Digital Watch

A Report Submitted
as Requirements for the Mini Project 2B Course of
Semester VI, AY 2023-2024

Submitted by

Mohd Raza	**02**
Aditya Jadhav	**14**
Himanshu Jogi	**15**
Mayuti Kadam	**16**

Supervised by

Dr. Sudhakar Mande, Prof. Ankur and Prof. Lakshmi Iyer
Department of Electronics and Telecommunication Engineering
Don Bosco Institute of Technology,Mumbai India

May 2024

Don Bosco Institute of Technology
(Affiliated to the University of Mumbai)
Premier Automobiles Road, Kurla, Mumbai - 400070

Certificate

This is to certify that the Mini project entitled **Digital Watch** is a work of

Mohd Raza	**02**
Aditya Jadhav	**14**
Himanshu Jogi	**15**
Mayuri Kadam	**16**

submitted as fulfilment of the requirement for the Mini Project2B of **"Semester VI"** in **"Third Year of Engineering AY 2023-2024"**.

Prof. Lakshmi Iyer (Project Guide)	**Prof. Ankur G.** (Project Guide)
Dr.Sudhakar Mande (Project Co-ordinator/Guide)	**Prof. Namita Agarwal** (HOD, EXTC)

Don Bosco Institute of Technology

(Affiliated to the University of Mumbai)
Premier Automobiles Road, Kurla, Mumbai - 400070

Mini Project 2B Report Approval

This Mini project report entitled **'Digital Watch'** by **Mohd Raza, Aditya Jadhav, Himanshu Jogi, Mayuri Kadam** is approved for the completion of Mini Project 2B course **of Sem VI of AY 2023-2024 in Dept. of Electronics & Telecommunication Engineering.**

Examiners

1. _____

2. _____

Date : 06 / 05 / 24

Place : **Kurla, Mumbai**

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or works have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

GROUP MEMBERS:

1. Mohd Raza _____
2. Aditya Jadhav _____
3. Himanshu Jogi _____
4. Mayuri Kadam _____

Date: 06 / 05 / 2024

Contents

List of Figures

Abstract

The proposed project aims to design and implement a digital watch using an FPGA (Field Programmable Gate Array) board. The digital watch will display the current time, date, and other relevant information on a display module. The project will involve the design and development of a custom digital logic circuit that can accurately track time, update the display, and handle user input for setting the time and date.

Chapter 1

Introduction

1.1 Introduction:

The project "Digital Watch" using an FPGA board focuses on designing and implementing a digital clock that displays time digitally on a 7-segment display. The digital watch project utilizes Verilog Hardware Description Language to create functional blocks for time tracking, display, and user input. The clock operates in a 12 / 24-hour format and allows users to set the time and alarm using switches on the FPGA board. The design involves generating timing signals, incrementing hours, minutes, and seconds, and managing alarm settings. The project emphasizes the use of FPGA for hardware implementation, showcasing the flexibility and scalability of FPGA-based digital logic design. Additionally, the project explores the significance of FPGA in integrating complex logic func-

tions efficiently and cost-effectively. The implementation of the digital clock project demonstrates the practical application of FPGA technology in creating intelligent and multifunctional digital devices.

The key features of the digital watch project include:

Time Tracking

- Implement a real-time clock module using the FPGA's internal clock and counters
- Accurately track hours, minutes, and seconds, accounting for leap years and daylight saving time.

Display

- Design a user-friendly interface to display the current time, date, and other relevant information.

User Input

- Implement a user input system, such as buttons or switches, to allow the user to set the time and date.
- Provide a intuitive and easy-to-use interface for setting the time and date.

Power Management

- Ensure the digital watch can operate efficiently and with low power consumption.

1.1.1 Applications

The applications of a project based on a "Digital Watch" using an FPGA board are diverse and can be seen in various fields. Some key applications include:

- Displaying Time: The primary function of a digital watch is to display time, which can be achieved using a 7-segment display or an LCD screen. The FPGA board can be programmed to generate accurate timing signals and display the time on the display.
- Stopwatch Feature: The FPGA board can be programmed to implement a stopwatch feature, which can be useful in various applications.
- Portability: The digital clock implementation with FPGA board can be applied directly in various designs based on FPGA, making it a portable solution for different applications.
- Timekeeping: The digital watch project can be used as a precise timekeeping device in everyday life, ensuring accurate time display and management.
- Education: Such projects are valuable for educational purposes, allowing students to learn about digital logic design, FPGA programming, and real-time clock systems.

- Sports and Fitness: Digital watches with stopwatch functionality are commonly used in sports and fitness activities to time workouts, races, or intervals accurately.
- Industrial Automation: In industrial settings, digital watches can be integrated into automation systems for time-sensitive processes and scheduling.
- Research and Development: Digital watches with advanced features like alarms and timers can be utilized in research labs for experiments that require precise timing.
- Accessibility: Digital watches can be designed with features to assist individuals with visual impairments, such as audible time announcements or tactile feedback.
- Embedded Systems: Projects involving digital watches on FPGA boards showcase the capabilities of FPGA technology in creating efficient and customizable embedded systems.

Chapter 2

Basic Idea and Working

FPGA programming involves configuring the FPGA to display time on a 7-segment display. The FPGA board can provide functionality ranging from basic transistor-like operations to complex combinational and sequential logic functions. Time and alarm settings are adjusted using dip switches on the board. Incrementing and decrementing minutes and hours is achieved through push buttons on the board. A global clock generates a one-second signal, which is utilized to create timing signals for the clock. The clock time is managed by a counter module that includes second, minute, and hour counters updated by an internal clock signal. The display module is responsible for showcasing the time on the 7-segment display, with digital operations like counting, comparing, incrementing, and decrementing enhancing the digital clock design.

2.1 Project Objective and Outcomes:

The main objective of this project is to design a user friendly Digital Watch.

Customizability: - Ability to customize the functionality and appearance of digital watch.

High performance :- FPGAs offer high performance computing capabilities, enabling the digital watch to execute timekeeping function accurate and efficiently.

Low power Consumption :- FPGA -based digital watches can achieve low power consumption.

Display the time digitally using 7-segment displays on the FPGA board. Implement the clock functionality by generating accurate timing signals using the FPGA's on-board clock and counters.

Allow the user to set the current time and alarm time using buttons, switches, or a keypad on the FPGA board.

Implement alarm functionality where the clock triggers an audible alarm when the current time matches the set alarm time.

Enhance the design with additional features like a stopwatch, countdown timer, or date display.

Utilize the FPGA's programmable logic to handle tasks

like button debouncing, time keeping, and display control.

Integrate the FPGA with other components like microcontrollers or LCD displays to provide a more user-friendly interface.

Demonstrate the flexibility and capabilities of FPGAs in implementing real-time digital logic designs.

Outcomes for a project based on a "Digital Watch" made using an FPGA Board could include:

High Precision Timing: FPGA-based digital watches can achieve incredibly accurate timekeeping due to the precise clocking capabilities of FPGAs, ensuring minimal time drift over extended periods.

Customizable Algorithms: FPGAs allow for easy algorithm changes in the future, making them suitable for applications where algorithm flexibility is required.

Incremental Design: Projects involving digital watches on FPGA boards can benefit from incremental design approaches, where small portions of the circuit are created and tested before building larger circuits, illustrating key engineering concepts.

Complex Functionality: FPGA-based digital watches can offer more than just timekeeping, potentially incorporating additional features like alarms, timers, and even fitness tracking functionalities due to the flexibility of FPGA programming.

Educational Value: Projects involving FPGA-based digital watches can serve as valuable learning experiences for beginners in FPGA programming, providing a hands-on opportunity to understand digital logic design and implementation.

These outcomes highlight the versatility, precision, and educational value of projects centered around creating digital watches using FPGA boards.

2.2 Working of circuit

The digital watch implemented on an FPGA in a 12-hour format with A, B, and C representing 10, 11, and 12 respectively.

Clock and Counter Mechanism : The core of the digital watch is a counter that increments at every rising edge of the clock signal. The clock signal is generated by dividing down the main FPGA clock, typically 100 MHz, to create a 1 Hz clock signal using a parameter called "onesecond".

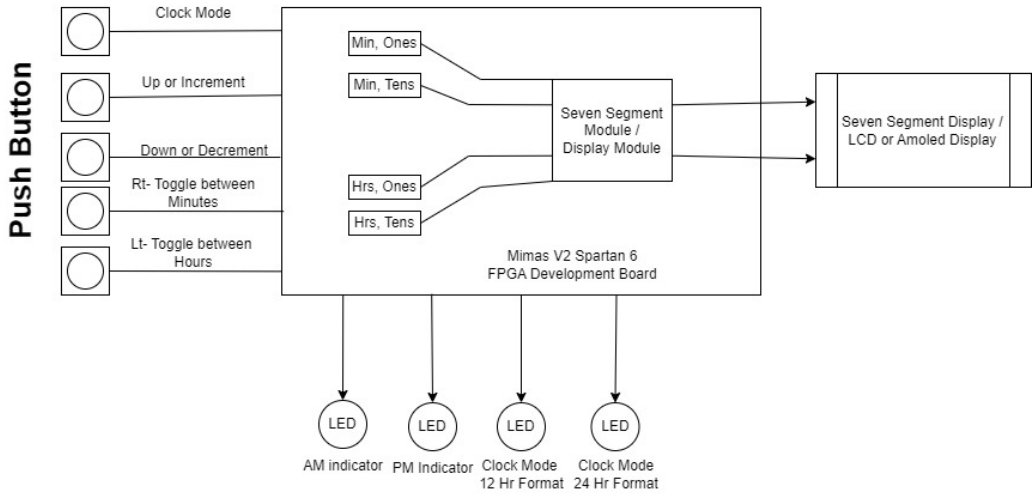


Figure 2.1: Block Diagram of Digital Watch - FPGA

Time Keeping Logic : On every rising edge of the 1 Hz clock, the module first checks if the reset signal

(rst) is high. If so, all output values are reset to their initial state. If the reset signal is low, the module checks the increment hour (hrup) and increment minute (minup) signals. If either of these is high, the hour or minute digits are incremented by one to adjust the time.

If the reset, hour increment, and minute increment signals are all low, and the enable signal (en) is high, the clock starts operating. It waits for the counter to reach the "onesecond" value, then increments the second digits. After incrementing the seconds, the module checks if the seconds have reached 60. If so, it increments the minute digits. Similarly, when the minutes reach 60, it increments the hour digits.

12-Hour Format Conversion : The hour digits are displayed in a 12-hour format, where A, B, and C represent 10, 11, and 12 respectively. This is achieved by using a conversion logic that maps the hour values from 0-23 to the appropriate 12-hour format display. Output Displays The digital watch module has six outputs, each with four bits, to display the time in a 12-hour format: s1 and s2 for the second digits m1 and m2 for the minute digits h1 and h2 for the hour digits These outputs are connected to the FPGA's seven-segment displays and LEDs to show the time.

In summary, the key aspects of this 12-hour digital

watch implementation on an FPGA are the use of counters, state machine logic, and conversion of the hour digits to the 12-hour format. The watch accurately keeps time by incrementing the seconds, minutes, and hours based on the 1 Hz clock signal, and displays the time in a 12-hour format using the FPGA's output peripherals.

Required Components for this Project:

1) FPGA Board

Chapter 3

Project Implementation

3.1 Circuit Implementation

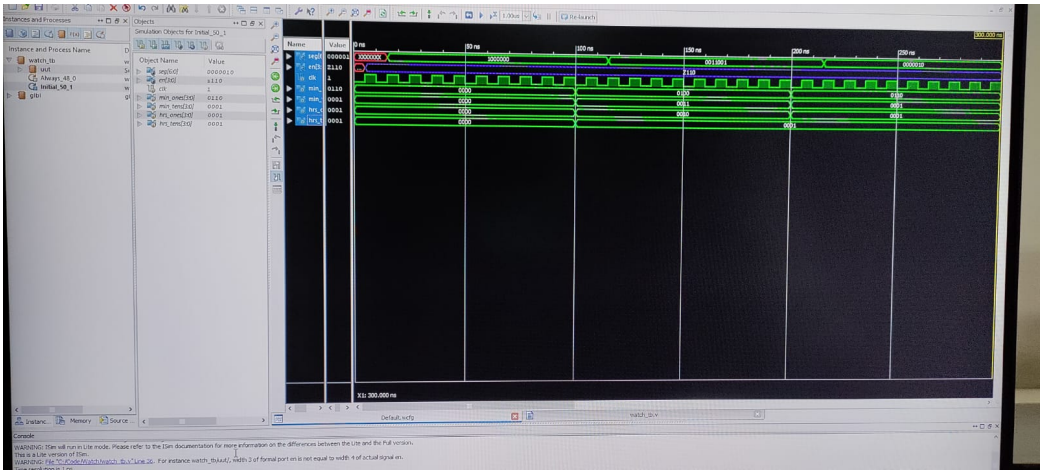


Figure 3.1: Software Output

3.2 Implementation

The digital watch in a 12-hour format using an FPGA can be implemented as follows: The digital clock module has six outputs, each with four bits, to display the time in a 12-hour format. These outputs are:

- s1 and s2 for the second digits
- m1 and m2 for the minute digits
- h1 and h2 for the hour digits

The working principle is as follows:

- There is an integer counter in the module that increments at every rising edge of the clock signal.
- A parameter called "onesecond" represents one second when using a 100 MHz clock.
- On every rising edge of the clock, the module first checks if the reset signal (rst) is high. If so, all output values are reset.
- If rst is low, the module checks the increment hour (hrup) and increment minute (minup) signals. If either of these is high, the hour or minute digits are incremented by one to adjust the time.

- If rst, hrup, and minup are low and the enable signal (en) is high, the clock starts operating. It waits for the counter to reach the "onesecond" value, then increments the second digits.
- After incrementing the seconds, the module checks if the seconds have reached 60. If so, it increments the minute digits. Similarly, when the minutes reach 60, it increments the hour digits.
- The hour digits are displayed in a 12-hour format, where A, B, and C represent 10, 11, and 12 respectively.

The key aspects of this 12-hour digital clock implementation on an FPGA are the use of counters, state machine logic, and conversion of the hour digits to the 12-hour format.

Write the code in Verilog format and we will get required output on the board and also in the simulation done int the software.

Chapter 4

Verilog Code

4.1 12 hour format

```
‘timescale 1ns / 1ps
module Top_Module(
input clk,
input center,
input right,
input left,
input up,
input down,
output [6:0] seg,
output [3:0] en,
output AMPM_indicator_led,
output clock_mode_indicator_led);

reg [31:0] counter = 0;
```

```
parameter max_count = 100_000_000;
reg [5:0] hrs, min, sec = 0;
reg [3:0] min_ones, min_tens = 0;
reg [3:0] hrs_ones, hrs_tens = 0;
reg toggle = 0;

reg pm = 0;
assign AMPM_indicator_led = pm;
reg clock_mode = 0;
assign clock_mode_indicator_led = clock_mode;

Seg7 SSM(clk, min_ones, min_tens, hrs_ones,
         hrs_tens, seg, en);

parameter display_time = 1'b0;
parameter set_time = 1'b1;
reg current_mode = set_time;

always @(posedge clk) begin
case(current_mode)
display_time: begin
if (~center) begin
clock_mode <= 0;
current_mode <= set_time;
counter <= 0;
```

```
toggle <= 0;
sec <=0;
end

if (counter <max_count) begin
counter <= counter + 1;
end else begin
counter <= 0;
sec <= sec + 1;
end
end

set_time: begin
if (center) begin
clock_mode <= 1;
current_mode <= display_time;
end

if (counter < 24_999_999) begin
counter <= counter + 1;
end else begin
counter <= 0;
case (toggle)
1'b0: begin
if (~up) begin
```

```
min <= min + 1;
end
if (~down) begin
if (min > 0) begin
min <= min - 1;
end
else if (hrs > 1) begin
hrs <= hrs - 1;
min <= 59;
end else if (hrs == 1) begin
hrs <= 12;
min <= 59;
end
end

if (~left || ~right) begin
toggle <= 1;
end
end
1'b1: begin
if (~up) begin
hrs <= hrs + 1;
end
if (~down) begin
if (hrs > 1)begin
```

```
hrs <= hrs -1;
end else if (hrs == 1) begin
hrs <= 12;
end
end
if (~right || ~left) begin
toggle <= 0;
end
end
endcase
end
end
endcase

if (sec >= 60) begin
sec <= 0;
min <= min + 1;
end
if (min >= 60) begin
min <= 0;
hrs <= hrs + 1;
end
if (hrs >= 24) begin
hrs <= 0;
end
```

```
//AM_PM Time
else begin
min_ones <= min % 10;
min_tens <= min / 10;
if (hrs < 12) begin
if (hrs == 0) begin
hrs_ones <= 12;
end else begin
hrs_ones <= hrs ;
end
pm <= 0;
end else begin
if (hrs == 12) begin
hrs_ones <= 12;
end else begin
hrs_ones <= (hrs - 12);
end
pm <= 1;
end
end
end
```

endmodule

4.2 7 Segment

```
‘timescale 1ns / 1ps
module Seg7(
input clk,
input [3:0] min_ones,
input [3:0] min_tens,input [3:0] hrs_ones,
input [3:0] hrs_tens,
output reg [6:0] seg,
output reg [3:0] en
);

// Port identification declaring wires and registers
reg [1:0] digit_display = 0;
reg [6:0] display [3:0];

reg [18:0] counter = 0;
parameter max_count = 499_999; //(100 MHz/ 2*2)-1
                                    =499999
wire [3:0] four_bit [3:0];

// Assigning values that needs to be reflected
   on 7-Segment Display
```

```
assign four_bit[0] = min_ones;
assign four_bit[1] = min_tens;
assign four_bit[2] = hrs_ones;
assign four_bit[3] = hrs_tens;

//100 MHz clock for enabling each display for 10ms
always @(posedge clk) begin
if (counter < max_count) begin
counter <= counter + 1;
end else begin
digit_display <= digit_display + 1;
counter <= 0;
end


// BCD to 7-Segment display, check for values for
  minutes and hours

case(four_bit[digit_display])
4'b0000: display[digit_display]<= 7'b1000000; //0
4'b0001: display[digit_display]<= 7'b1111001; //1
4'b0010: display[digit_display]<= 7'b0100100; //2
4'b0011: display[digit_display]<= 7'b0110000; //3
4'b0100: display[digit_display]<= 7'b0011001; //4
4'b0101: display[digit_display]<= 7'b0010010; //5
```

```
4'b0110: display[digit_display]<= 7'b0000010; //6
4'b0111: display[digit_display]<= 7'b1111000; //7
4'b1000: display[digit_display]<= 7'b0000000; //8
4'b1001: display[digit_display]<= 7'b0010000; //9
4'b1010: display[digit_display]<= 7'b0001000; //A
4'b1011: display[digit_display]<= 7'b0000011; //B
4'b1100: display[digit_display]<= 7'b1000110; //C

endcase
// Enabling Each segment and displaying the digits
case(digit_display) // cycles through display
0: begin
en <= 4'b1110;
seg <= display[0];
end

1: begin
en <= 4'b1101;
seg <= display[1];
end

2: begin
en <= 4'b1011;
seg <= display[2];
end
```

```
endcase
end
endmodule
```

4.3 UCF

```
CONFIG VCCAUX  = 3.3;

NET "clk" PERIOD = 100 MHz;
NET "clk" IOSTANDARD = LVCMOS33;

#SW1
NET "left" LOC = M18;
NET "left" IOSTANDARD = LVCMOS33;
NET "left" DRIVE = 8;
NET "left" SLEW = FAST;
NET "left" PULLUP;
#SW2
NET "right" LOC = L18;
NET "right" IOSTANDARD = LVCMOS33;
NET "right" DRIVE = 8;
NET "right" SLEW = FAST;
NET "right" PULLUP;
#SW3
```

```
NET "down" LOC = M16;
NET "down" IOSTANDARD = LVCMOS33;
NET "down" DRIVE = 8;
NET "down" SLEW = FAST;
NET "down" PULLUP;
#SW4
NET "up" LOC = L17;
NET "up" IOSTANDARD = LVCMOS33;
NET "up" DRIVE = 8;
NET "up" SLEW = FAST;
NET "up" PULLUP;
#DP1

NET "center" LOC = F17;
NET "center" IOSTANDARD = LVCMOS33;
NET "center" DRIVE = 8;
NET "center" SLEW = FAST;
NET "center" PULLUP;
#a
NET "seg[0]" LOC = A3;
NET "seg[0]" IOSTANDARD = LVCMOS33;
NET "seg[0]" DRIVE = 8;
NET "seg[0]" SLEW = FAST;
#b
NET "seg[1]" LOC = B4;
```

```
NET "seg[1]" IOSTANDARD = LVCMOS33;
NET "seg[1]" DRIVE = 8;
NET "seg[1]" SLEW = FAST;
#c
NET "seg[2]" LOC = A4;
NET "seg[2]" IOSTANDARD = LVCMOS33;
NET "seg[2]" DRIVE = 8;
NET "seg[2]" SLEW = FAST;
#d
NET "seg[3]" LOC = C4;
NET "seg[3]" IOSTANDARD = LVCMOS33;
NET "seg[3]" DRIVE = 8;
NET "seg[3]" SLEW = FAST;
#e
NET "seg[4]" LOC = C5;
NET "seg[4]" IOSTANDARD = LVCMOS33;
NET "seg[4]" DRIVE = 8;
NET "seg[4]" SLEW = FAST;
#f
NET "seg[5]" LOC = D6;
NET "seg[5]" IOSTANDARD = LVCMOS33;
NET "seg[5]" DRIVE = 8;
NET "seg[5]" SLEW = FAST;
#g
NET "seg[6]" LOC = C6;
```

```
NET "seg[6]" IOSTANDARD = LVCMOS33;
NET "seg[6]" DRIVE = 8;
NET "seg[6]" SLEW = FAST;

#Enables for Seven Segment
NET "en[2]" LOC = B3;
NET "en[2]" IOSTANDARD = LVCMOS33;
NET "en[2]" DRIVE = 8;
NET "en[2]" SLEW = FAST;
NET "en[1]" LOC = A2;
NET "en[1]" IOSTANDARD = LVCMOS33;
NET "en[1]" DRIVE = 8;
NET "en[1]" SLEW = FAST;
NET "en[0]" LOC = B2;
NET "en[0]" IOSTANDARD = LVCMOS33;
NET "en[0]" DRIVE = 8;
NET "en[0]" SLEW = FAST;

#D1
NET "AMPM_indicator_led" LOC = P15;
NET "AMPM_indicator_led" IOSTANDARD = LVCMOS33;
NET "AMPM_indicator_led" DRIVE = 8;
NET "AMPM_indicator_led" SLEW = FAST;
```

```
#D4
NET "clock_mode_indicator_led" LOC = U17;
NET "clock_mode_indicator_led" IOSTANDARD = LVCMOS33;
NET "clock_mode_indicator_led" DRIVE = 8;
NET "clock_mode_indicator_led" SLEW = FAST;

# PlanAhead Generated physical constraints
INST "clk_BUFGP" LOC = V10;
NET "clk" LOC = V10;
```

4.4 7 Segment Test Bench

```
‘timescale 1ns / 1ps
module watch_tb;
// Inputs
reg clk;
reg [3:0] min_ones;
reg [3:0] min_tens;
reg [3:0] hrs_ones;
reg [3:0] hrs_tens;

// Outputs
wire [6:0] seg;
wire [3:0] en;
```

```
// Instantiate the Unit Under Test (UUT)
Seg7 uut (
.clk(clk),
.min_ones(min_ones),
.min_tens(min_tens),
.hrs_ones(hrs_ones),
.hrs_tens(hrs_tens),
.seg(seg),
.en(en) );
always #5 clk=~clk;

initial begin
// Initialize Inputs
clk = 0;
min_ones = 0;
min_tens = 0;
hrs_ones = 0;
hrs_tens = 0;

// Wait 100 ns for global reset to finish
#100;

// Example: Simulate 12:34 to display
min_ones = 4;
```

```
min_tens = 3;
hrs_ones = 2;
hrs_tens = 1;
#100; // Wait 100ns

// Example: Simulate 11:16 to display
min_ones = 6;
min_tens = 1;
hrs_ones = 1;
hrs_tens = 1;
#100; // Wait 100ns

finish; // End the Simulation
end

endmodule
```

4.5 Main module Test Bench

```
‘timescale 1ns / 1ps

module Top_Module_tb;

// Inputs
reg clk;
```

```
reg center;
reg right;
reg left;
reg up;
reg down;

// Outputs
wire [7:0] seg;
wire [3:0] en;
wire AMPM_indicator_led;
wire clock_mode_indicator_led;

// Instantiate the Unit Under Test (UUT)
Top_Module uut (
.clk(clk),
.center(center),
.right(right),
.left(left),
.up(up),
.down(down),
.seg(seg),
.en(en),
.AMPM_indicator_led(AMPM_indicator_led),
.clock_mode_indicator_led(clock_mode_indicator_led));
always #5 clk = ~clk;
```

```
initial begin
// Initialize Inputs
clk = 0;
center = 0;
right = 0;
left = 0;
up = 0;
down = 0;
// Wait 100 ns for global reset to finish
#100;
// Add stimulus here
end
endmodule
```

Chapter 5

Detail Description of Components:

5.1 FPGA Board

The Numato Mimas V2 is a Spartan 6 FPGA development board that provides a good platform for learning and experimenting with FPGA technology. Here are the key details about this board:

- FPGA: The Mimas V2 features a Xilinx Spartan 6 LX9 FPGA in a CSG324 package. This FPGA has two built-in memory controllers that can be used to interface with external memory.
- Memory: The board includes 512Mbit of LPDDR SDRAM memory that is connected to one of the FPGA's memory controllers. This provides a good amount of volatile storage for applications like image

processing and data logging.

- Interfaces: The board has various interfaces including HDMI, VGA, audio, SD card slot, and GPIO pins that can be used to connect external peripherals.
- Programming: The Mimas V2 can be programmed using Xilinx's ISE or EDK tools. It supports both JTAG and USB-based programming, with the USB interface acting as a JTAG programmer as well as a UART adapter.
- Expansion: The board has an expansion connector that allows adding custom daughter cards or modules to extend its functionality.
- Availability of Sample Code: Numato provides sample Verilog code and projects to help users get started, such as an LCD interfacing example.
- Community Support: There is an active community around the Numato boards, with resources and tutorials available online to aid in development.

In summary, the Numato Mimas V2 is a capable Spartan 6 FPGA development board that provides a good balance of features, performance, and ease of use for both beginners and experienced FPGA developers.

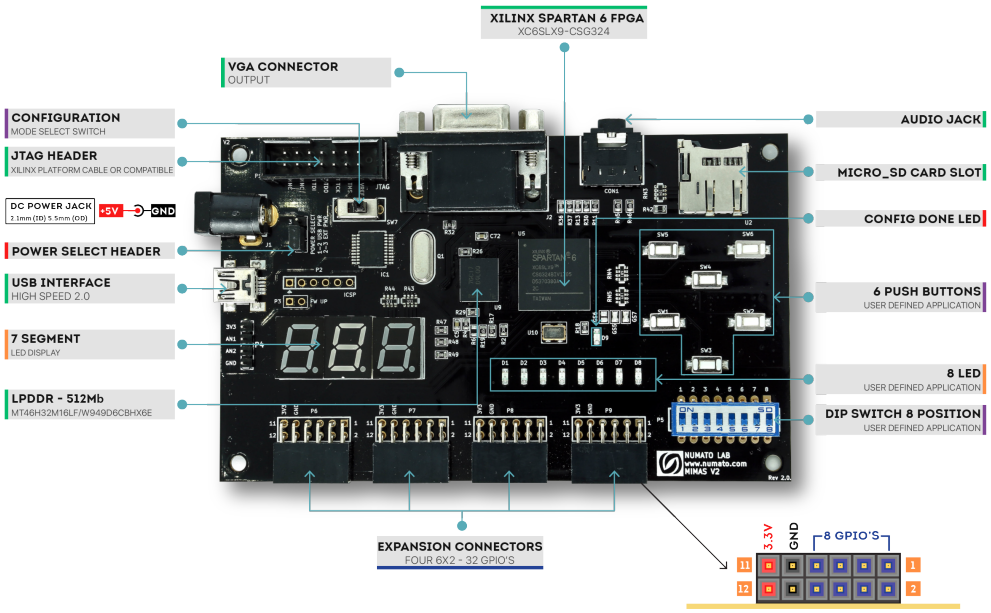


Figure 5.1: FPGA Board

5.1.1 Specifications

FPGA: AMD Spartan-6 XC6SLX9 in CSG324 package
USB 2.0 interface for onboard flash programming
FPGA configuration via JTAG and USB
8 LEDs, Six Push Buttons, and 8-way DIP switch for user-defined purposes
VGA Connector, Stereo Jack, Micro SD Card Adapter
Three-Digit Seven Segment Displays
32 IOs for user-defined purposes
Four 6×2 Expansion Connectors

Chapter 6

Advantages and Disadvantages

Advantages:
The FPGA-based digital watch project offers several advantages. Firstly, the use of an FPGA allows for a highly customizable and flexible design, enabling the implementation of advanced features like a 12-hour time format with A, B, and C representing 10, 11, and 12 respectively. The FPGA's programmable logic allows the designer to create a tailored time-keeping and display logic, optimizing the performance and functionality of the digital watch. Additionally, the FPGA platform provides the opportunity to integrate various input and output peripherals, such as switches, buttons, and seven-segment displays, to create a complete and interactive digital watch system. This integration showcases the FPGA's ability to serve as a versatile platform for building complex digital systems. Furthermore, the project demonstrates the

FPGA's capability in handling real-time constraints and time-keeping requirements, which is crucial for the accurate operation of a digital watch.

Disadvantages:

One potential disadvantage of the FPGA-based digital watch project is the relatively higher development cost compared to a traditional digital watch design. The FPGA board and the associated development tools can be more expensive than a dedicated microcontroller or application-specific integrated circuit (ASIC) solution. Additionally, the design and implementation of the digital watch logic on the FPGA may require more time and expertise in hardware description languages, such as VHDL or Verilog, as well as familiarity with FPGA development tools. This increased complexity and development effort may not be necessary for a simple digital watch application, where a more cost-effective microcontroller-based solution could be sufficient.

The FPGA-based digital watch project offers advantages like customizability, flexibility, and integration of various I/O peripherals, but may have higher development costs and complexity compared to a microcontroller-based design. The choice between FPGA and microcontroller depends on the specific project requirements and constraints.

Chapter 7

Conclusion

The "Digital Watch" project on an FPGA board involves implementing a 12-hour digital clock with time keeping logic, 12-hour format conversion, and integration with the board's input/output peripherals to create a fully functional digital watch. The key aspects of the implementation include the use of counters, state machine logic, and output display peripherals to accurately keep and display the time. The project showcases the versatility of FPGAs in building complex digital systems and the ability to implement custom time-keeping and display functionalities. The digital watch uses a counter that increments at every rising edge of a 1 Hz clock signal generated by dividing down the main FPGA clock. The watch has logic to handle time keeping, including incrementing seconds, minutes, and hours. It resets the time when the hour reaches 23. The watch displays the time in a 12-hour

format, where A, B, and C represent 10, 11, and 12 respectively. This is achieved through conversion logic that maps the hour values from 0-23 to the appropriate 12-hour format. The watch has six outputs, each with four bits, to display the time on the FPGA board's seven-segment displays and LEDs. The seconds are displayed on the LEDs, while the hours and minutes are displayed on the seven-segment displays. The watch has user input signals to enable the clock, reset the time, and increment the hour and minute values manually. These inputs are connected to the FPGA board's switches and buttons. The project demonstrates the use of counters, state machine logic, and output display peripherals to implement a functional digital watch on an FPGA platform. It showcases the capabilities of FPGAs in building complex digital systems.

In summary, the "Digital Watch" project on an FPGA board involves implementing a 12-hour digital clock with time keeping logic, 12-hour format conversion, and integration with the board's input/output peripherals to create a fully functional digital watch.

Bibliography

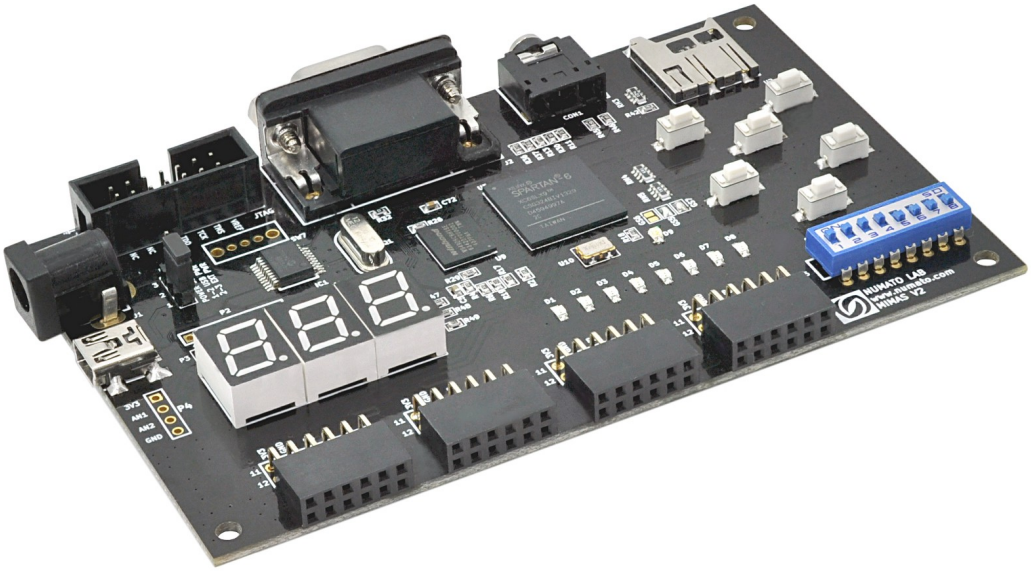
[1] Zhu, Juan Hua, Ang Wu, and Juan Fang Zhu. "Research and design of digital clock based on FPGA." Advanced Materials Research 187 (2011): 741-745.

[2] Zhu, J.H., Wu, A. and Zhu, J.F., 2011. Research and design of digital clock based on FPGA. Advanced Materials Research, 187, pp.741-745.

[3] Sivakumar, M. S., Prabu, R. T., Jayanandan, I. (2014). Design of digital clock calendar using FPGA. In IETE 45th mid term symposium on broadband-technologies and services for rural India MTS (Vol. 14).

Appendix A

Datasheets



Mimas V2 Spartan 6 FPGA Development Board

User Guide

NUMATO LAB

www.numato.com

Rev 9

Get in touch with us!

Please feel free to send a mail to one of the mail IDs below or use the Contact Us page at **http://www.numato.com** to drop us a quick message.

Technical Help
Got technical questions? Please write to **help@numato.com**

Sales Team
Questions about making payments, volume discounts, academic/open source discounts, purchase orders and quotes? Please write to **sales@numato.com**

Webmaster
Questions/Suggestions about our website? Please write to **webmaster@numato.com**

 Like us on Facebook! https://www.facebook.com/numato

Visit our blog **http://www.numato.cc** for news, updates and specials.

Mailing Address
Numato Systems Pvt Ltd
1st Floor, #56C Wipro Avenue
Phase 1 - Electronic City
Bangalore, KA-560100, India

* Mail orders, phone orders and direct pick up are not available at this time. Please visit our online store to place your order. Estimated shipping time to your address will be displayed in the shopping cart before checkout.

Introduction

MIMAS V2 is a feature packed yet low cost FPGA Development board featuring Xilinx Spartan-6 FPGA. It is specially designed for experimenting and learning system design with FPGAs. This development board features SPARTAN XC6SLX9 CSG324 FPGA with onboard 512Mb DDR SDRAM. The USB 2.0 interface provides fast and easy configuration download to the on-board SPI flash. No need to buy an expensive programmer or special downloader cable to download the bit stream to the board.

Applications

- Product Prototype Development
- Signal Processing
- Learning Digital Electronics
- Educational tool for schools and universities

Board features

- FPGA: Spartan XC6SLX9 in CSG324 package
- DDR: 166MHz 512Mb LPDDR (MT46H32M16LF/W949D6CBHX6E)
- Flash memory: 16 Mb SPI flash memory (M25P16)
- USB 2.0 interface for On-board flash programming
- FPGA configuration via JTAG and USB
- 8 LEDs Six Push Buttons and 8 way DIP switch for user defined purposes
- VGA Connector
- Stereo Jack
- Micro SD Card Adapter
- Three Digit Seven Segment Display.
- 32 IOs for user defined purposes
- Four 6×2 Expansion Connectors
- On-board voltage regulators for single power rail operation

How to use the module

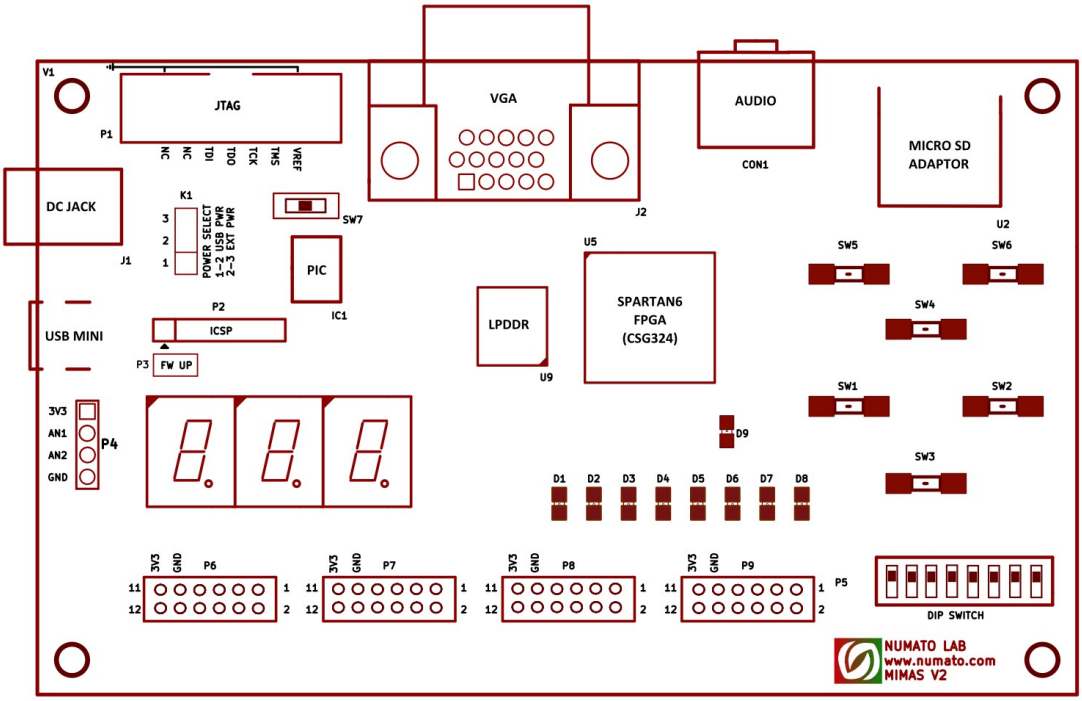
The following section describes how to use this module.

Components/Tools required

Along with the module, you may need the items in the list below for easy and fast installation.

1. USB A to Mini B cable.
2. DC Power supply (Optional).

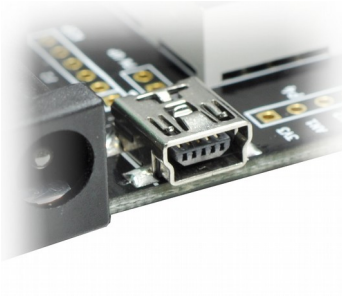
Connection Diagram



This diagram should be used as a reference only. For detailed information, see MIMAS V2 schematics at the end of this documentation. Details of individual connectors are as below.

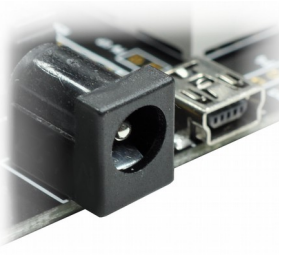
USB Interface

The on board full speed USB controller helps a computer to communicate with this module. Use a USB A to Mini B cable to connect with a PC. By default the module is powered from USB so make sure not to overcrowd unpowered USB hubs.



Visit http://numato.com/cables-accessories to buy cables and accessories for this product.

DC Power Supply



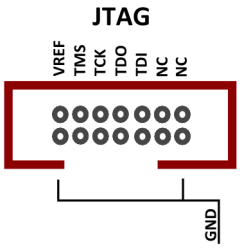
This module uses +5V power supply to function properly. **By default the board is configured to use +5V supply from USB. So an external +5V power is not required unless USB port is unable to supply enough current. In most cases USB ports are capable of providing enough current for the module. Current requirement for this board largely depends on your application. Please consult FPGA datasheet for more details on power requirements.** If for any reason, an external 5V power supply needs to be used for the module, the Power select jumper should be configured properly before connecting the power supply. Please refer to the marking on the board for more details.

Power Select

The Power Select header K1 is used to configure the power source for the board. The jumper in pin 1 and 2 is shorted to switch the power source to on board USB port and pin 2 and 3 to use the external DC power.

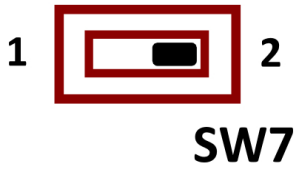
JTAG Connector

JTAG connector provides access to FPGA's JTAG pins. A XILINX platform cable can be used for JTAG programming.



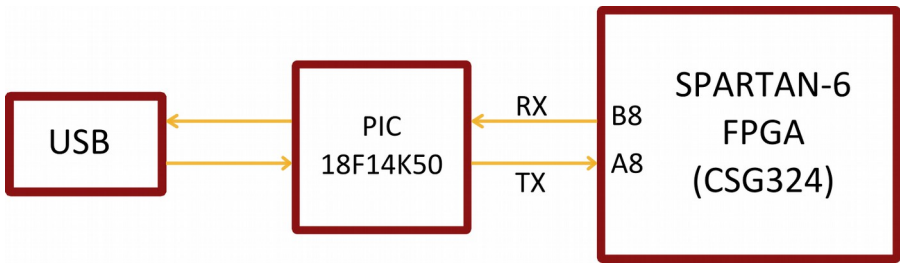
Configuration Mode Selection

Slide switch SW7 is used to switch between the USB configuration mode and UART. Slide the switch to Position 1 to download bit stream through USB configuration tool and Position 2 to use the interface as a UART in order to communicate from your code in FPGA with the PC. By default the board is shipped with slide switch position in USB configuration tool mode.



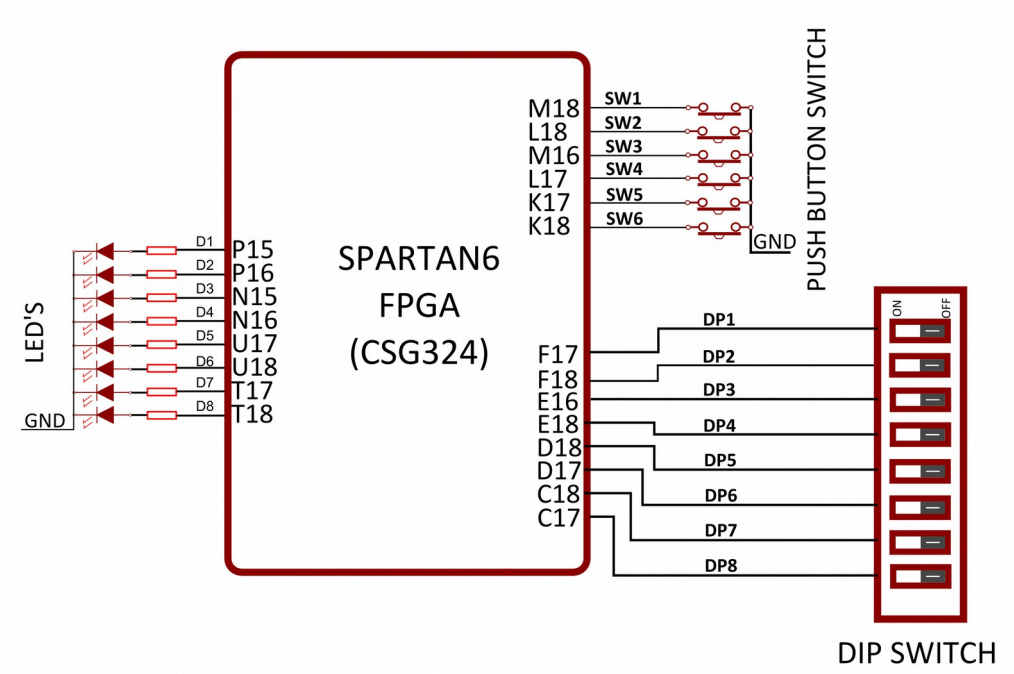
UART

The MIMAS V2 includes USB-UART, which helps to establish the communication between the code in the FPGA and any application running on the PC. Data can be send and received from the FPGA by using Serial Terminal at baud rate 19200.



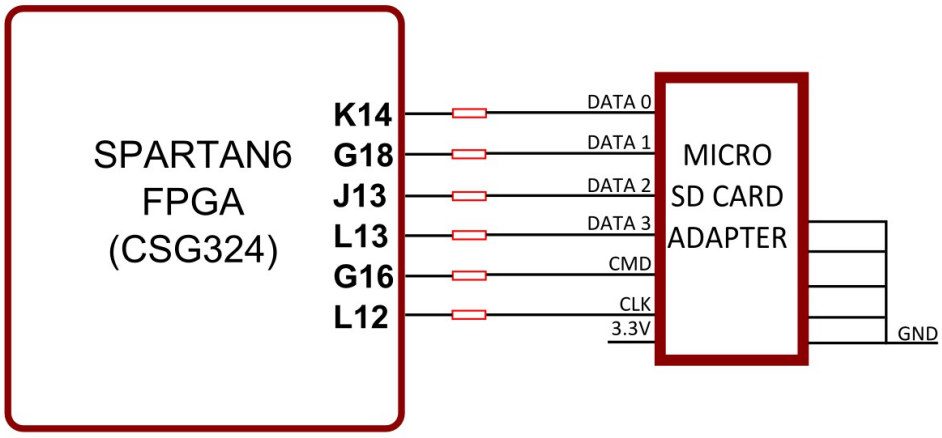
LED, Push Button and Dip Switch

MIMAS V2 has six push button switches, an eight position DIP switch and eight LEDs for human interaction. All switches are directly connected to Spartan 6 FPGA and can be used in your design with minimal effort.



Micro SD

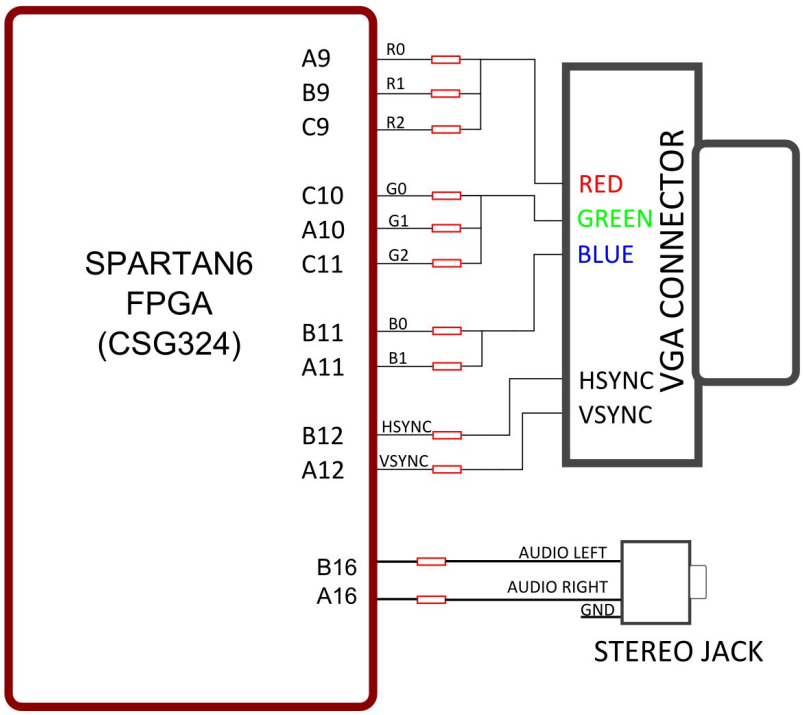
MIMAS V2 features a Micro SD adapter on-board. By installing a Micro SD card, you can add data logging, media storage and other file storage to your design.



VGA and Audio

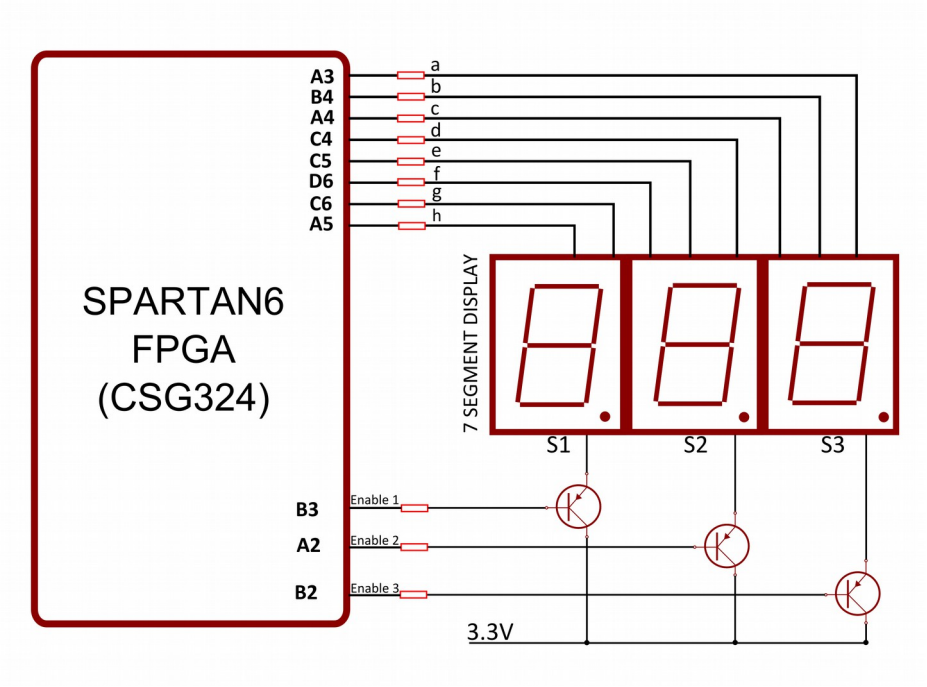
The VGA interface provides this board the ability to generate VGA signals from FPGA and display information any Display/monitor that supports standard VGA connector. This VGA interface uses resistor network based DAC for easy code implementation. This 8 bit VGA interface can display up to 256 colors.

Two IOs on the FPGA are dedicated for generating two channels of audio. Different audio tones can be generated by using PWM and Frequency synthesis.



7Segment LED Display

This board features three 7-segment LED display multiplexed for low pin count operation. Each module can be separately turned on and off with the three switching transistors.



GPIOs

This board is equipped with 32 user IO pins that can be used for various custom applications. Pin assignments on the connectors are available in the tables below.

HEADER P6

Header Pin No.	Pin description	Spartan-6 (CSG324) Pin No.
1	IO_L43P_2	U7
2	IO_L43N_2	V7
3	IO_L63P_2	T4
4	IO_L63N_2	V4
5	IO_L49P_D3_2	U5
6	IO_L49N_D4_2	V5
7	IO_L62P_D5_2	R3
8	IO_L62N_D6_2	T3
9	GND	NA
10	GND	NA
11	VCCAUX	NA
12	VCCAUX	NA

HEADER P7

Header Pin No.	Pin description	Spartan-6 (CSG324) Pin No.
1	IO_L41P_2	U8
2	IO_L41N_VREF_2	V8
3	IO_L31P_GCLK31_D14_2	R8
4	IO_L31N_GCLK30_D15_2	T8
5	IO_L48P_D7_2	R5
6	IO_L48N_RDWR_B_VREF_2	T5
7	IO_L32P_GCLK29_2	T9
8	IO_L32N_GCLK28_2	V9

9	GND	NA
10	GND	NA
11	VCCAUX	NA
12	VCCAUX	NA

HEADER P8

Header Pin No.	Pin description	Spartan-6 (CSG324) Pin No.
1	IO_L16P_2	R11
2	IO_L16N_VREF_2	T11
3	IO_L29P_GCLK3_2	R10
4	IO_L29N_GCLK2_2	T10
5	IO_L14P_D11_2	U13
6	IO_L14N_D12_2	V13
7	IO_L23P_2	U11
8	IO_L23N_2	V11
9	GND	NA
10	GND	NA
11	VCCAUX	NA
12	VCCAUX	NA

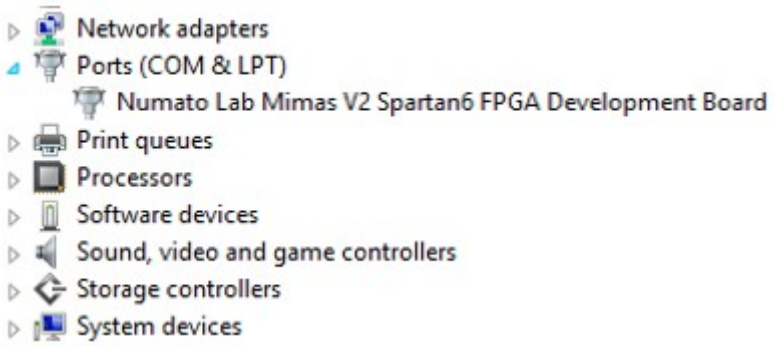
HEADER P9

Header Pin No.	Pin description	Spartan-6 (CSG324) Pin No.
1	IO_L43P_GCLK5_M1DQ4_1	H17
2	IO_L43N_GCLK4_M1DQ5_1	H18
3	IO_L44P_A3_M1DQ6_1	J16
4	IO_L44N_A2_M1DQ7_1	J18
5	IO_L41P_GCLK9_IRDY1_M1RASN_1	K15
6	IO_L41N_GCLK8_M1CASN_1	K16
7	IO_L42P_GCLK7_M1UDM_1	L15

8	IO_L42N_GCLK6_TRDY1_M1LDM_1	L16
9	GND	NA
10	GND	NA
11	VCCAUX	NA
12	VCCAUX	NA

Driver Installation

Windows



This product requires a driver to be installed for proper functioning when used with Windows. The driver package can be downloaded from the product page. To install the driver, unzip the contents of the downloaded driver package to a folder. Attach USB cable to the PC and when asked by Windows device installation wizard, point to the folder where driver files are present. When driver installation is complete, the module should appear in Windows Device Manager as a serial port (see the picture on the right). Note down the name of the serial port (COM1, COM2 etc..). This information is required while programming the module with configuration tool.

Linux

To use this product with Linux, USB CDC driver needs to be compiled in with the kernel. Fortunately, **most Linux distributions (Ubuntu, Redhat, Debian etc..) has this driver pre-installed.** The chances of you requiring to rebuild the kernel to include the USB CDC driver is very slim. When connected to a Linux machine, this product should appear as a serial port in the /dev directory. Usually the name of the device will be “ttyACMx” or similar. The name may be different depending on the Linux distribution you have.

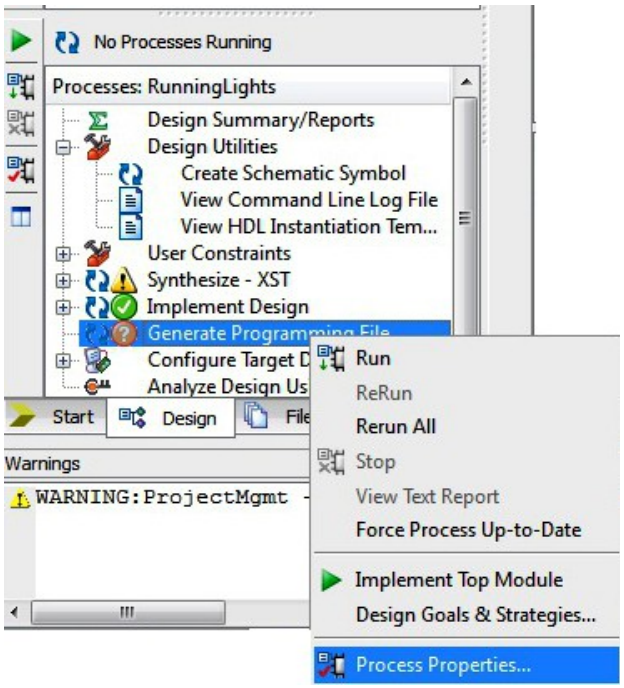
Mac

Similar to Linux, Mac operating system comes with the required drivers pre-installed. When connected to a Mac computer, the device should appear as a serial port.

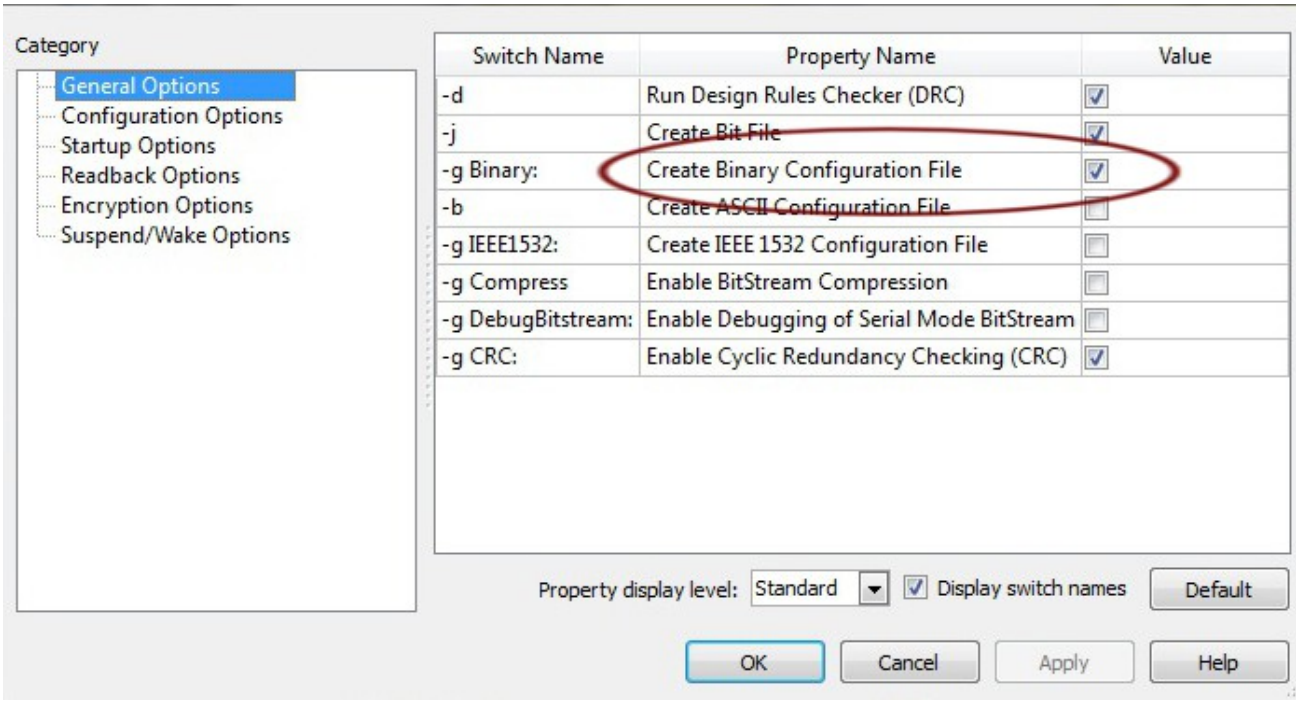
Generating Bit Stream for MIMAS V2

HDL design needs to be converted to bit stream before it can be programmed to FPGA. MIMAS V2 configuration tool at this time accepts only binary (.bin) bit stream created by XILINX ISE (http://www.xilinx.com/tools/webpack.htm). Once the HDL is synthesized, it is easy to create a binary bit stream out of it. Please follow the Steps below to generate binary bit stream from your design using ISE Web Pack.

Step 1: Right click on the “Generate Programming File” option in “Processes” window.



Step 2: Select “Process Properties” from the pop up menu. In the dialog box, check “Create Binary Configuration File” Check box and click “Apply”.



Step 3**:** Click “OK” to close the dialog box. Right click on “Generate Programming File” option again and select “Run”. Now you will be able to find a .bin file in the project directory and that file can be used for MIMAS V2 configuration.

Powering Up MIMAS V2

MIMAS V2 can be powered directly from USB port so make sure that you are using a USB port that can power the board properly. It is recommended to connect the board directly to the PC instead using a hub. It is practically very difficult to estimate the power consumption of the board, as it depends heavily on your design and the clock used. XILINX provides tools to estimate the power consumption. In any case if power from USB is not enough for your application, an external supply can be applied to the board. MIMAS V2 requires two different voltages, a 3.3V and a 1.2V supply. On-board regulators derive these voltages from the USB/Ext power supply.

Configuring MIMAS V2

The MIMAS V2 Spartan6 module can be configured by two methods,

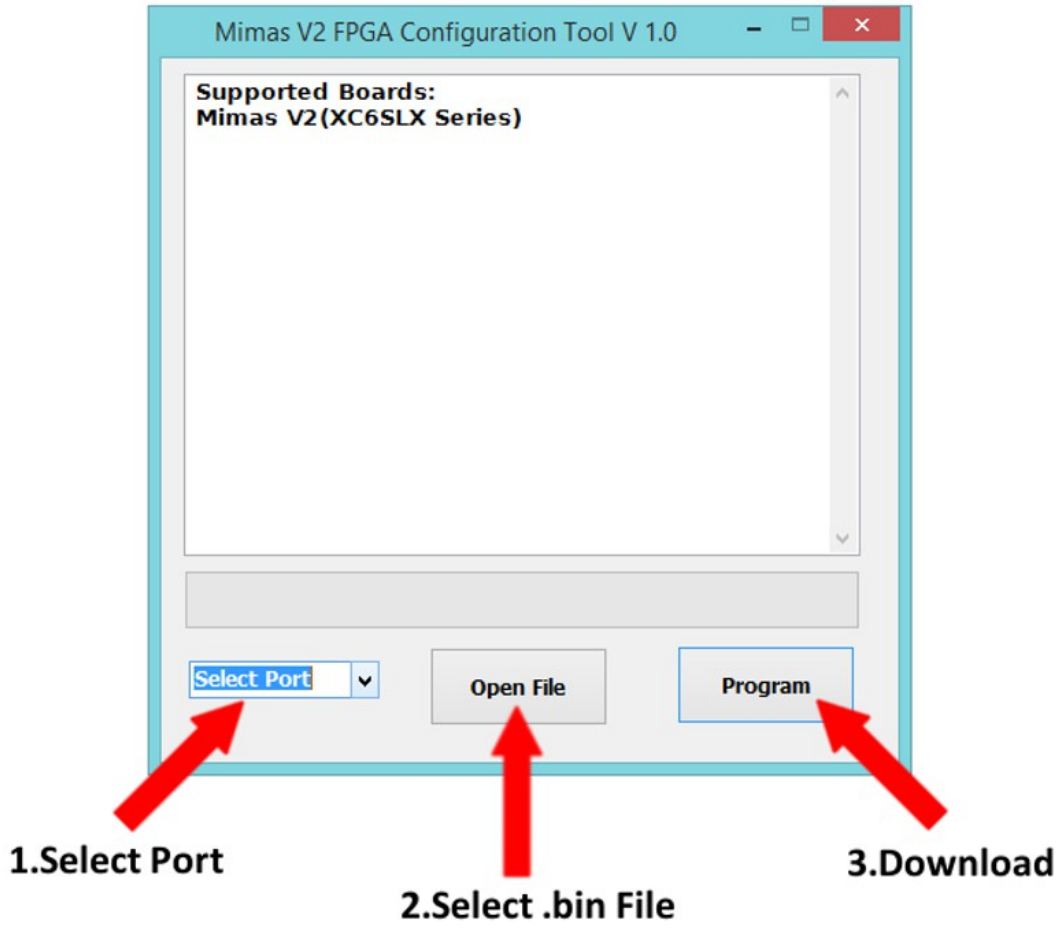
a) Using MIMAS V2 configuration tool through USB.

b) Using the Xilinx programming cable..

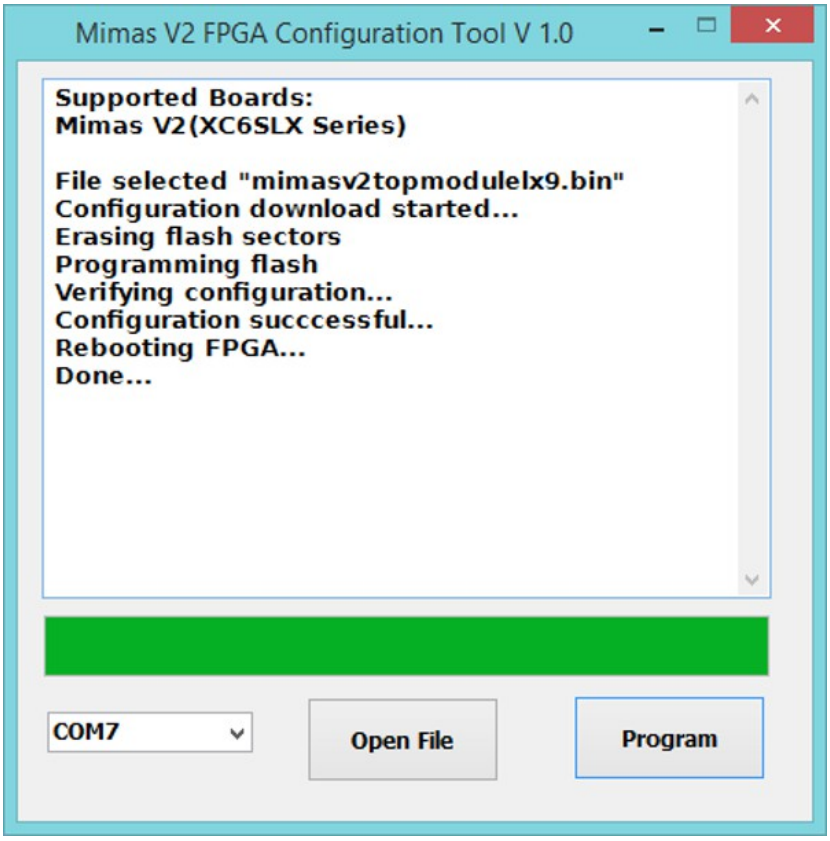
Configuring MIMAS V2 using configuration tool

MIMAS V2 has an on-board micro-controller which facilitates easy reprogramming of on-board SPI flash through USB interface. The micro-controller receives bit stream from the host application and program it in to the SPI Flash and lets the FPGA boot from the flash. The MIMAS V2 configuration application can be downloaded from www.numato.com for free. When MIMAS V2 is connected to PC, it shows up as a COM port in Device Manager. Run configuration application, select correct COM Port before downloading bit stream. Click on “Open File” to select the bit stream file (.bin) and press “Program” button to download the bit stream. Wait till the download process is finished. Once the download process is over, the configuration controller will try to boot the FPGA from the SPI Flash automatically. Follow the below steps.

Step 1: Make sure you have selected USB configuration mode (Slide SW7 to position 1. Refer to the section “Configuration Mode Selection” for more information). Run MIMAS V2 Configuration Tool and select the correct port (Refer to section “Driver installation” for more information on finding port number). Click Open file button and select the .bin file.



Step 2: Click on “Program” button. Wait till “Done” appears on the screen.

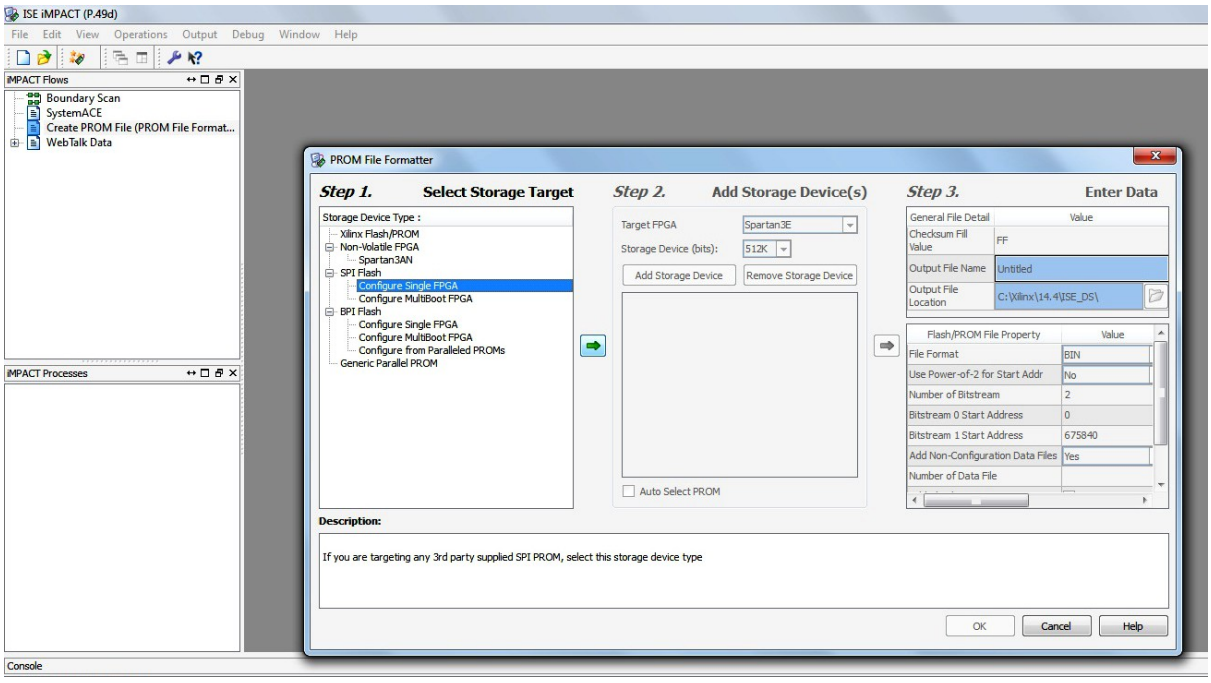


Configuring MIMAS V2 using JTAG

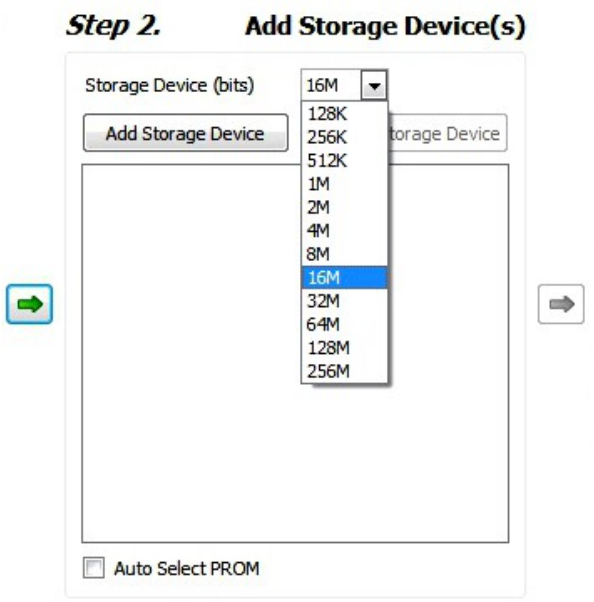
MIMAS V2 Spartan6 module features an on-board JTAG connector which facilitates easy reprogramming of SRAM and on-board SPI flash through JTAG programmer like “XILINX Platform-cable usb”. Programming MIMAS V2 using JTAG requires “XILINX ISE iMPACT” software which is bundled with XILINX ISE Design Suite. To program SPI flash we need a ".mcs" file which needs to be generated from the ".bit" file. Steps for generating ".mcs" file is discussed below. Programming FPGA SRAM does not require a mcs file to be generated.

Generating ".mcs" file for MIMAS V2

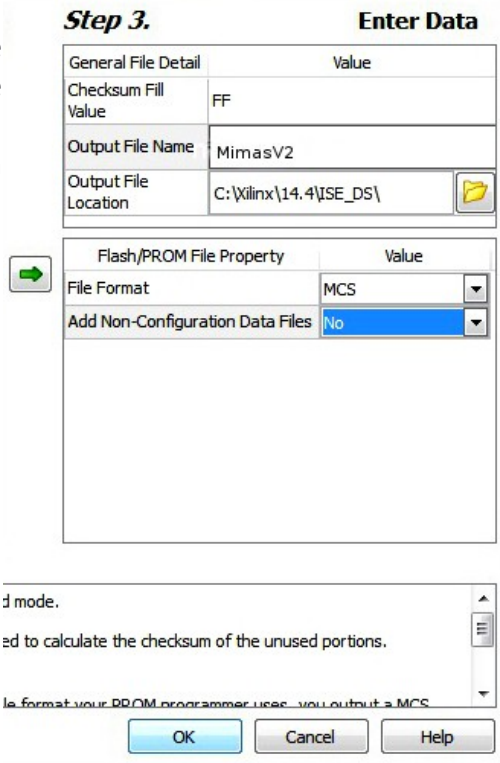
Step 1: Open ISE iMPACT. Click on “Create PROM file(PROM file formatter)”.In the dialog box, select “Configure Single FPGA” in storage device type. Then click on the green arrow at the right side.



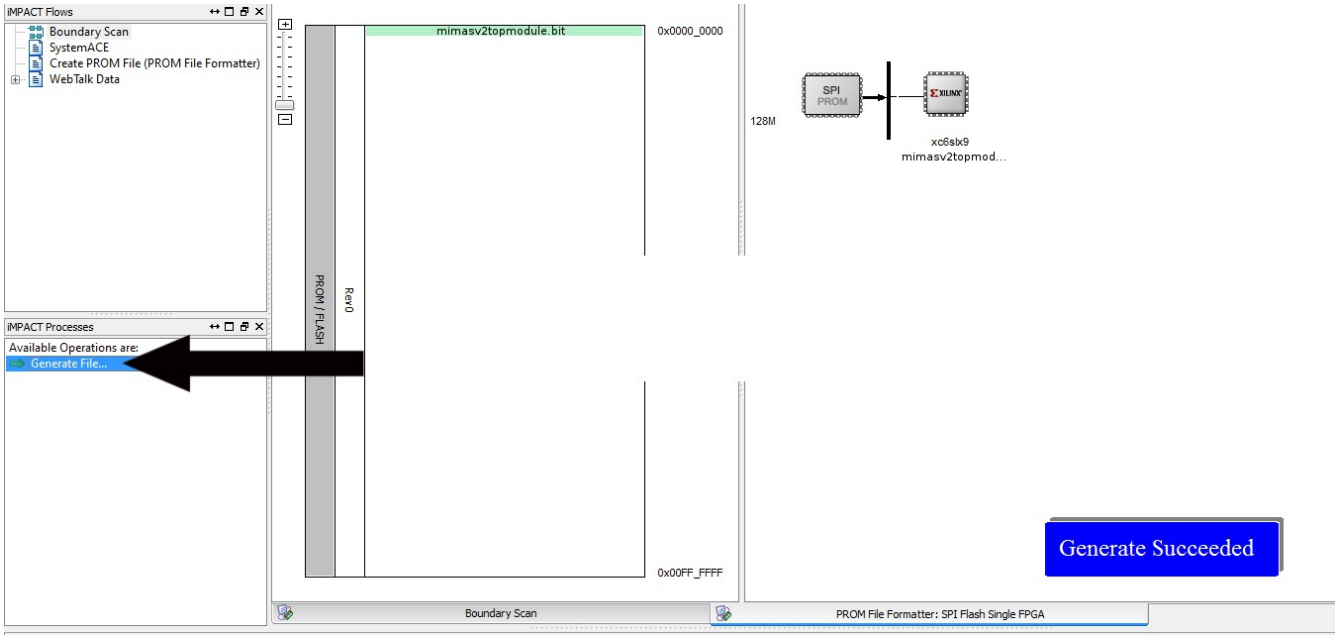
Step 2: Select 16M in Storage Device (bits) list. Now click on “Add Storage Device”, then the green arrow at the right side.



Step 3: Set an output file name and an output file location (the ".mcs" file will be generated at this location which will be required later for programming the FPGA), then click OK twice, then select the ".bit" file we already generated then click Open and click NO when it prompts to add another device file.

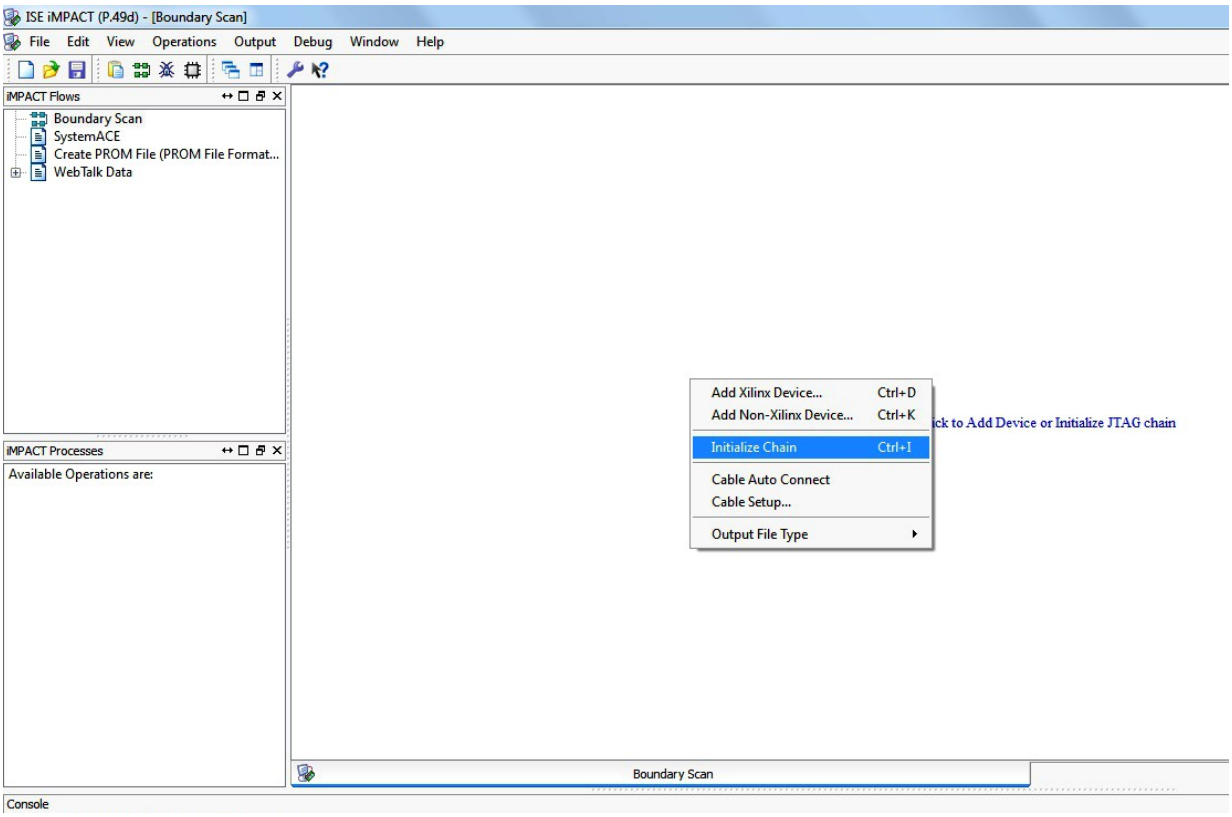


Step 4: Double click on “Generate File”. A message “Generate Succeeded” will be displayed as shown in fig below if the mcs file is generated successfully.

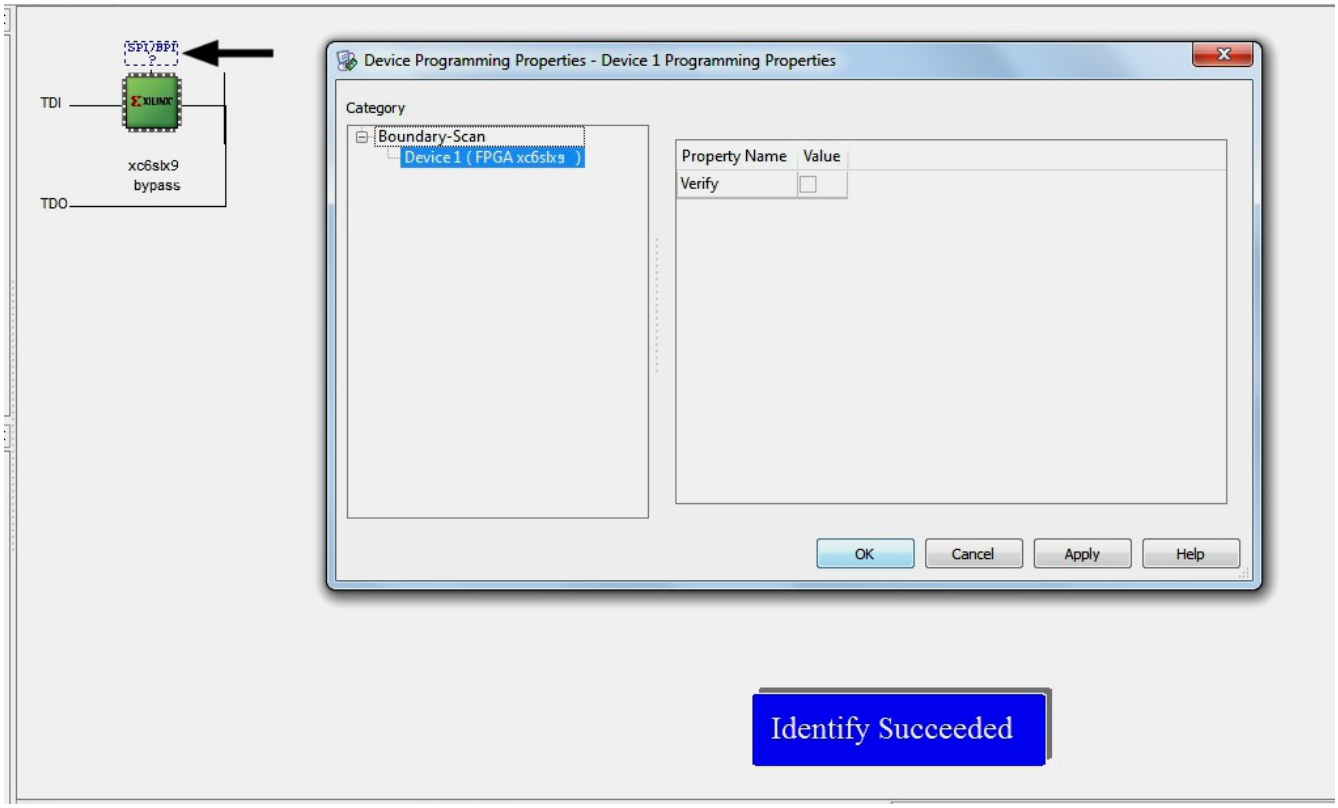


Programming onboard SPI flash using ISE iMPACT

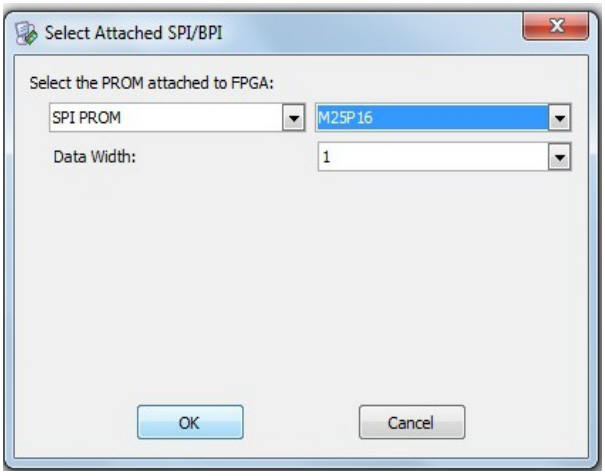
Step 1: Make sure Xilinx Platform Cable USB is connected properly to the board. Open ISE iMPACT. Click on "Boundary Scan" in the iMPACT flows window in the left top corner. Then right click on the window panel in the right side. Select "Initialize Chain".



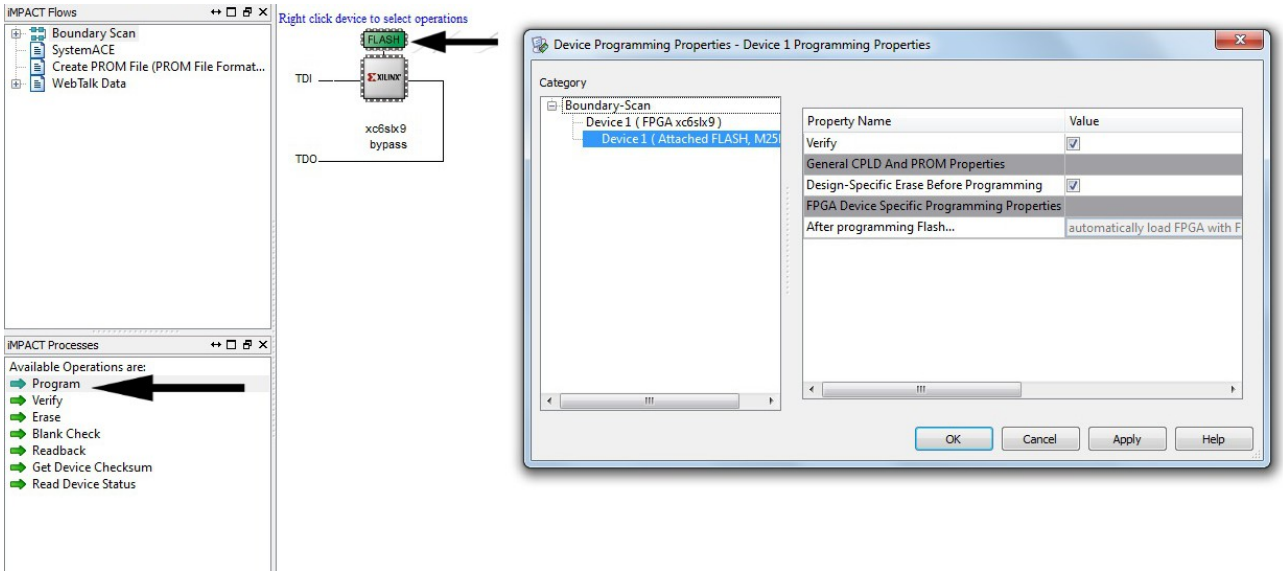
Step 2: If the device is detected properly you will get a pop up window as shown below, Click OK. Then right click on the SPI/BPI (next to the black arrow in the below fig.), select Add SPI/BPI Flash.



Step 3: Select the ".mcs" file we already created and click OK. Now choose “M25P16” in the dialogue box appeared, then click OK.



Step 4: Click on “Flash”, Double Click on Program, select OK. If the programming is successful, a confirmation message will be displayed.

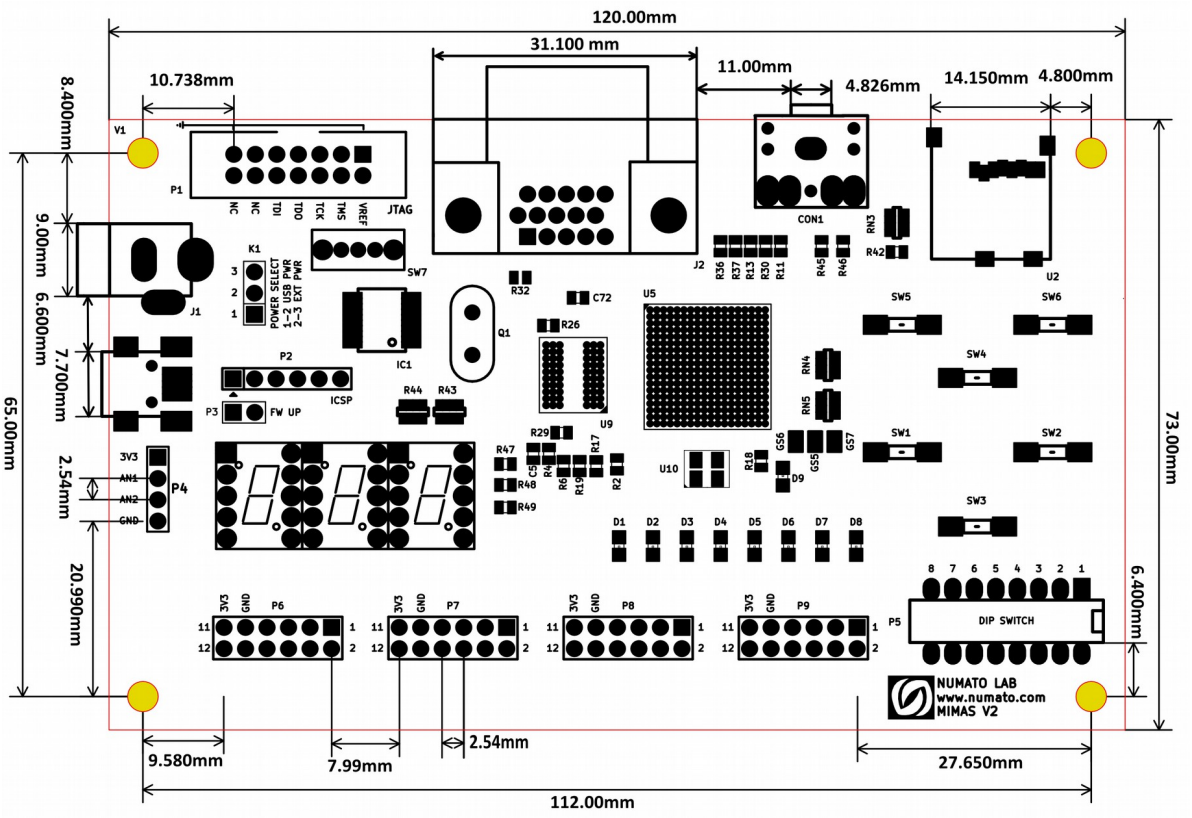


Technical Specifications

Parameter *	Value	Unit
Basic Specifications		
Number of GPIOs	32	
Number of LEDs	8	
Number of Push Buttons	6	
SPI Flash Memory (M25P16)	16	Mb
Power supply voltage (USB or external)	5 - 7	V
FPGA Specifications		
Internal supply voltage relative to GND	–0.5 to 1.25	V
Auxiliary supply voltage relative to GND	–0.5 to 3.75	V
Output drivers supply voltage relative to GND	–0.5 to 3.75	V

* All parameters considered nominal. Numato Systems Pvt Ltd reserve the right to modify products without notice.

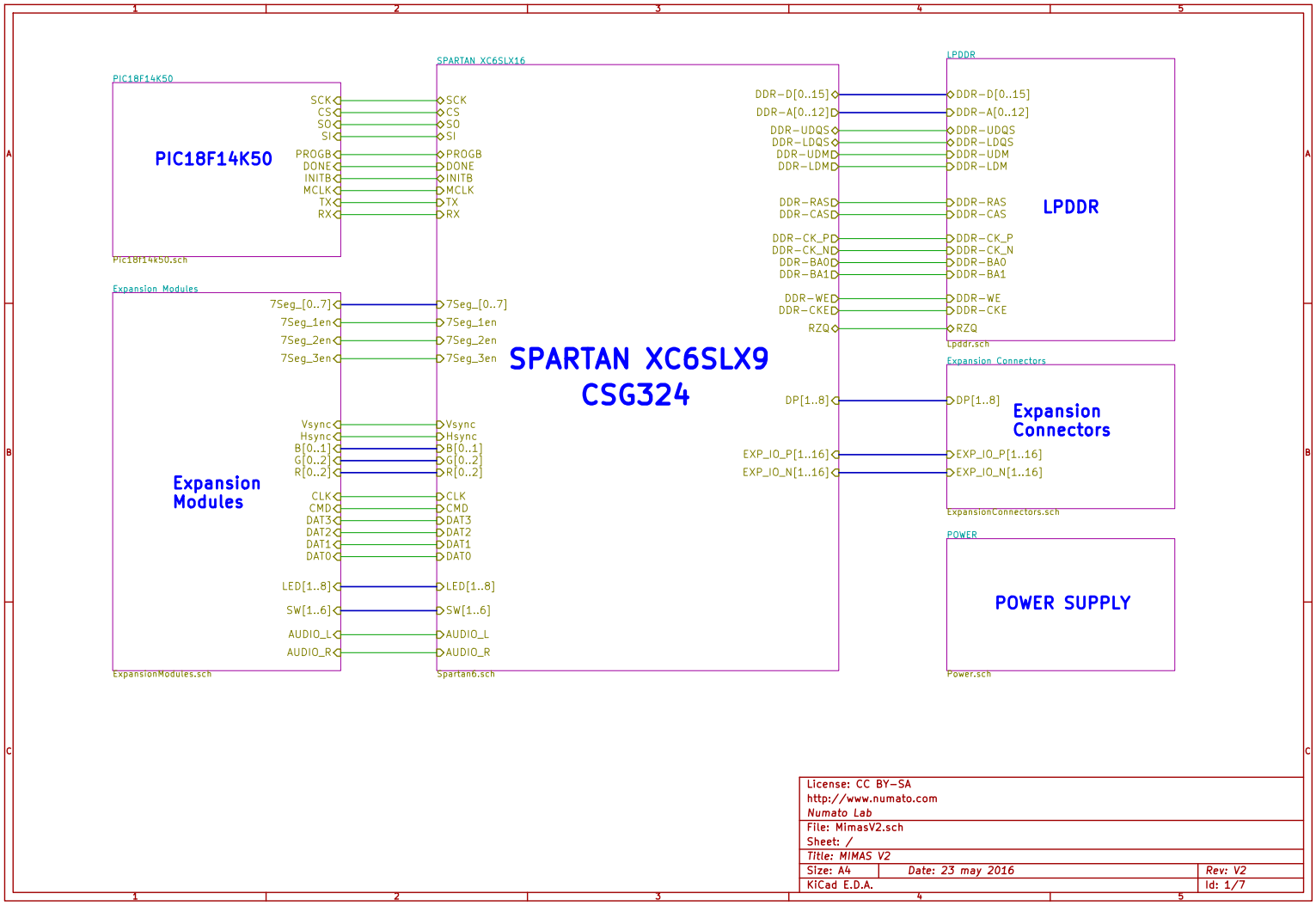
Physical Dimensions

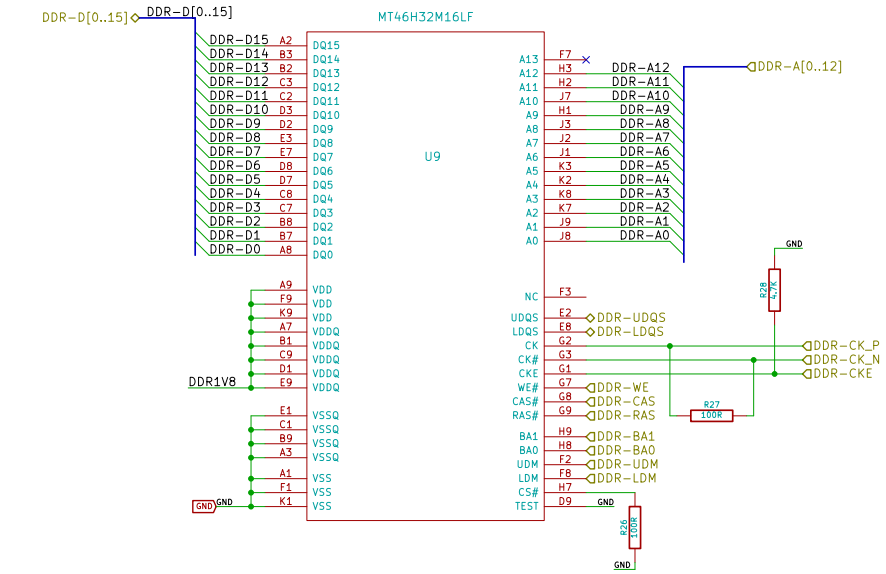


L x W x H : 120.00 mm x73.00 mm x 17 mm
Mechanical Hole Diameter : 3.2 mm

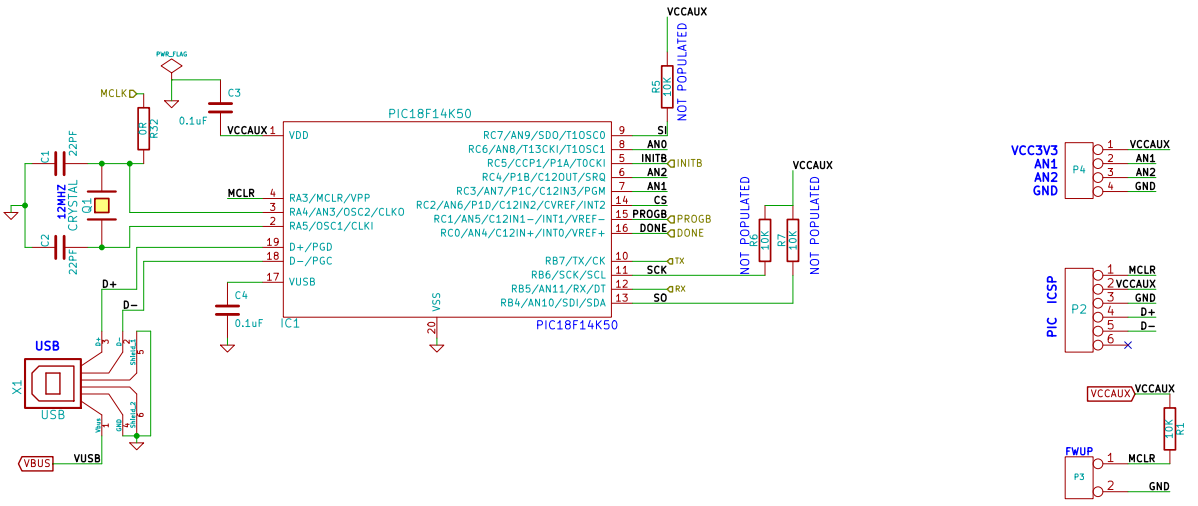
Schematics

See next page.

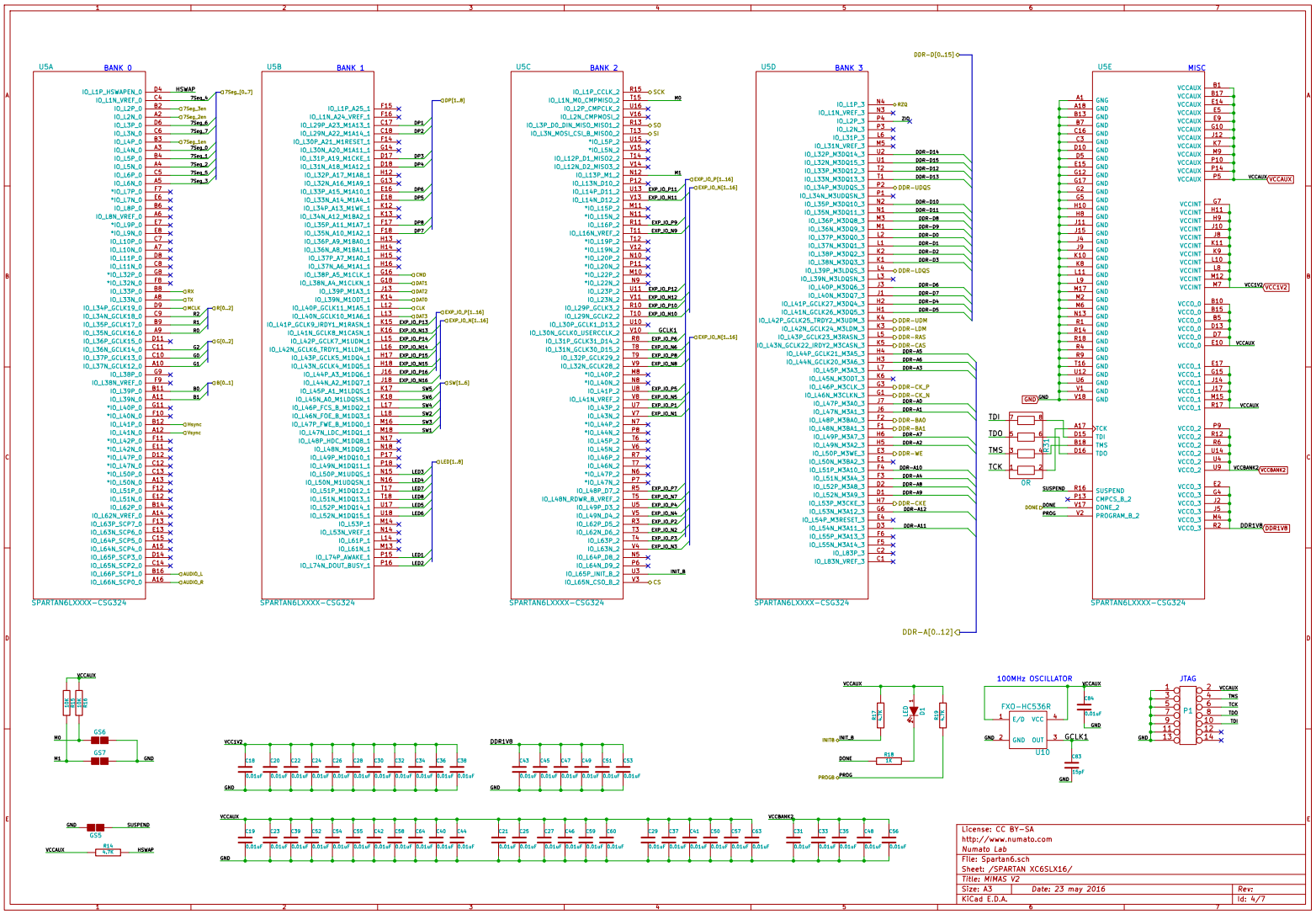

PIC18F14K50
SCK
CS
SO
SI
PROGB
DONE
INITB
MCLK
TX
RX
PIC18F14K50
Pic18f14k50.sch
SPARTAN XC6SLX16
SCK
CS
SO
SI
PROGB
DONE
INITB
MCLK
TX
RX
SPARTAN XC6SLX9
CSG324
Spartan6.sch
Expansion Modules
7Seg_[0..7]
7Seg_1en
7Seg_2en
7Seg_3en
Vsync
Hsync
B[0..1]
G[0..2]
R[0..2]
CLK
CMD
DAT3
DAT2
DAT1
DAT0
LED[1..8]
SW[1..6]
AUDIO_L
AUDIO_R
Expansion Modules
ExpansionModules.sch
7Seg_[0..7]
7Seg_1en
7Seg_2en
7Seg_3en
Vsync
Hsync
B[0..1]
G[0..2]
R[0..2]
CLK
CMD
DAT3
DAT2
DAT1
DAT0
LED[1..8]
SW[1..6]
AUDIO_L
AUDIO_R
LPDDR
DDR-D[0..15]
DDR-A[0..12]
DDR-UDQS
DDR-LDQS
DDR-UDM
DDR-LDM
DDR-RAS
DDR-CAS
DDR-CK_P
DDR-CK_N
DDR-BA0
DDR-BA1
DDR-WE
DDR-CKE
RZQ
LPDDR
Lpddr.sch
DDR-D[0..15]
DDR-A[0..12]
DDR-UDQS
DDR-LDQS
DDR-UDM
DDR-LDM
DDR-RAS
DDR-CAS
DDR-CK_P
DDR-CK_N
DDR-BA0
DDR-BA1
DDR-WE
DDR-CKE
RZQ
Expansion Connectors
DP[1..8]
EXP_IO_P[1..16]
EXP_IO_N[1..16]
Expansion Connectors
ExpansionConnectors.sch
DP[1..8]
EXP_IO_P[1..16]
EXP_IO_N[1..16]
POWER
POWER SUPPLY
Power.sch
License: CC BY-SA
http://www.numato.com
Numato Lab
File: MimasV2.sch
Sheet: /
Title: MIMAS V2
Size: A4
Date: 23 may 2016
Rev: V2
KiCad E.D.A.
Id: 1/7

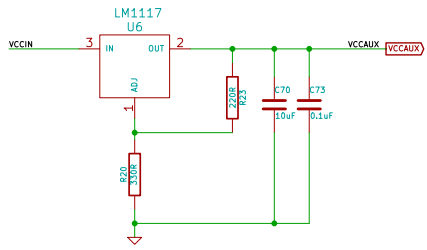


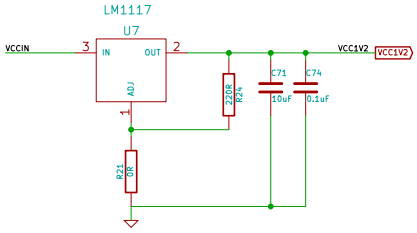
File: Lpddr.sch		
Sheet: /LPDDR/		
Title: MIMAS V2		
Size: A4	Date: 23 may 2016	Rev:
KiCad E.D.A.		Id: 2/7

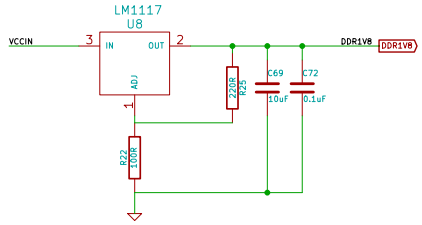


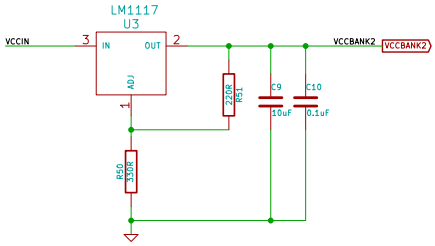
License: CC BY-SA		
http://www.numato.com		
Numato Lab		
File: Pic18f14k50.sch		
Sheet: /PIC18F14K50/		
Title: MIMAS V2		
Size: A4	Date: 23 may 2016	Rev:
KiCad E.D.A.		Id: 3/7

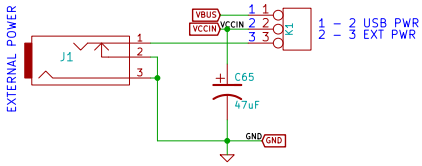

U5A
BANK 0
SPARTAN6LXXXX-CSG324
U5B
BANK 1
SPARTAN6LXXXX-CSG324
U5C
BANK 2
SPARTAN6LXXXX-CSG324
U5D
BANK 3
SPARTAN6LXXXX-CSG324
U5E
MISC
SPARTAN6LXXXX-CSG324
DDR-D[0..15]
DDR-A[0..12]
100MHz OSCILLATOR
FXO-HC536R
U10
JTAG
P1
VCCAUX
VCC1V2
DDR1V8
VCCBANK2
License: CC BY-SA
http://www.numato.com
Numato Lab
File: Spartan6.sch
Sheet: /SPARTAN XC6SLX16/
Title: MIMAS V2
Size: A3
Date: 23 may 2016
Rev:
KiCad E.D.A.
Id: 4/7



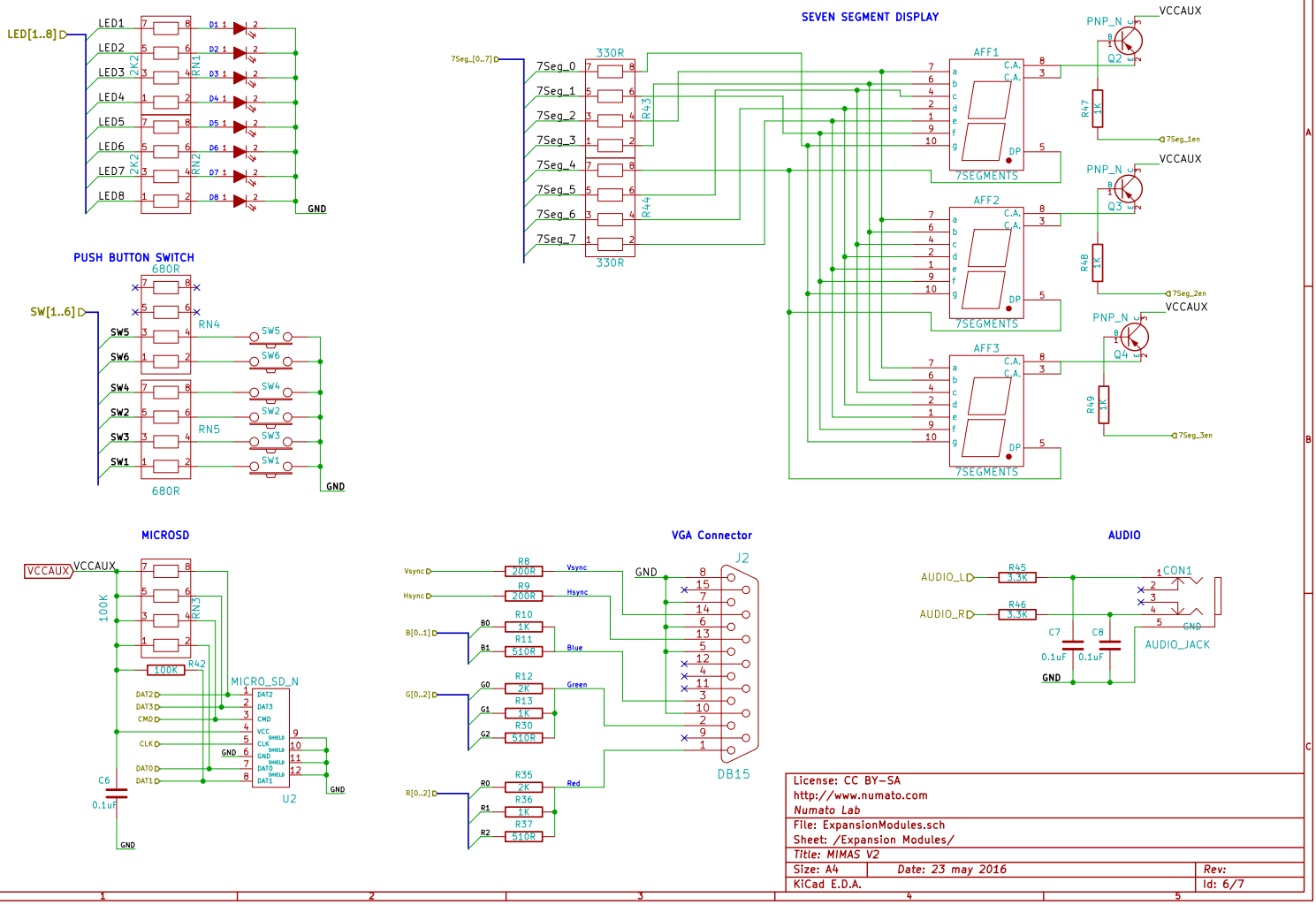


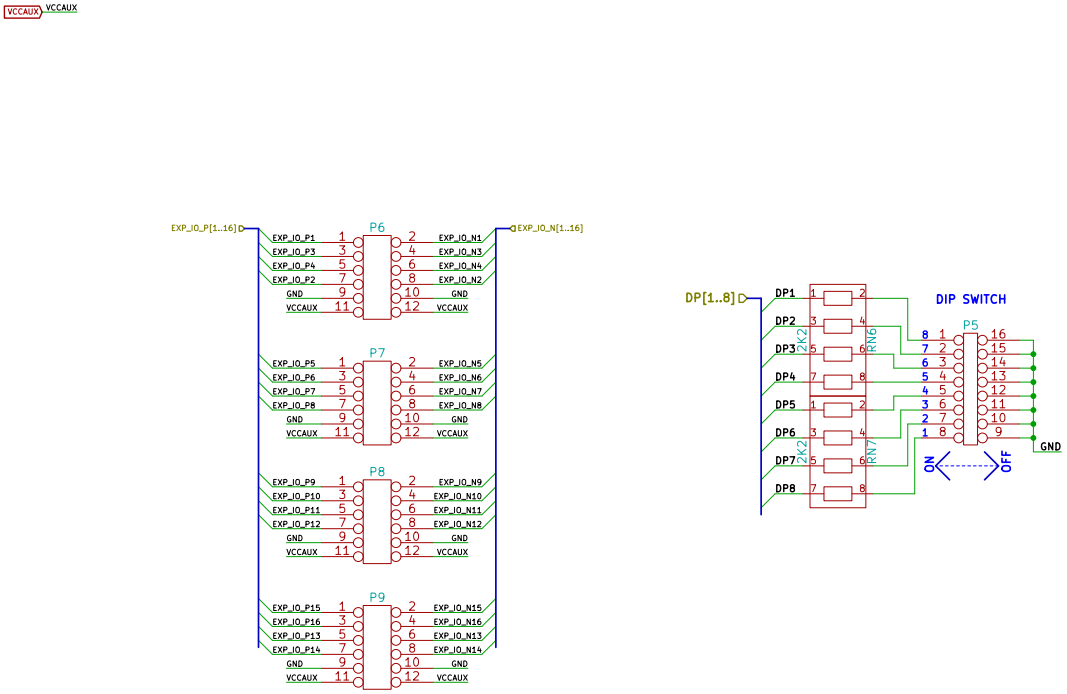






License: CC BY-SA http://www.numato.com *Numato Lab*		
File: Power.sch Sheet: /POWER/		
Title: MIMAS V2		
Size: A4	*Date: 23 may 2016*	*Rev:*
KiCad E.D.A.		Id: 5/7


LED[1..8]
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
2K2
RN1
RN2
GND
PUSH BUTTON SWITCH
SW[1..6]
680R
RN4
RN5
SW5
SW6
SW4
SW2
SW3
SW1
SEVEN SEGMENT DISPLAY
7Seg_[0..7]
330R
R43
R44
7Seg_0
7Seg_1
7Seg_2
7Seg_3
7Seg_4
7Seg_5
7Seg_6
7Seg_7
AFF1
AFF2
AFF3
7SEGMENTS
C.A.
DP
PNP_N
Q2
Q3
Q4
VCCAUX
R47
R48
R49
1K
7Seg_1en
7Seg_2en
7Seg_3en
MICROSD
VCCAUX
100K
RN3
R42
MICRO_SD_N
DAT2
DAT3
CMD
CLK
DAT0
DAT1
GND
C6
0.1uF
U2
VGA Connector
J2
DB15
Vsync
Hsync
B[0..1]
G[0..2]
R[0..2]
R8
R9
R10
R11
R12
R13
R30
R35
R36
R37
200R
1K
2K
510R
Blue
Green
Red
AUDIO
AUDIO_L
AUDIO_R
R45
R46
3.3K
C7
C8
0.1uF
CON1
AUDIO_JACK
License: CC BY-SA
http://www.numato.com
Numato Lab
File: ExpansionModules.sch
Sheet: /Expansion Modules/
Title: MIMAS V2
Size: A4
Date: 23 may 2016
Rev:
KiCad E.D.A.
Id: 6/7



License: CC BY-SA		
http://www.numato.com		
Numato Lab		
File: ExpansionConnectors.sch		
Sheet: /Expansion Connectors/		
Title: MIMAS V2		
Size: A4	*Date: 23 may 2016*	*Rev:*
KiCad E.D.A.		Id: 7/7