



## VLSI DESIGN PROJECTS/INTERNSHIP

### College Name

→SRM Institute of Science and Technology.

### Company Name

→Vyorius.

### Domain

→VLSI design.

## Project topics

1. Basic circuits written in Verilog/VHDL, simulated and implemented on the FPGA
2. UART communication to print a single character
3. FSM Designs: Mealy & Moore Machines and Up Down Counter

## Hardware and software used

### **Hardware,**

Mimas A7 Mini FPGA Development Board

Mimas A7 Mini is an easy-to-use FPGA Development board featuring Artix 7 FPGA (XC7A35T – FTG256C package) with FTDI's FT2232H Dual-Channel USB device. It is an Artix-7 based replacement and upgrades of Mimas Spartan 6 FPGA Board. It is specially designed for the development and integration of FPGA based accelerated features to other designs. The USB 2.0 host interface based on popular FT2232H offers high bandwidth data

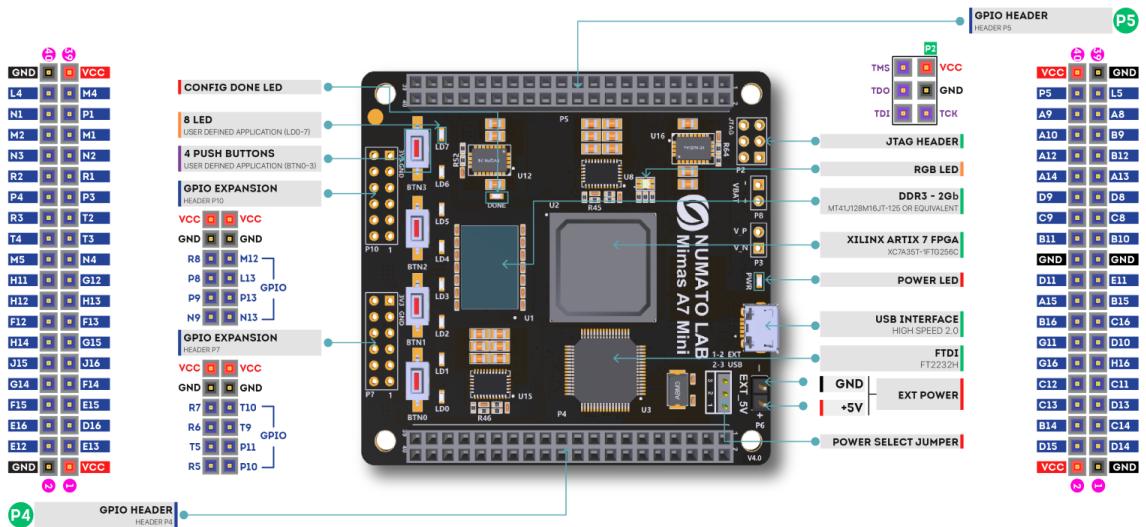
transfer and board programming without the need for any external programming adapters.

## **Features,**

- Device: Xilinx Artix 7 FPGA (XC7A35T-1FTG256C)
  - DDR3: 2Gb DDR3 (MT41J128M16JT-125 or equivalent)
  - Built-in programming interface. No expensive JTAG adapters needed for programming the board
  - Onboard 128Mb flash memory for FPGA configuration storage and custom user data storage
  - High-Speed USB 2.0 interface for On-board flash programming. FT2232H Channel B is dedicated to JTAG Programming. Channel A can be used for custom applications
  - 100MHz CMOS oscillator
  - 8 LEDs, 1 RGB LED and 4 Push Buttons for user-defined purposes
  - FPGA configuration via JTAG and USB
  - Maximum IOs for user-defined purposes
- FPGA – 70 IOs (35 professionally length matched Differential Pairs) and two 2×6 Expansion Headers

## **Applications,**

- Product Prototype Development
- Accelerated computing integration
- Development and testing of custom embedded processors
- Communication devices development
- Educational tool for Schools and Universities



## Specifications,

Attribute	Value
<b>Dimensions</b>	6 × 4 × 1 in
<b>FPGA</b>	<a href="#">XC7A35T – 1FTG256C</a>
<b>Memory</b>	<a href="#">DDR3 – 2Gb</a>
<b>Configuration Options</b>	<a href="#">JTAG</a> , <a href="#">USB</a>
<b>Host Interface</b>	<a href="#">USB 2.0</a>
<b>Primary Clock Frequency</b>	<a href="#">100MHz</a>
<b>Number Of GPIOs (Max)</b>	<a href="#">70</a>
<b>Number Of Diff Pairs</b>	<a href="#">35</a>

## Software



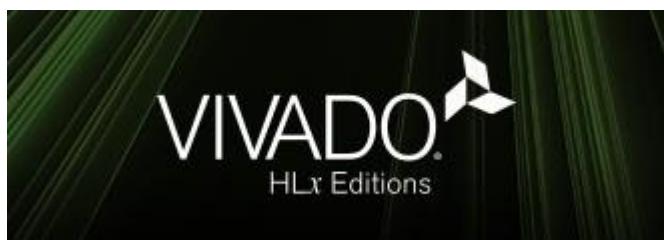
Icarus Verilog is a Verilog simulation and synthesis tool. It operates as a compiler, compiling source code written in Verilog (IEEE-1364) into some target format. For batch simulation, the compiler can

generate an intermediate form called vvp assembly. This intermediate form is executed by the ``vvp" command. For synthesis, the compiler generates netlists in the desired format.

The compiler proper is intended to parse and elaborate design descriptions written to the IEEE standard *IEEE Std 1364-2005*. This is a fairly large and complex standard, so it will take some time to fill all the dark alleys of the standard, but that's the goal.

Icarus Verilog is a work in progress, and since the language standard is not standing still either, it probably always will be. That is as it should be. However, I will make stable releases from time to time, and will endeavour to not retract any features that appear in these stable releases. The quick links above will show the current stable release.

The main porting target is Linux, although it works well on many similar operating systems. Various people have contributed precompiled binaries of stable releases for a variety of targets. These releases are ported by volunteers, so what binaries are available depends on who takes the time to do the packaging. Icarus Verilog has been ported to That Other Operating System, as a command line tool, and there are installers for users without compilers. You can compile it entirely with free tools, too, although there are precompiled binaries of stable releases.



**Vivado Design Suite** is a software suite produced by [Xilinx](#) for synthesis and analysis of [HDL](#) designs, superseding [Xilinx ISE](#) with additional features for [system on a chip](#) development and [high-level synthesis](#).<sup>[1][2][3][4]</sup> Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE).<sup>[5][6][7]</sup>

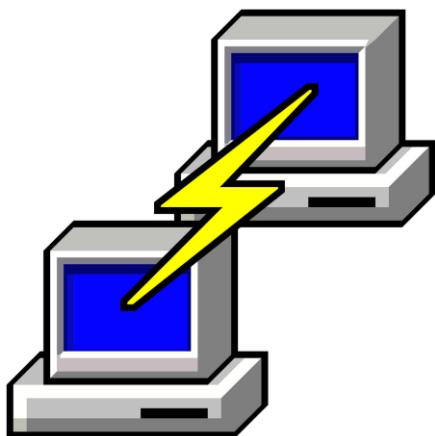
Like the later versions of [ISE](#), Vivado includes the in-built logic simulator [ISIM](#).<sup>[8]</sup> Vivado also introduces high-level synthesis, with a toolchain that converts C code into programmable logic.<sup>[9]</sup>

Replacing the 15-year-old ISE with Vivado Design Suite took 1000 [person-years](#) and cost US\$200 million.



Tenagra is an FPGA System management tool for configuring and communicating with Numato Lab's supported FPGA modules and development platforms. This software is designed to be a single interface for managing the devices and exercising some of the available features. Currently, Tenagra supports configuring the FPGA module/board (programming) and Memory Exerciser that can transfer data between various memories on the device. This includes both external DDR Memory and Block RAM available within the FPGA device. With Tenagra, you can create multiple configuration setups with different bitstreams and settings for each device model so that switching between multiple bitstreams is a breeze. This is especially helpful during development where the device may need to be reprogrammed with various bitstreams repeatedly.

#### **Driver for Mimas A7 Mini Module**



PuTTY is a free implementation of SSH (**and telnet**) for PCs running **Microsoft Windows** (it also includes an xterm terminal emulator). You will find PuTTY useful if you want to access an account on a Unix or other multi-user system from a PC (for example your own or one in an internet cafe).

### 3. FSM Designs: Mealy & Moore Machines and Up Down Counter

Step 1- Connect the real time FPGA Mimas A7 mini board, open the device manager to set its configuration. Now we have to get

tenegra a software which acts like a bridge between the FPGA board and the pc, where the program will be uploaded. You can get tenegra under Numato Lab

Due to annual holidays orders will be shipping from first week of January 2022. [Business](#)

[sales@numato.com](mailto:sales@numato.com)

Type a keyword or SKU 

**NUMATO LAB®**

PRODUCTS DEVELOPER'S CORNER HELP WHERE TO BUY COMPANY  

# Easy to use products helpful tools

Transform your ideas in to working products with minimum lead time and competitive cost.



**AUTOMATION & DATA ACQUISITION**

<b>GPIO Modules</b>	<b>Relay Modules</b>
Bluetooth GPIO	Bluetooth Relay
Ethernet GPIO	Ethernet Relay
Modbus GPIO	Modbus Relay
USB GPIO	Relay Controller
WiFi GPIO	Solid State Relay

**OTHER PRODUCTS**

Power Over Ethernet	USB Relay
Adapters & Connectors	WiFi Relay
Power Supply	
USB To Serial	

**FPGA & ACCELERATED COMPUTING**

Xilinx Spartan 3	
Xilinx Spartan 6	
Xilinx Spartan 7	
Xilinx Artix7	
Xilinx Kintex 7	
Xilinx Zynq	
Intel MAX 10	
FMC Modules	
Expansion Modules	

**SOFTWARE & IP**

XO-Bus Lite Host Interface IP	
Tenagra-FPGA System Mgmt Software	
Rhea Device Management Software	
Theia Android Application	

**NUMATO LAB®**

PRODUCTS DEVELOPER'S CORNER HELP WHERE TO BUY COMPANY  



 Downloads

 Knowledge Base

User Manual

Tenagra-01.03-Beta Release

Tenagra-01.02-Beta Release

Tenagra (Older release)

Getting started with Tenagra FPGA System Management Software

Specifications

Attribute	Value

After this we have to download few drivers' fir Mimas A7 Mini Module.

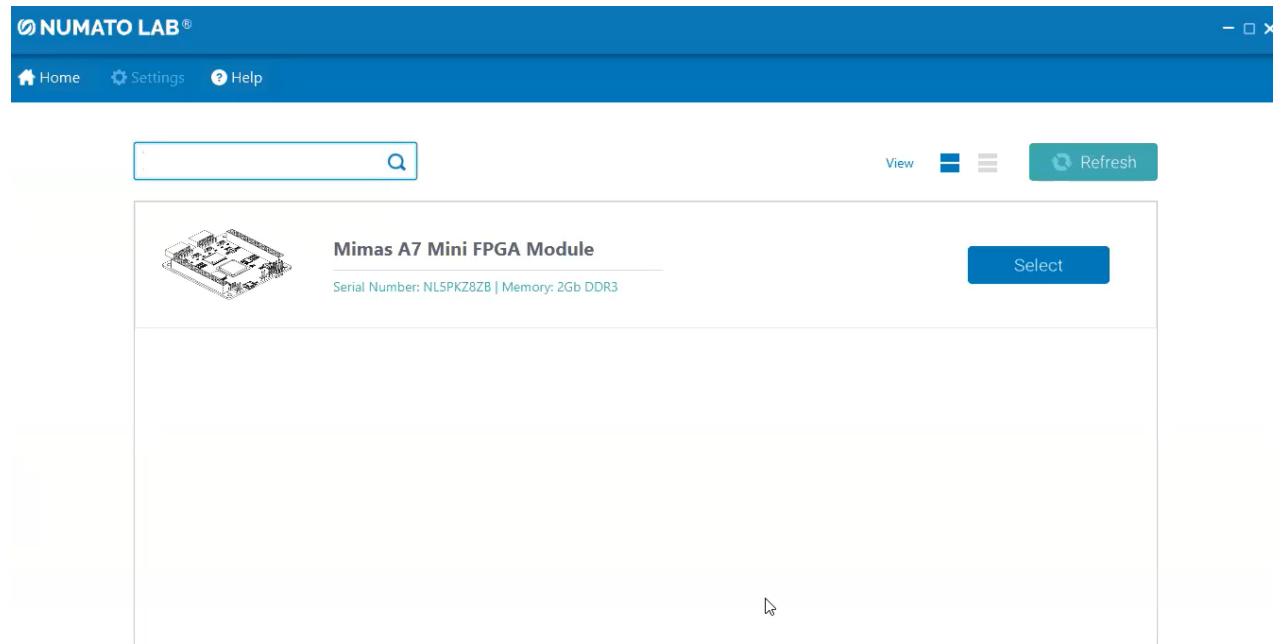
 Downloads

User Manual
<a href="#">Driver for Mimas A7 Mini Module</a>
Tenagra FPGA Management Suite
Schematics
XDC File For Vivado Designs
<a href="#">View/Download 3D Model (STEP)</a>

## Specifications

Attribute	Value
Dimensions	6 x 4 x 1 in

Launch tenagra where we will be viewing two ways or methods to put the program in our FPGA .



The screenshot shows the Tenagra software interface. At the top, there is a blue header bar with the NUMATO LAB® logo, a search bar, and various menu options like View, Refresh, and a settings icon. Below the header, a main panel displays a product card for the "Mimas A7 Mini FPGA Module". The card features an image of the module, its name, and a serial number. A "Select" button is located on the right side of the card. The background of the software has a light gray grid pattern.

We could have used Xilinx jtag , a Xilinx platform cable, as it is quite expensive, we will not be using it rather we will using either of the two ways, XVC Server or Program device.

The screenshot shows the NUMATO LAB® software interface. At the top, there's a navigation bar with 'Home', 'Settings', and 'Help' buttons. Below the navigation bar, there's a large image of the Mimas A7 Mini FPGA module. To the right of the image, there's a section titled 'Information' with details about the device: 'Device: Xilinx Artix 7 FPGA (XC7A35T-1FTG256C)', 'DDR3: 2Gb DDR3 (MT41J128M16JT-125 or equivalent)', and a 'Schematics' link. Below this, there are four colored boxes: a blue box labeled 'Program device' with a small laptop icon, a green box labeled 'Memory exerciser' with a small memory chip icon, a blue box labeled 'XVC Server' with a small server icon, and a dark grey box labeled 'Product documentation' with a small document icon. On the left side of the main content area, there's a table with the following data:

Device Name	Mimas A7 Mini FPGA Module
Manufacturer	Numato Systems Pvt. Ltd.
Host Interface	USB
VID/PID	0x2A19 / 0x100E
FPGA	Xilinx Artix 7
Configuration Memory	SPI Flash
Serial Number	NL5PKZ8ZB

0.0.0.01.03 Beta

© 2019 Numato Systems Pvt. All Rights Reserved. [Privacy Policy](#) [Terms of Use](#)

XVC server is nothing but Xilinx virtual cable, you can research and find out what it is. So, this is one way another way which we will be using to program is by an integrated circuit, FT2232HQ an onboard chip already available on Mimas A7 Mini, used for usb communication usb 2.0 communication not usb 3.0 with a speed of 480Mb/s. So, if we conclude we are using normal option that is program device to install the program on the FPGA.

< Back to device information

## Mimas A7 Mini FPGA M...

FPGA Module

**Program device**

[View demo](#)

Instant actions

Use these actions without running any configuration

[Erase now](#) [Verify now](#)

Drag and drop a configuration file here

Supported files: Bitstream (.bit), Raw Binary Bitstream (.bin)

or

[Add a configuration from your computer](#)

[Product documentation](#) [Firmware Update](#)

0.0.0.01.03 Beta

© 2019 Numato Systems Pvt. All Rights Reserved. [Privacy Policy](#) [Terms of](#)

Add the bit file saved from Xilinx vivado integrated with this FPGA.

< Back to device information

## Mimas A7 Mini FPGA ...

FPGA Module

**Program device**

[View demo](#)

Instant actions

Use these actions without running any configuration

[Erase now](#) [Verify now](#)

Current configuration

**MimasA7Mini\_...** [Edit](#)

File name: MimasA7Mini\_...

Select memory:

SRAM  Flash memory

[Run](#)

**Available configurations**

	Refresh
<input checked="" type="radio"/> MimasA7Mini_Microblaze_Helloworld.bit	<a href="#">See comments</a>
<a href="#">Ready!</a>	

[Add more configurations](#)

[Product documentation](#) [Firmware Update](#)

0.0.0.01.03 Beta

© 2019 Numato Systems Pvt. All Rights Reserved. [Privacy Policy](#) [Terms of](#)

Run and configure the board with the program and after that if we have to launch putty software.

**NUMATO LAB®**

Home Settings Help

< Back to device information

## Mimas A7 Mini FPGA ...

FPGA Module

Program device

View demo

Instant actions

Use these actions without running any configuration

Erase now Verify now

Current configuration

MimasA7Mini\_Microblaze\_Helloworld.bit

File name: MimasA7Mini\_Microblaze\_Helloworld.bit

Select memory:

SRAM Flash memory

Run Configuration completed

Available configurations

MimasA7Mini\_Microblaze\_Helloworld.bit See comments Configuration completed

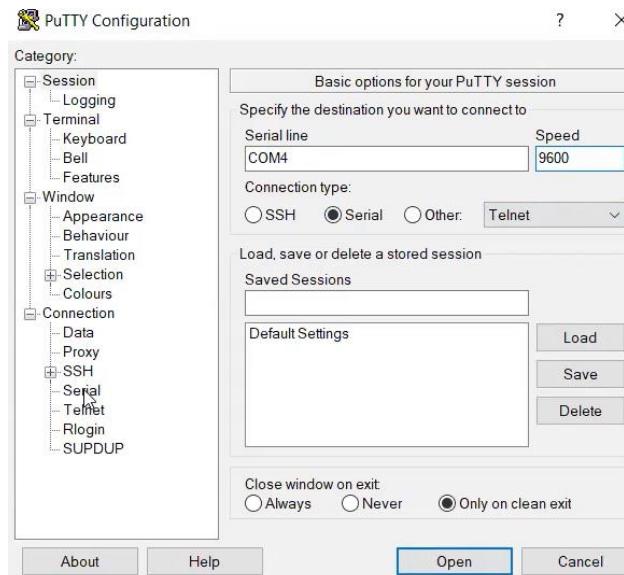
Add more configurations

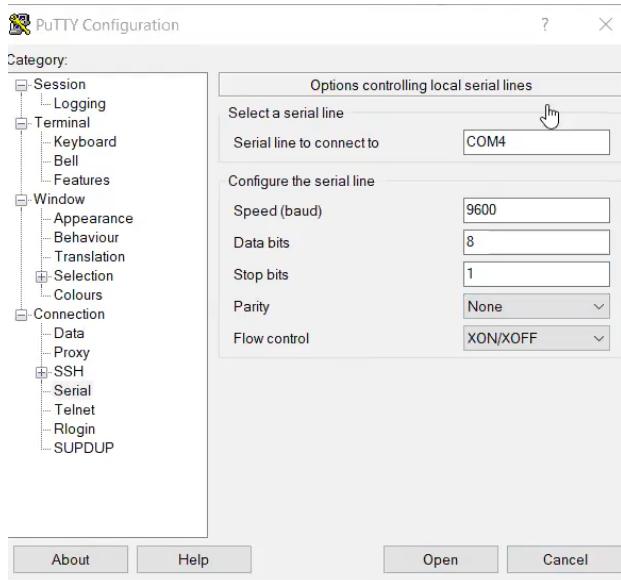
Product documentation Firmware Update

0.00.01.03 Beta

© 2019 Numato Systems Pvt. All Rights Reserved. Privacy Policy Terms of

**Step 2- Launch putty our serial monitors where it will receive data and make necessary changes with respect to port and connection type and so on.**





Putty session will continuously print Hello world after we upload the program to our FPGA and even, we can stop it from happening with a reset button present on the board.

A screenshot of a terminal window titled 'COM4 - PuTTY'. The window displays the text 'Hello World' repeated multiple times, indicating a continuous loop of output. The terminal has standard window controls at the top right.

Open the C file where the program for hello world is written.

```
helloworld.c - Notepad
File Edit Format View Help
*****
*
* Copyright (C) 2009 - 2014 Xilinx, Inc. All rights reserved.
*
* Permission is hereby granted, free of charge, to any person obtaining a copy
* of this software and associated documentation files (the "Software"), to deal
* in the Software without restriction, including without limitation the rights
* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
* copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
*
* Use of the Software is limited solely to applications:
* (a) running on a Xilinx device, or
* (b) that interact with a Xilinx device through a bus or interconnect.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
```

```
helloworld.c - Notepad
File Edit Format View Help
*
* (a) running on a Xilinx device, or
* (b) that interact with a Xilinx device through a bus or interconnect.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
* XILINX BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
* WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF
* OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*
* Except as contained in this notice, the name of the Xilinx shall not be used
* in advertising or otherwise to promote the sale, use or other dealings in
* this Software without prior written authorization from Xilinx.
*
*****/
```

---

```
/*
* helloworld.c: simple test application
*
* This application configures UART 16550 to baud rate 9600.
* PS7 UART (Zynq) is not initialized by this application, since
* bootrom/bsp configures it to baud rate 115200
```

```

helloworld.c - Notepad
File Edit Format View Help
* helloworld.c: simple test application
*
* This application configures UART 16550 to baud rate 9600.
* PS7 UART (Zynq) is not initialized by this application, since
* bootrom/bsp configures it to baud rate 115200
*
* -----
* | UART TYPE BAUD RATE |
* -----
* uartns550 9600
* uartlite Configurable only in HW design
* ps7_uart 115200 (configured by bootrom/bsp)
*/



#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"

int main()
{
    while(1)
    {
* -----
* | UART TYPE BAUD RATE |
* -----
* uartns550 9600
* uartlite Configurable only in HW design
* ps7_uart 115200 (configured by bootrom/bsp)
*/



#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"

|
int main()
{
    while(1)
    {
        print("Hello World\n\r");
    }
}

```

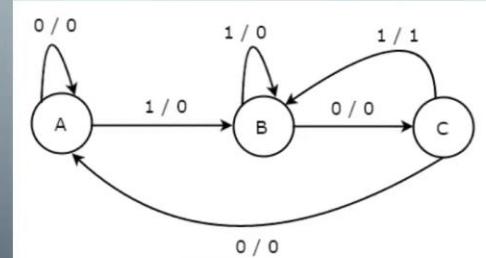
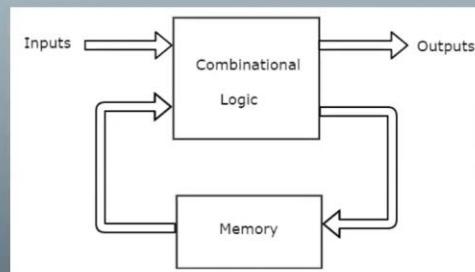
Things to know before proceeding,

# Sequential circuits

- Finite State Machines
  - The behavior of synchronous sequential circuits can be represented in the graphical form and it is known as **state diagram**.
  - Decision making logic and Control of the Digital System.
  - There are two types of FSMs.
    - Mealy State Machine
    - Moore State Machine

# Sequential circuits

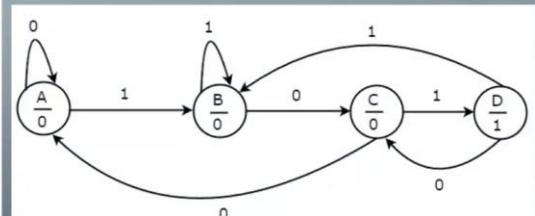
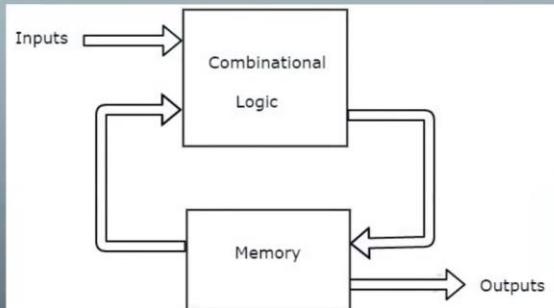
- Finite State Machines
    - Mealy State Machine
- A Finite State Machine is said to be Mealy state machine, if outputs depend on both present inputs & present states.



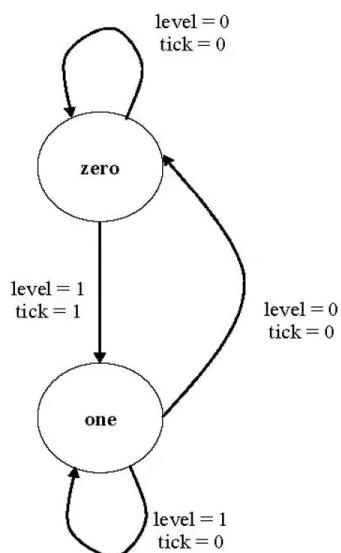
# Sequential circuits

- Finite State Machines
  - Moore State Machine

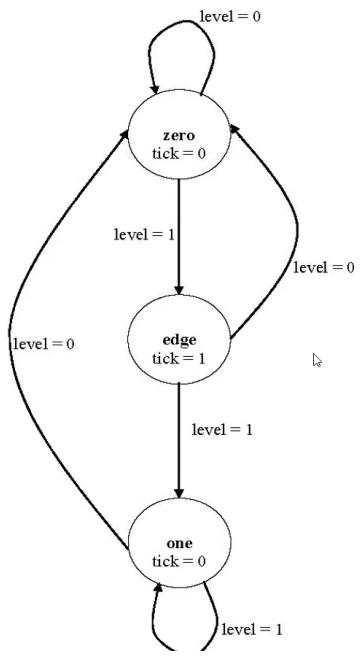
A Finite State Machine is said to be Moore state machine, if outputs depend only on present states.



Mealy state machines



Moore state machines



Step 3- Open notepad to write the Verilog code, a code to explain the basics of mealy and Moore state machine

```

C:\Users\Kishore Damam\Desktop\fsms.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
File Edit View Insert Project New Open Save Save As Find Replace Properties Help
fsms.v X
1 // edgeDetector.v
2 // Moore and Mealy Implementation
3
4 module edgeDetector
5 (
6   input wire clk, reset,
7   input wire level,
8   output reg Mealy_tick, Moore_tick
9 );
10
11 localparam // 2 states are required for Mealy
12   zeroMealy = 1'b0,
13   oneMealy = 1'b1;
14
15 localparam [1:0] // 3 states are required for Moore
16   zeroMoore = 2'b00,
17   edgeMoore = 2'b01,
18   oneMoore = 2'b10;
19
20 reg stateMealy_reg, stateMealy_next;
21 reg[1:0] stateMoore_reg, stateMoore_next;
22
23 always @(posedge clk, posedge reset)
24 begin
  length: 2,603 lines: 81 Ln: 7 Col: 23 Pos: 132 Windows (CR LF) UTF-8 INS
Verilog file
25
26   zeroMoore = 2'b00,
27   edgeMoore = 2'b01,
28   oneMoore = 2'b10;
29
30 reg stateMealy_reg, stateMealy_next;
31 reg[1:0] stateMoore_reg, stateMoore_next;
32
33 always @(posedge clk, posedge reset)
34 begin
  if(reset) // go to state zero if rese
    begin
      stateMealy_reg <= zeroMealy;
      stateMoore_reg <= zeroMoore;
    end
  else // otherwise update the states
    begin
      stateMealy_reg <= stateMealy_next;
      stateMoore_reg <= stateMoore_next;
    end
  end
35
36
37 // Mealy Design
38 always @(stateMealy_reg, level)
39 begin
  length: 2,603 lines: 81 Ln: 16 Col: 14 Sel: 9 | 1 Windows (CR LF) UTF-8 INS
Verilog file

```

```

C:\Users\Kishore Damam\Desktop\fsms.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fsms.v
31 begin
32     stateMealy_reg <= stateMealy_next;
33     stateMoore_reg <= stateMoore_next;
34 end
35
36
37 // Mealy Design
38 always @(stateMealy_reg, level)
39 begin
40     // store current state as next
41     stateMealy_next = stateMealy_reg; // required: when no case statement is satisfied
42
43     Mealy_tick = 1'b0; // set tick to zero (so that 'tick = 1' is available for 1 cycle of
44     case(stateMealy_reg)
45         zeroMealy: // set 'tick = 1' if state = zero and level = '1'
46             if(level)
47                 begin // if level is 1, then go to state one,
48                     stateMealy_next = oneMealy; // otherwise remain in same state.
49                     Mealy_tick = 1'b1;
50                 end
51         oneMealy:
52             if(~level) // if level is 0, then go to zero state,
53                 stateMealy_next = zeroMealy; // otherwise remain in one state.
54     endcase

```

Verilog file length: 2,603 lines: 81 Ln: 16 Col: 14 Sel: 9|1 Windows (CR LF) UTF-8 INS

```

C:\Users\Kishore Damam\Desktop\fsms.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fsms.v
43 Mealy_tick = 1'b0; // set tick to zero (so that 'tick = 1' is available for 1 cycle of
44 case(stateMealy_reg)
45     zeroMealy: // set 'tick = 1' if state = zero and level = '1'
46         if(level)
47             begin // if level is 1, then go to state one,
48                 stateMealy_next = oneMealy; // otherwise remain in same state.
49                 Mealy_tick = 1'b1;
50             end
51     oneMealy:
52         if(~level) // if level is 0, then go to zero state,
53             stateMealy_next = zeroMealy; // otherwise remain in one state.
54     endcase
55
56
57 // Moore Design
58 always @(stateMoore_reg, level)
59 begin
60     // store current state as next
61     stateMoore_next = stateMoore_reg; // required: when no case statement is satisfied
62
63     Moore_tick = 1'b0; // set tick to zero (so that 'tick = 1' is available for 1 cycle of
64     case(stateMoore_reg)
65         zeroMoore: // if state is zero,
66             if(level) // and level is 1

```

Verilog file length: 2,603 lines: 81 Ln: 16 Col: 14 Sel: 9|1 Windows (CR LF) UTF-8 INS

A screenshot of the Notepad++ application window. The title bar reads "C:\Users\Kishore Damam\Desktop\fsms.v - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Find, and Copy. The status bar at the bottom shows "Verilog file", "length : 2,603 lines : 81", "Ln : 16 Col : 14 Sel : 9 | 1", "Windows (CR LF)", "UTF-8", and "INS". The main code editor window displays the following Verilog code:

```
64 case(stateMoore reg)
65     zeroMoore: // if state is zero,
66         if(level) // and level is 1
67             stateMoore_next = edgeMoore; // then go to state edge.
68     edgeMoore:
69         begin
70             Moore_tick = 1'bl; // set the tick to 1.
71             if(level) // if level is 1,
72                 stateMoore_next = oneMoore; // go to state one,
73             else
74                 stateMoore_next = zeroMoore; // else go to state zero.
75         end
76     oneMoore:
77         if(~level) // if level is 0,
78             stateMoore_next = zeroMoore; // then go to state zero.
79     endcase
80 end
81 endmodule
```

Open notepad to write the verilog code, a code to explain the UART module

A screenshot of the Notepad++ application window. The title bar reads "C:\Users\Kishore Damam\Desktop\uart\_tx.v - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Find, and Copy. The status bar at the bottom shows "Verilog file", "length : 2,603 lines : 81", "Ln : 16 Col : 14 Sel : 9 | 1", "Windows (CR LF)", "UTF-8", and "INS". The main code editor window displays the following Verilog code:

```
1 //////////////////////////////////////////////////////////////////
2 // File Downloaded from http://www.nandland.com
3 //////////////////////////////////////////////////////////////////
4 // This file contains the UART Transmitter. This transmitter is able
5 // to transmit 8 bits of serial data, one start bit, one stop bit,
6 // and no parity bit. When transmit is complete o_Tx_done will be
7 // driven high for one clock cycle.
8 //
9 // Set Parameter CLKS_PER_BIT as follows:
10 // CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
11 // Example: 25 MHz Clock, 115200 baud UART i
12 // (25000000)/(115200) = 217
13
```

```
C:\Users\Kishore Damam\Desktop\uart_txv - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fimx.v uart_txv
7 // driven high for one clock cycle.
8 //
9 // Set Parameter CLKS_PER_BIT as follows:
10 // CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
11 // Example: 25 MHz Clock, 115200 baud UART
12 // (25000000)/(115200) = 217
13
14 module UART_TX
15 #(parameter CLKS_PER_BIT = 217)
16 (
17     input      i_Clock,
18     input      i_TX_DV,
19     input [7:0] i_TX_Byte,
20     output     o_TX_Active,
21     output reg o_TX_Serial,
22     output     o_TX_Done
23 );
24
25 parameter IDLE      = 3'b000;
26 parameter TX_START_BIT = 3'b001;
27 parameter TX_DATA_BITS = 3'b010;
28 parameter TX_STOP_BIT = 3'b011;
29 parameter CLEANUP   = 3'b100;
30

Verilog file length: 4,079 lines: 147 Ln: 135 Col: 7 Pos: 3,906 Windows (CR LF) UTF-8 INS
```

```
C:\Users\Kishore Damam\Desktop\uart_txv - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fimx.v uart_txv
25 parameter IDLE      = 3'b000;
26 parameter TX_START_BIT = 3'b001;
27 parameter TX_DATA_BITS = 3'b010;
28 parameter TX_STOP_BIT = 3'b011;
29 parameter CLEANUP   = 3'b100;
30
31 reg [2:0] r_SM_Main      = 0;
32 reg [7:0] r_Clock_Count = 0;
33 reg [2:0] r_Bit_Index    = 0;
34 reg [7:0] r_TX_Data      = 0;
35 reg      r_TX_Done       = 0;
36 reg      r_TX_Active     = 0;
37
38 always @ (posedge i_Clock)
39 begin
40
41     case (r_SM_Main)
42         IDLE :
43             begin
44                 o_TX_Serial  <= 1'b1;           // Drive Line High for Idle
45                 r_TX_Done    <= 1'b0;
46                 r_Clock_Count <= 0;
47                 r_Bit_Index  <= 0;
```

The screenshot shows a Notepad++ window with the file name "uart\_tx.v". The code is a Verilog module for a UART transmitter. It includes logic to handle the start bit, serial output, and clock counting. The code uses standard Verilog syntax with comments explaining the logic.

```
43 begin
44     o_TX_Serial    <= 1'b1;           // Drive Line High for Idle
45     r_TX_Done      <= 1'b0;
46     r_Clock_Count <= 0;
47     r_Bit_Index    <= 0;
48
49     if (i_TX_DV == 1'b1)
50     begin
51         r_TX_Active <= 1'b1;
52         r_TX_Data   <= i_TX_Byte;
53         r_SM_Main   <= TX_START_BIT;
54     end
55     else
56         r_SM_Main <= IDLE;
57 end // case: IDLE
58
59
60 // Send out Start Bit. Start bit = 0
61 TX_START_BIT :
62 begin
63     o_TX_Serial <= 1'b0;
64
65     // Wait CLKS_PER_BIT-1 clock cycles for start bit to finish
66     if (r_Clock_Count < CLKS_PER_BIT-1)
```

The screenshot shows a Notepad++ window with the file "uart\_tx.v" open. The code is written in Verilog and defines a module for a UART transmitter. It includes logic to send start bits, data bits, and stop bits, along with clock management and state machines. The code uses standard Verilog syntax with comments explaining the logic.

```
58
59
60     // Send out Start Bit. Start bit = 0
61 TX_START_BIT :
62     begin
63         o_TX_Serial <= 1'b0;
64
65         // Wait CLKS_PER_BIT-1 clock cycles for start bit to finish
66         if (r_Clock_Count < CLKS_PER_BIT-1)
67             begin
68                 r_Clock_Count <= r_Clock_Count + 1;
69                 r_SM_Main      <= TX_START_BIT;
70             end
71         else
72             begin
73                 r_Clock_Count <= 0;
74                 r_SM_Main      <= TX_DATA_BITS;
75             end
76         end // case: TX_START_BIT
77
78
79     // Wait CLKS_PER_BIT-1 clock cycles for data bits to finish
80 TX_DATA_BITS :
81     begin
```

```
76
77
78
79 // Wait CLKS_PER_BIT-1 clock cycles for data bits to finish
80 TX_DATA_BITS :
81 begin
82     o_TX_Serial <= r_TX_Data[r_Bit_Index];
83
84     if (r_Clock_Count < CLKS_PER_BIT-1)
85     begin
86         r_Clock_Count <= r_Clock_Count + 1;
87         r_SM_Main      <= TX_DATA_BITS;
88     end
89     else
90     begin
91         r_Clock_Count <= 0;
92
93         // Check if we have sent out all bits
94         if (r_Bit_Index < 7)
95         begin
96             r_Bit_Index <= r_Bit_Index + 1;
97             r_SM_Main      <= TX_DATA_BITS;
98         end
99         else
100    begin
101        r_SM_Main      <= TX_DATA_BITS;
102    end
103
104    end
105 end // case: TX_DATA_BITS
```

```
108 // Send out Stop bit. Stop bit = 1
109 TX_STOP_BIT :
110 begin
111     o_TX_Serial <= 1'b1;
112
113     // Wait CLKS_PER_BIT-1 clock cycles for Stop bit to finish
114     if (r_Clock_Count < CLKS_PER_BIT-1)
115     begin
116         r_Clock_Count <= r_Clock_Count + 1;
117         r_SM_Main      <= TX_STOP_BIT;
118     end
119     else
120     begin
```

C:\Users\Kishore Damam\Desktop\uart\_tx.v - Notepad++

```

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
File Edit View Insert Project Tools Options Plugins Window Help
fsm.v uart_tx.v

124      r_TX_Active <= 1'b0;
125    end
126  end // case: TX_STOP_BIT
127
128
129  // Stay here 1 clock
130  CLEANUP :
131  begin
132    r_TX_Done <= 1'b1;
133    r_SM_Main <= IDLE;
134  end
135
136
137  default :
138    r_SM_Main <= IDLE;
139
140  endcase
141 end
142
143 assign o_TX_Active = r_TX_Active;
144 assign o_TX_Done   = r_TX_Done;
145
146 endmodule
147

```

Verilog file length:4,079 lines:147 Ln :147 Col :1 Pos :4,080 Windows (CR LF) UTF-8 INS

## Output

