**HARDWARE PROGRAMMING**:-

It has 5 modules.

**AHBUART.v**:- Main module instantiated by the AHB interface when UART peripheral has to be used.

**Baudgen.v**:- Since UART is asynchronous both Tx and Rx will be agreed to operate at the same rate. Generates a baud rate of 19.2Kbps from a processor frequency of 50 MHz.

**Fifo.v**:- Contains a first in first out register which holds the data to be sent to Tx and the data received from Rx as the operating speed of processor and UART doesn’t match. Has few status flags to indicate whether the register is empty or full.

**uart\_rx.v**:- Input to this module is the data received from the serial port. It ignores the start bit and then stores 8 bit data and neglects the stop bit.

**uart\_tx.v**:- Input to this module is from FIFO register. A data packet will be created in this module. It adds one active low start bit to a 8 bit data followed by one active high stop bit. This packet will be sent to serial port. Start and stop bit have been added to indicate the starting and ending of transmission.

**CADENCE ASIC DESIGN FLOW:-**

Netlist Generation: A netlist will be created at first for the layout synthesis using the available technology and library files.

**Physical Design:-**

1. **Design Import**:

Import the netlist (.v file)

Setup the max and min timing constraints, choose the supply lines (VDD and VSS) and the delay corners.

Add the constraints file (.sdc file)

1. **Floor planning:** Floor plan defines the actual form, the layout will take, the global and detailed routing grids, the rows to host the core cells and the I/O pad cells, the area for power rings, the (pre)placement of blocks/macros, and the location of the corner cells.
2. **Power Planning:** Choosing the position for the supply rails, the metals to be used and horizontal or vertical position. Power rings will get created.
3. **Placement and route**: The first step, placement, involves deciding where to place all [logic](https://en.wikipedia.org/wiki/Logic) elements in a generally limited amount of space. This is followed by routing, which decides the exact design of all the wires needed to connect the placed components.
4. **Timing analysis**: Worst case and best case timing constraints will be checked in this process. Contains both Pre CTS and Post CTS timing analysis.CTS is for creating the balanced clock path for all the circuits in the design.
5. **Routing the design**: This step generates all the wires that are required to connect the cells as defined in the imported Verilog netlist.
6. **RC Extraction:** To obtain the RC values of the interconnects which contributes to the significant part of delay in the circuit.

**FPGA IMPLEMENTATION**

The FPGA board used was **BASYS 3, Artix 7** and with the help of Xilinx Vivado design suite the UART module of ARM Cortex M0 processor was implemented. The following steps were followed:

1. **Modifications in the top level module**: The 3 clocks of the processor were firstly assigned to the common clock. CLK was made as the only input and the rest inputs were made into wires and outputs as registers. This was done since the inputs and outputs are provided via VIO (Virtual Input Output) setup from the IP Catalog.
2. **Instantiation of VIO and ILA** (Integrated Logic Analyzer): Using IP Catalog the VIO and ILA was given the information about the inputs and output like width in terms of bits. Input to the module becomes output of the VIO and output of the module becomes input to the VIO. After running both VIO and ILA, the generated instantiation was placed in the top level module.
3. **Elaboration** of the design
4. **Synthesis**: Synthesize the design to create a schematic. Open the synthesized design and review timing constraint definition, I/O planning and design analysis.
5. **Implementation:** Open the implemented design to analyze timing, power, resource utilization, routing, and cross-probing.
6. **Generate Bit stream** and program the device
7. **VIO and ILA**: Provide inputs via VIO. Check the outputs using ILA waveforms. Verify the functionality in hardware using the Basys 3 board.