangalore Section & CPGC, Vidyavardhaka College of Engineering, Mysore, pp. 1104 – 1109, 8-9 Sept. 2017,
[9] Dr. T.C.Manjunath, Pavithra G., Rashmi Jagadisha, "Design & development of an efficient path planning mechanism for a Micro-Robot", *IEAE's International Conference on Emerging Trends in Science & Engineering* (*ICETSE-2017*), Paper id ICETSE-219, Associated by Institute for Exploring Advances in Engg. (IEAE), Coorg Institute of Technology, Ponnampet, Karnataka, India, Sl. No. 42, pp. 227-234, 11-12 May 2017.
[10] Pavithra G., "Micro-controller based interface to analyze data errors in the ship networks", *National Conference on New Advances in Computer & Communications* (*NACC-13*), Bangalore, Karnataka in association with ISTE, Paper id 12, pp. 35-39, 19 Apr. 2013
[11] Dr. T.C.Manjunath, Pavithra G., "Control of different types of equipments using micro- controlling devices", *Proc. of Bhilai Inst. of Tech. Conference-2015* (BITCON-2015), *National Conf. on Innovative Advances in Control & Power Engg.*, organized by Dept. of Electrical Engg., BIT-Bhilai in association with ISTE, Institution of Engineers IE(I) & conducted at Bhilai Inst. of Tech., Durgapur, Chhattisgarh, ID No. 7, Tech. Session-III, Day-2, pp. 38-46 (hard copy), pp. 841-849 in *e*-proceedings, 20-21 Feb. 2015.
[12] Dr. Manjunath *et.al.*, "Microcontrollers", *Text Book-Subhas Stores*, 2019.