GUJARAT TECHNOLOGICAL UNIVERSITY

Subject Name: Internship/ Project

Subject Code: 3181101

Internship Report
On
'VLSI'
At
Indo German Tool Room

Submitted to
Institute Code:017
Institute Name: Vishwakarma Government Engineering College

Under the Guidance of
Prof. Jagruti K. Naik (Assistant Professor)

In partial Fulfilment of the Requirement of the award of the degree of
Bachelor of Engineering
Offered By
Gujarat Technological University, Ahmedabad

Prepared by:

Kapadiya Harshkumar Girishkumar
210170111128

ELECTRONICS & COMMUNICATION Semester - VIII)
Month & Year:
June/July 2024

Abstract

This internship project focuses on foundational learning and practical implementation within the realm of Very-Large-Scale Integration (VLSI) and digital electronics. The primary objective is to gain a comprehensive understanding of basic digital electronics concepts and apply this knowledge through the implementation of digital circuits using DSCH software. The project encompasses the design, simulation, and analysis of various digital circuits, providing hands-on experience with digital design methodologies. This initiative not only strengthens theoretical understanding but also enhances practical skills essential for a career in VLSI design and digital electronics.

Acknowledgement

I would like to express my deepest gratitude to my external mentor, Jatin Sir for their invaluable guidance, support, and expertise throughout this internship. Their insights and encouragement have been instrumental in the successful completion of this project. I am equally thankful to my internal mentor, Prof. Jagruti K. Naik whose continuous support, constructive feedback, and mentorship have significantly contributed to my learning and growth in the field of VLSI and digital electronics. I also extend my appreciation to Indo German Tool Room for providing me with this opportunity and resources necessary for this internship.

Index

Figures Of Topic 3:

Figures Of Topic 4:

Intoduction To Digital Electronics:

Digital electronics is a field of electronics involving the study and use of digital signals for the design and operation of electronic devices and systems. Unlike analog electronics, which deals with continuous signals, digital electronics deals with discrete signals, typically represented by binary numbers (0s and 1s). This field forms the backbone of modern electronic devices, including computers, smartphones, and digital communication systems.

Types of Digital Electronics Components :

1. **Combinational Logic Circuits**

   - **Description**: These circuits do not have memory and their output is solely determined by the current inputs.
   - **Example**: An adder circuit.
     - *Half Adder*: A circuit that adds two single-bit binary numbers and produces a sum and a carry output.
     - *Full Adder*: A circuit that adds three single-bit binary numbers (two significant bits and a carry bit) and produces a sum and a carry output.

2. **Sequential Logic Circuits**

   - **Description**: These circuits have memory elements and their output depends on the current inputs as well as the past sequence of inputs.
   - **Example**: A flip-flop.
     - *D Flip-Flop*: A circuit that captures the value of the D input at a definite portion of the clock cycle and outputs this value until the next clock cycle.

3. **Microprocessors and Microcontrollers**

   - **Description**: Microprocessors are the central processing units of a computer, performing arithmetic and logic operations. Microcontrollers integrate a microprocessor with peripheral devices like memory, input/output interfaces, etc.
   - **Example**: An 8051 microcontroller.
     - *8051 Microcontroller*: A widely used microcontroller in embedded systems, known for its simplicity and versatility in controlling various electronic devices.

4. **Programmable Logic Devices (PLDs)**

- **Description**: These are digital devices used to build reconfigurable digital circuits. Unlike fixed-function circuits, PLDs can be programmed to perform a wide variety of logic functions.
- **Example**: Field-Programmable Gate Array (FPGA).
  - *FPGA*: A semiconductor device that can be programmed to perform complex logic operations. It consists of an array of configurable logic blocks connected by programmable interconnects.

5. **Digital Signal Processors (DSPs)**

- **Description**: Specialized microprocessors designed for the efficient handling of digital signals. They perform mathematical operations like addition, subtraction, multiplication, and division very quickly.
- **Example**: Texas Instruments TMS320.
  - *TMS320*: A family of DSPs known for their high performance and efficiency in real-time signal processing applications.

Combinational Circuits:

Combinational circuits are types of digital circuits where the output depends solely on the current inputs. These circuits do not have any memory element, meaning they do not store any past input information. The behavior of combinational circuits can be described using truth tables, Boolean algebra, or logic gates.

Examples of Combinational Circuits:

Adders:

- **Half Adder**: Adds two single-bit binary numbers, producing a sum and a carry output.
- **Full Adder**: Adds three single-bit binary numbers (two significant bits and a carry bit), producing a sum and a carry output.

Subtractor:

- **Half Subtractor**: Subtracts two single-bit binary numbers, producing a difference and a borrow output.

- **Full Subtractor**: Subtracts two single-bit binary numbers along with a borrow input, producing a difference and a borrow output.

Multiplexer (MUX):

- Selects one of several input signals and forwards the selected input to a single output line.

Demultiplexer (DEMUX):

- Takes a single input signal and selects one of many data-output-lines, which is connected to the input.

Encoder:

- Converts information from one format or code to another, usually from binary to decimal.

Decoder:

- Converts encoded data back into its original format, usually from binary to decimal.

Comparators:

- Compares two binary numbers and determines their relative magnitude, producing outputs that indicate whether one number is greater than, less than, or equal to the other.

Logic Gates:

- Basic building blocks of combinational circuits, including AND, OR, NOT, NAND, NOR, XOR, and XNOR gates.

Sequential Circuits:

Sequential circuits are types of digital circuits where the output depends on the current inputs and the history of past inputs. These circuits include memory elements, which can store information, allowing them to have states. The behavior of sequential circuits can be described using state diagrams, state tables, and flip-flops.

Examples of Sequential Circuits:

Flip-Flops:

- **SR Flip-Flop (Set-Reset)**: A basic memory element that has two states, set and reset.
- **D Flip-Flop (Data or Delay)**: Captures the value of the input data at a definite portion of the clock cycle and holds this value until the next clock cycle.
- **JK Flip-Flop**: A more versatile flip-flop that can toggle its state, set, reset, or hold its value based on input conditions.
- **T Flip-Flop (Toggle)**: Toggles its output on each clock cycle if the input (T) is high.

Latches:

- **SR Latch**: A simple storage element that holds its state until the inputs change.
- **D Latch**: Stores the input data when the enable signal is active.

Counters:

- **Binary Counter**: Counts in binary, often used for counting events or dividing frequencies.
- **Decade Counter**: Counts from 0 to 9 and then resets to 0, used in digital clocks and similar applications.

Shift Registers:

- **Serial-in Serial-out (SISO)**: Shifts data serially through the register.
- **Serial-in Parallel-out (SIPO)**: Shifts data serially in and outputs data in parallel.
- **Parallel-in Serial-out (PISO)**: Loads data in parallel and shifts it out serially.
- **Parallel-in Parallel-out (PIPO)**: Loads and outputs data in parallel.

Finite State Machines (FSMs):

- **Moore Machine**: The output depends only on the current state.
- **Mealy Machine**: The output depends on the current state and the current inputs.

Introduction to DSCH Software:

DSCH (Digital Schematic) software is a versatile tool used for the design and simulation of digital circuits. Developed by Microelectronics Research & Development Corporation (MICROWIND), DSCH provides an intuitive graphical interface for creating digital schematics and testing their functionality. It is particularly popular in educational environments and among digital design engineers for its ease of use and comprehensive features.

Components of DSCH Software:

Schematic Editor:

- Allows users to create digital circuit schematics using a wide variety of logic gates and components such as AND, OR, NOT gates, flip-flops, multiplexers, and more.
- Provides drag-and-drop functionality for easy component placement and connection.

Simulation Engine:

- Enables the simulation of digital circuits to verify their logical behavior.
- Supports timing analysis and waveform generation to visualize signal changes over time.

Library of Components:

- Includes a comprehensive library of predefined digital components.
- Users can also create custom components and add them to the library.

User Interface:

- Intuitive and user-friendly, making it easy for both beginners and advanced users to design and simulate circuits.
- Provides tools for zooming, panning, and navigating large schematics.

Integration with MICROWIND:

- DSCH can be integrated with MICROWIND software for further analysis and layout generation of digital circuits at the transistor level.
- Allows for a seamless transition from digital design to physical layout.

Benefits of DSCH Software:

Ease of Use:

- The graphical interface simplifies the process of designing and simulating digital circuits, making it accessible to students and professionals alike.

Comprehensive Component Library:

- The extensive library of digital components helps users quickly build complex circuits without the need to create basic elements from scratch.

Simulation Capabilities:

- Real-time simulation helps users identify and correct errors in their designs before moving to physical implementation.

Educational Value:

- DSCH is widely used in academic settings to teach digital logic design, providing a hands-on learning experience for students.

Cost-Effective:

- Compared to other professional-grade digital design tools, DSCH is relatively affordable, making it a popular choice for educational institutions and small-scale projects.

Applications of DSCH Software:

Educational Purposes:

- Used extensively in universities and colleges to teach students the fundamentals of digital electronics and circuit design.

Prototyping and Design Verification:

- Engineers use DSCH for the initial design and verification of digital circuits before committing to more complex and costly physical implementations.

Research and Development:

- Facilitates the development of new digital components and circuits in research labs.

Small-Scale Digital Projects:

- Ideal for hobbyists and small-scale projects where cost and ease of use are critical factors.

Integration with VLSI Design:

- Used as a preliminary design tool before moving to VLSI design and physical layout using tools like MICROWIND.

Circuit Implementation Using Dsch Software:

Implementing a full subtractor circuit in DSCH software involves designing the schematic using basic logic gates and testing its functionality through simulation. Here are the steps to create a full subtractor in DSCH:

Steps to Implement a Full Subtractor in DSCH Software

1. Open DSCH Software

- Launch the DSCH software on your computer.

2. Create a New Project

- Start a new project by selecting File > New.

3. Place Input and Output Terminals

- Add three input terminals for the binary inputs A, B, and Borrow_in.
- Add two output terminals for Difference and Borrow_out.

4. Place Logic Gates

- To construct a full subtractor, you need XOR, AND, and OR gates. Place the following gates:
  - Two XOR gates for calculating the difference.
  - Two AND gates and one OR gate for calculating the borrow output.

5. Connect the Gates to Form a Full Subtractor

- The logic equations for a full subtractor are:
  - **Difference (D)** = $A \oplus B \oplus \text{Borrow\_in}$
  - **Borrow_out (B_out)** = $(A' \wedge B) \vee (A' \wedge \text{Borrow\_in}) \vee (B \wedge \text{Borrow\_in})$

- **Difference Calculation**:

  a. Connect A and B to the first XOR gate to get $A \oplus B$.
  b. Connect the output of the first XOR gate and Borrow_in to the second XOR gate to get Difference.

- **Borrow_out Calculation**:

  c. Connect A to a NOT gate to get A'.
  d. Connect A' and B to the first AND gate to get A' ∧ B.
  e. Connect A' and Borrow_in to the second AND gate to get A' ∧ Borrow_in.
  f. Connect B and Borrow_in to the third AND gate to get B ∧ Borrow_in.
  g. Connect the outputs of the three AND gates to an OR gate to get Borrow_out.

6. Connect the Outputs to Output Terminals

- Connect the output of the second XOR gate to the Difference terminal.
- Connect the output of the OR gate to the Borrow_out terminal.

7. Label and Arrange Components

- Label all the components and terminals for clarity.
- Arrange the components neatly to make the schematic easy to understand.

8. Save Your Project

- Save your project by selecting File > Save and give it a suitable name.

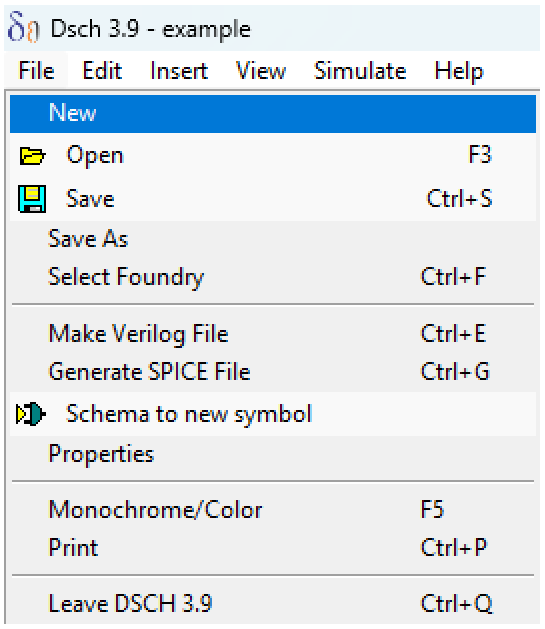
9. Simulate the Circuit

- Run the simulation by selecting Simulation > Start or by pressing the simulation button.
- Apply different input combinations to A, B, and Borrow_in to verify that the Difference and Borrow_out outputs are correct.

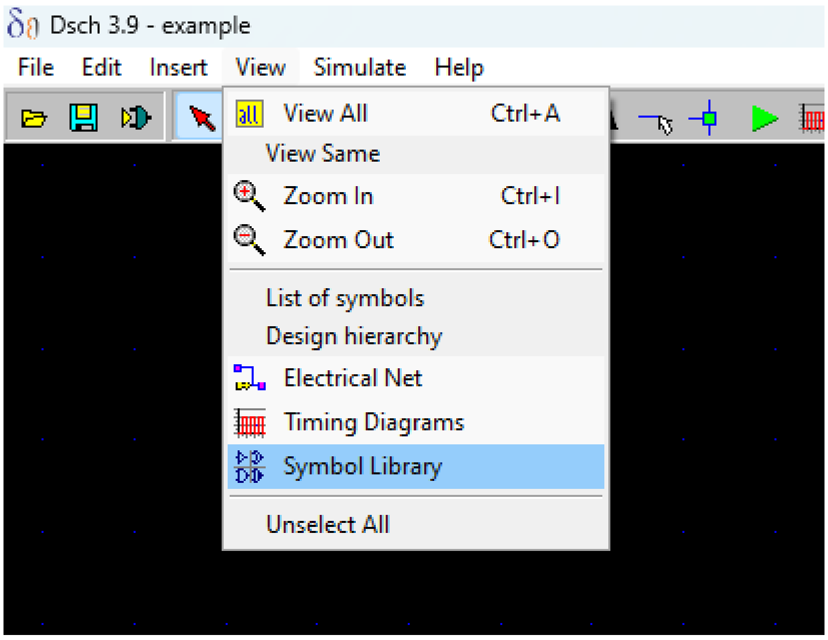
Figures(3):

1) **Tools:**

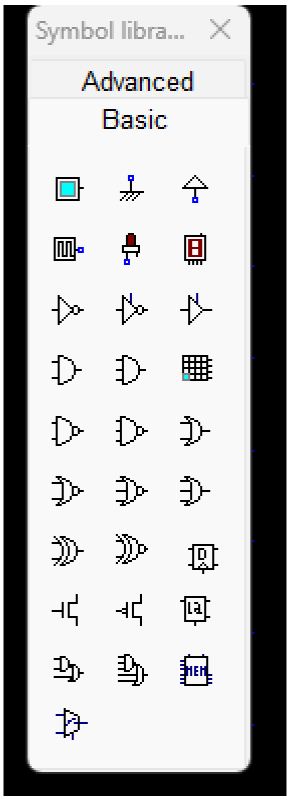
2) **Dsch New project:**



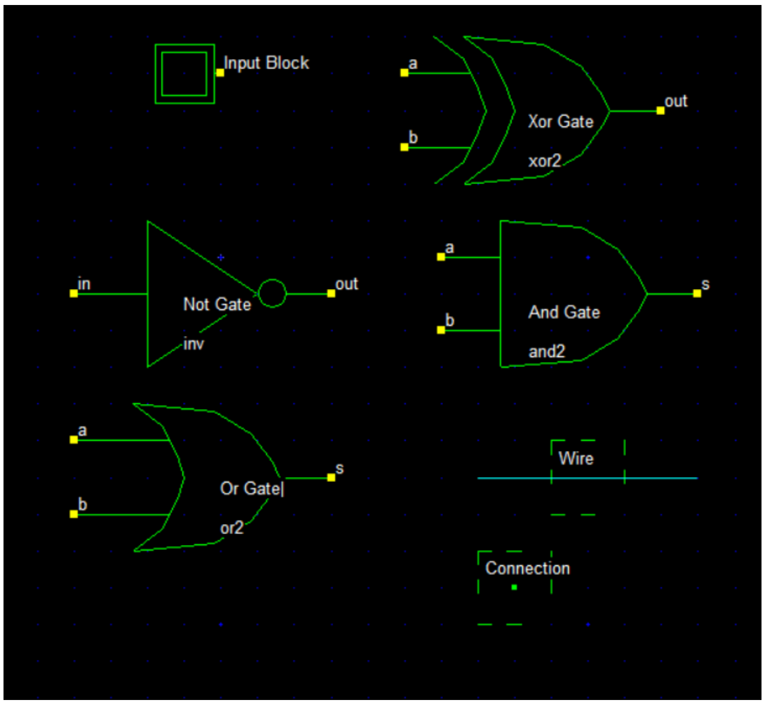
3) Component Choosing using Symbol Library:



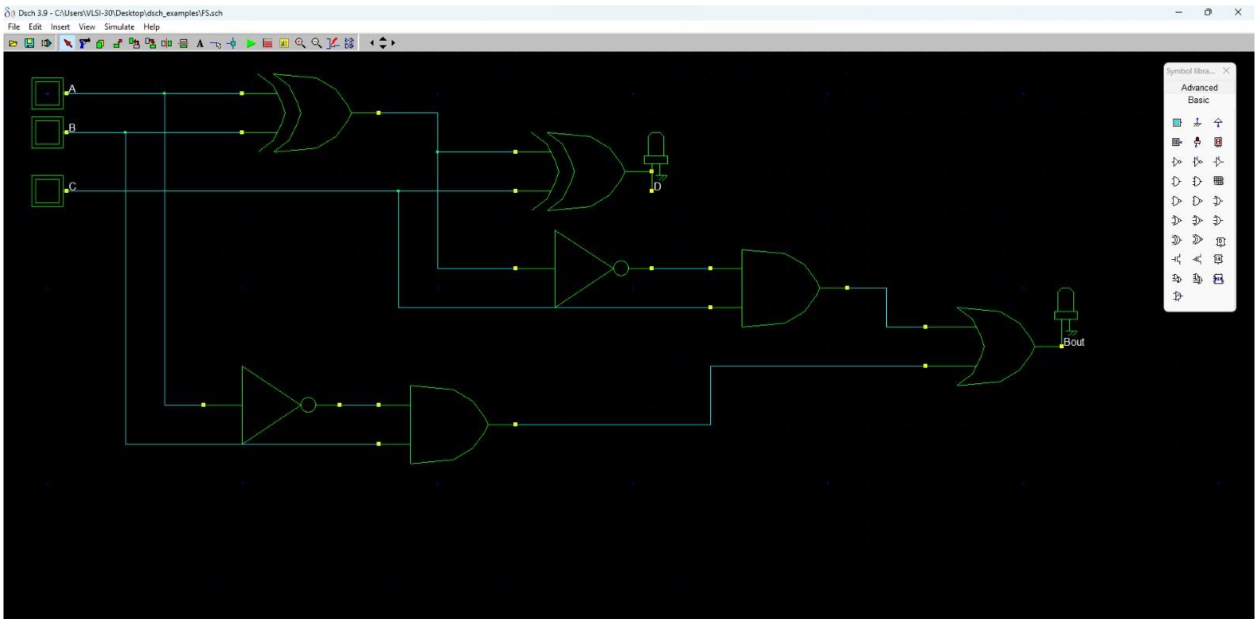
4) Symbol Library:



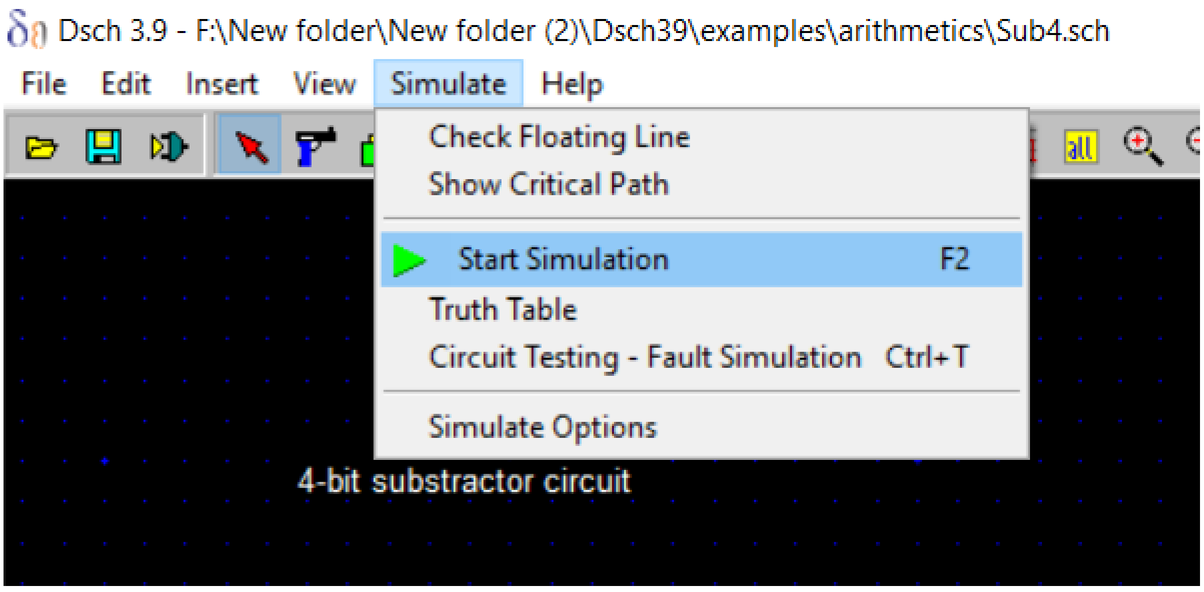
5) Component Used:



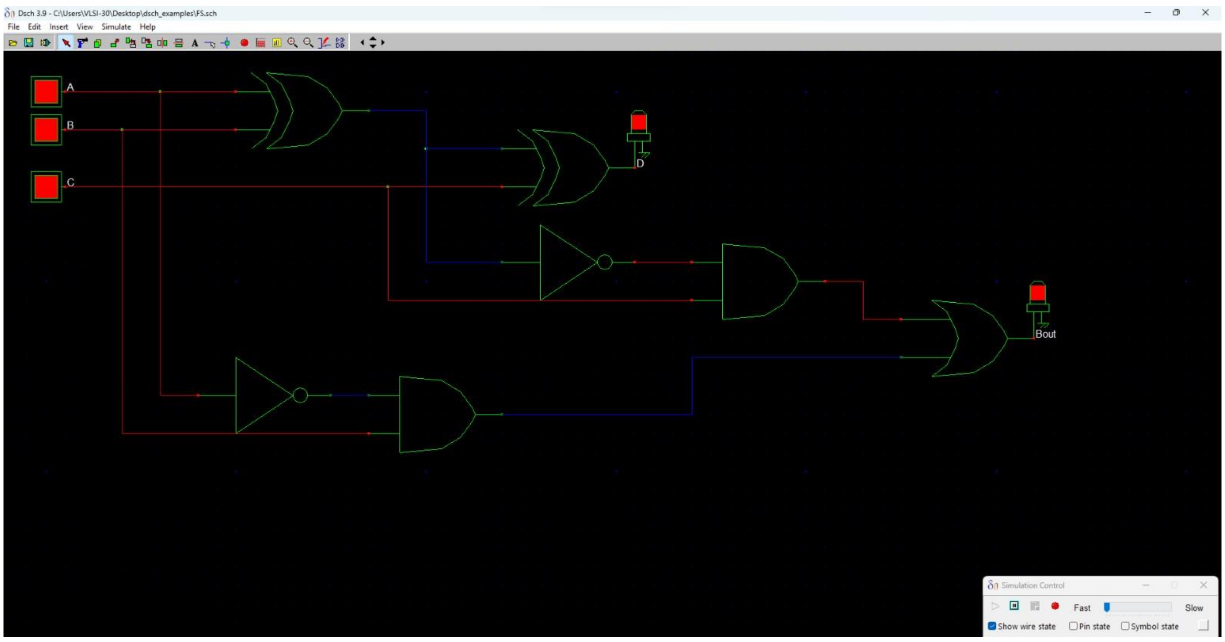
6) Example Circuit Diagram:


Dsch 3.9 - C:\Users\VLSI-30\Desktop\dsch_examples\FS.sch
File Edit Insert View Simulate Help
A
B
C
D
Bout
Symbol libra...
Advanced
Basic

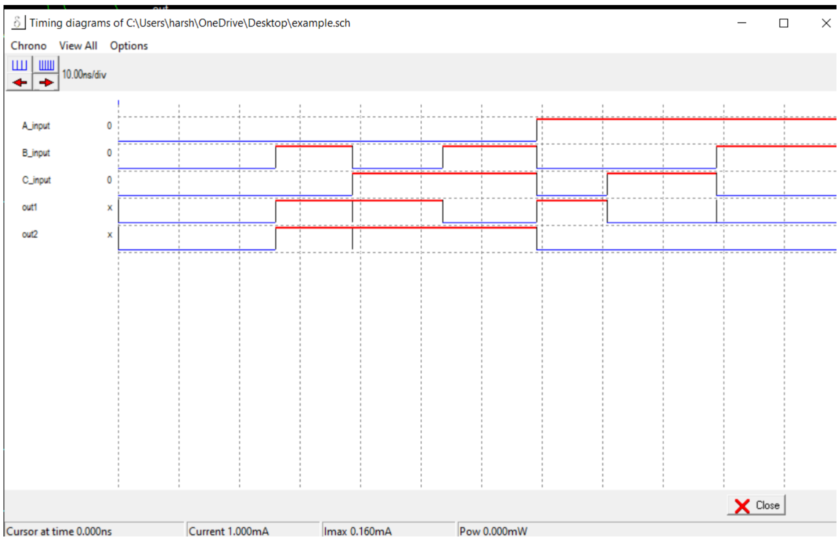
7) Circuit Simulation Menu:


Dsch 3.9 - F:\New folder\New folder (2)\Dsch39\examples\arithmetics\Sub4.sch
File Edit Insert View Simulate Help
Check Floating Line
Show Critical Path
Start Simulation F2
Truth Table
Circuit Testing - Fault Simulation Ctrl+T
Simulate Options
4-bit substractor circuit

8) Simulated Circuit Diagram:


Dsch 3.9 - C:\Users\VLSI-30\Desktop\dsch_examples\FS.sch
File Edit Insert View Simulate Help
A
B
C
D
Bout
Simulation Control
Fast
Slow
Show wire state
Pin state
Symbol state

9) Timing Diagram:



Abstraction of Layout in Microwind Software using DSCH:

Abstracting a DSCH circuit to a Microwind layout using a Verilog file involves several steps. This process includes designing the circuit in DSCH, exporting the design to a Verilog file, and then importing this Verilog file into Microwind for layout generation. Below are the detailed steps:

Step-by-Step Guide:

1. Design the Circuit in DSCH

Open DSCH Software:

- Launch DSCH on your computer.

Create the Circuit:

- Design your full subtractor or any other digital circuit using the schematic editor in DSCH.
- Ensure all connections are correct, and the circuit functions as intended.

Simulate the Circuit:

- Run the simulation to verify the logical behavior of your circuit. Make sure it produces the correct outputs for all input combinations.

2. Export the Circuit to Verilog

Generate Verilog Code:

- Once the circuit design is complete and verified, export the design to a Verilog file.
- In DSCH, go to File > Generate Verilog File (or similar option depending on the version you are using).

Save the Verilog File:

- Save the generated Verilog file to a known location on your computer.

3. Import the Verilog File into Microwind

Open Microwind Software:

- Launch Microwind on your computer.

Create a New Project:

- Start a new project by selecting File > New or File > New Design.

Import the Verilog File:

- Go to File > Import > Verilog (or similar option).
- Browse to the location where you saved the Verilog file from DSCH.
- Select the Verilog file and import it into Microwind.

Generate the Layout:

- Microwind will convert the Verilog code into a physical layout.
- It will generate the layout of your circuit based on the imported Verilog description.
- You can view and edit the layout in the layout editor.

Verify and Simulate the Layout:

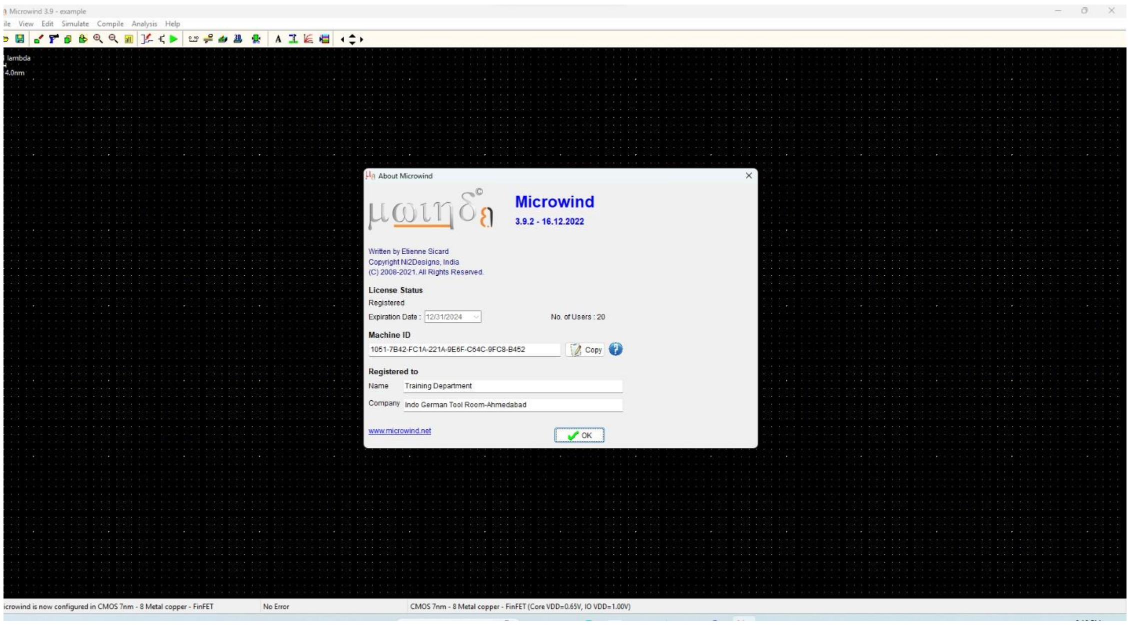
- Perform layout versus schematic (LVS) checks to ensure the layout matches the schematic.
- Run simulations to verify that the layout behaves as expected.

Benefits and Applications

- **Benefits**:
  - **Automation**: Automates the transition from schematic to layout, reducing design time.
  - **Accuracy**: Ensures that the physical implementation matches the logical design.
  - **Efficiency**: Streamlines the design process by integrating schematic capture, simulation, and layout generation.
- **Applications**:
  - **Educational**: Used in academic settings to teach digital and VLSI design concepts.
  - **Prototyping**: Useful for quickly prototyping digital circuits and validating their physical implementation.
  - **Research and Development**: Facilitates the design and testing of new digital circuits and systems.

Figures(4):

1) Open Microwind:


About Microwind

Microwind
3.9.2 - 16.12.2022

Written by Etienne Sicard
Copyright Ni2Designs, India
(C) 2008-2021. All Rights Reserved.

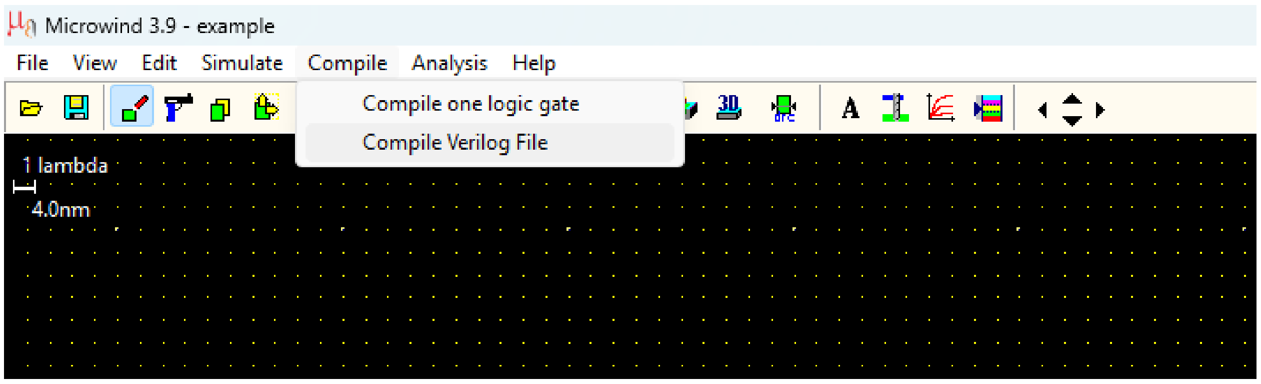
License Status
Registered
Expiration Date : 12/31/2024 No. of Users : 20

Machine ID
1051-7B42-FC1A-221A-9E6F-C64C-9FC8-B452

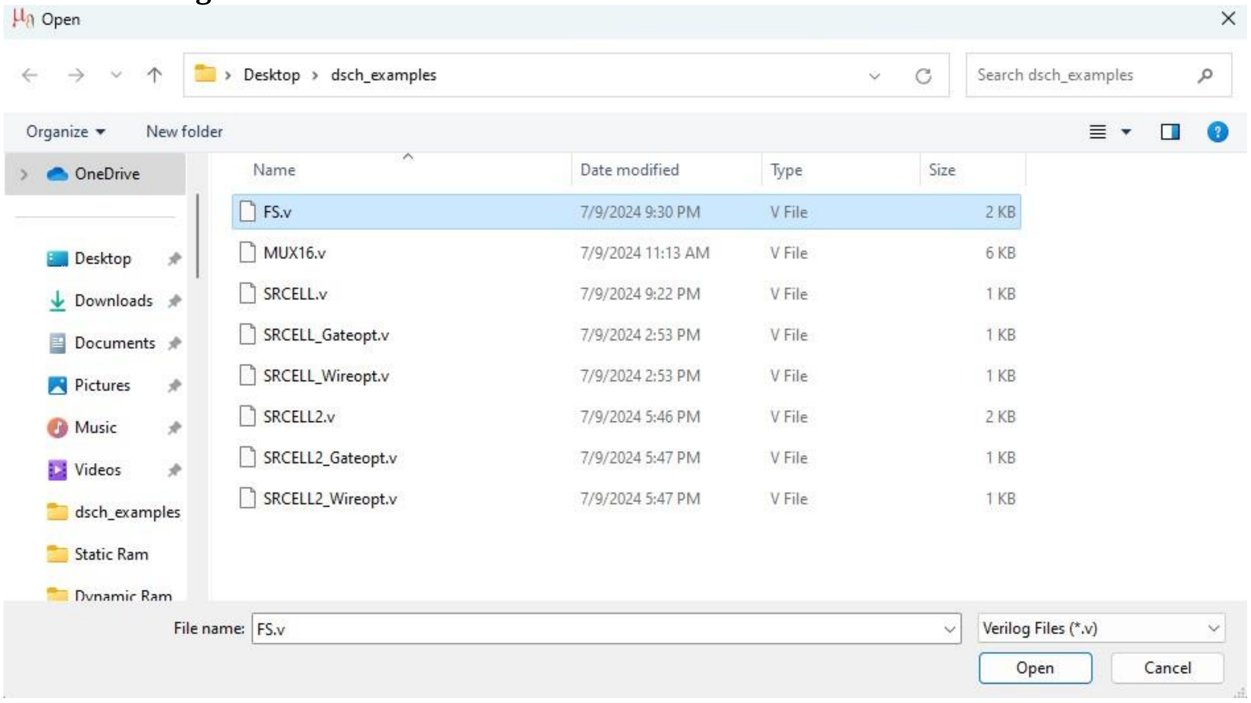
Registered to
Name: Training Department
Company: Indo German Tool Room-Ahmedabad

www.microwind.net

2) Compile Verilog File:



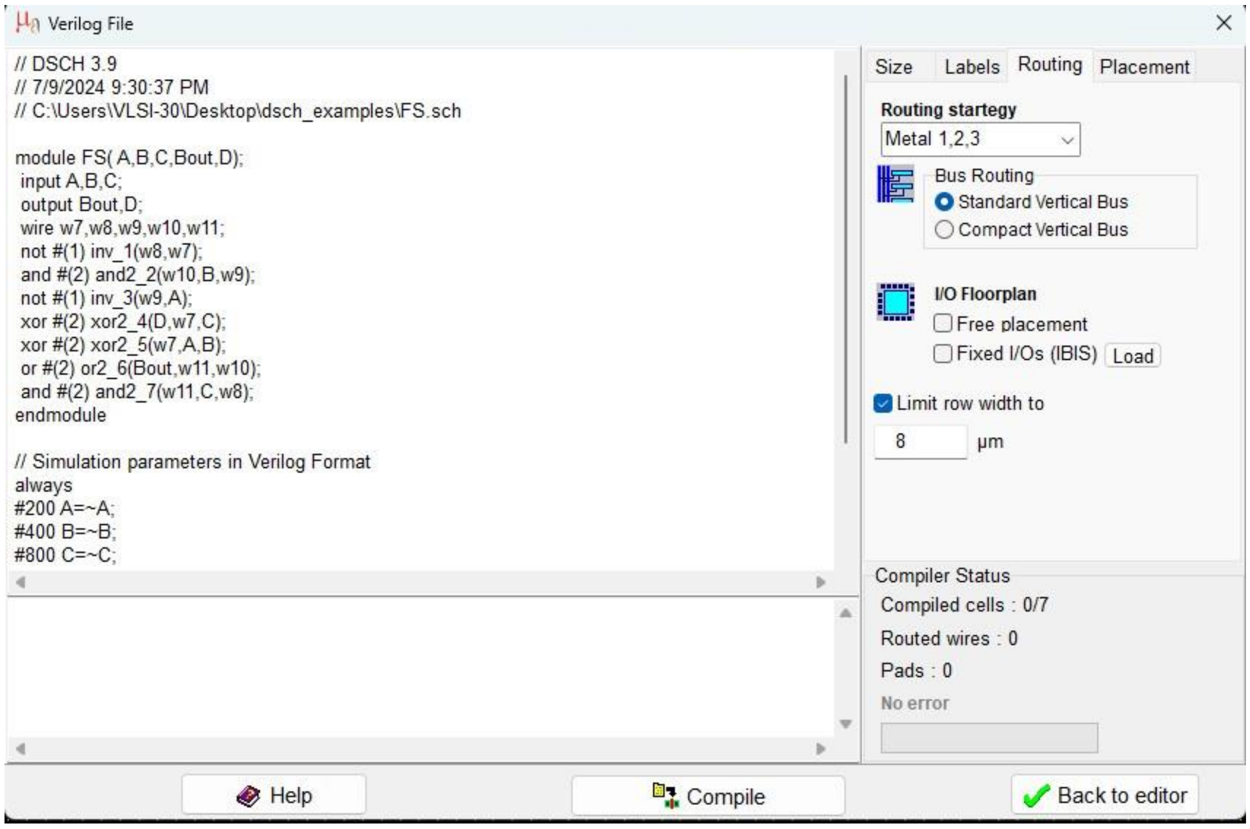
3) Choose Verilog file:



Name	Date modified	Type	Size
FS.v	7/9/2024 9:30 PM	V File	2 KB
MUX16.v	7/9/2024 11:13 AM	V File	6 KB
SRCELL.v	7/9/2024 9:22 PM	V File	1 KB
SRCELL_Gateopt.v	7/9/2024 2:53 PM	V File	1 KB
SRCELL_Wireopt.v	7/9/2024 2:53 PM	V File	1 KB
SRCELL2.v	7/9/2024 5:46 PM	V File	2 KB
SRCELL2_Gateopt.v	7/9/2024 5:47 PM	V File	1 KB
SRCELL2_Wireopt.v	7/9/2024 5:47 PM	V File	1 KB

File name: FS.v
Verilog Files (*.v)

4) Abstract Layout according to your requirements:

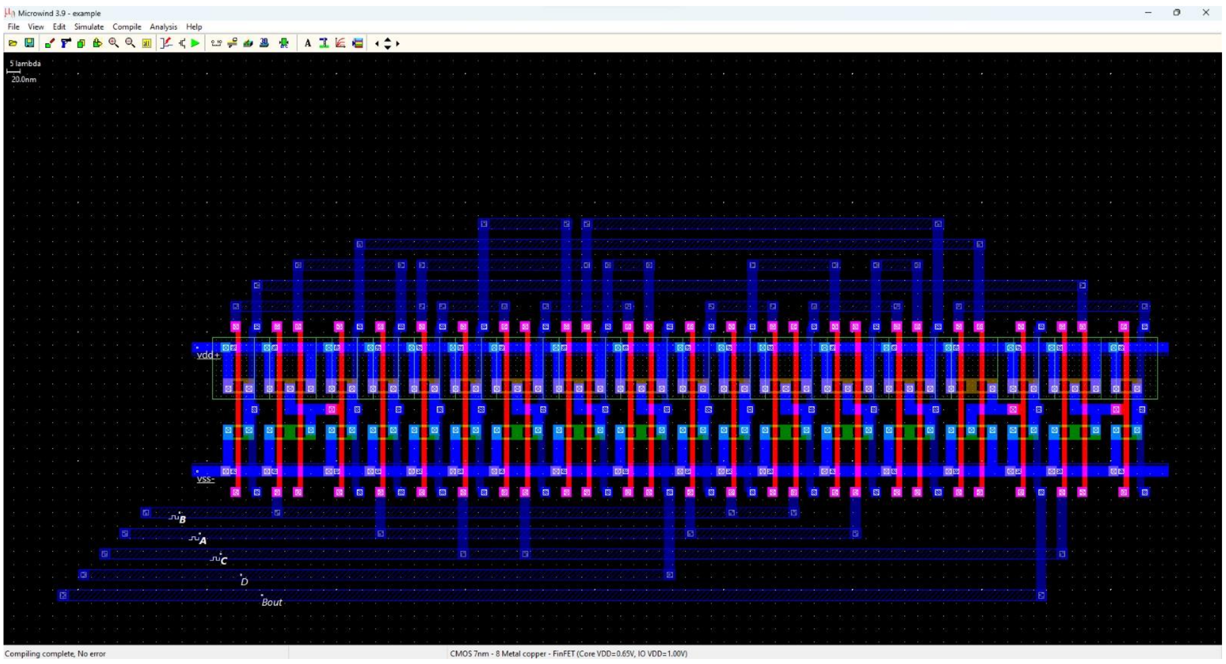


```
// DSCH 3.9
// 7/9/2024 9:30:37 PM
// C:\Users\VLSI-30\Desktop\dsch_examples\FS.sch

module FS( A,B,C,Bout,D);
 input A,B,C;
 output Bout,D;
 wire w7,w8,w9,w10,w11;
 not #(1) inv_1(w8,w7);
 and #(2) and2_2(w10,B,w9);
 not #(1) inv_3(w9,A);
 xor #(2) xor2_4(D,w7,C);
 xor #(2) xor2_5(w7,A,B);
 or #(2) or2_6(Bout,w11,w10);
 and #(2) and2_7(w11,C,w8);
endmodule

// Simulation parameters in Verilog Format
always
#200 A=~A;
#400 B=~B;
#800 C=~C;
```

Routing startegy: Metal 1,2,3
Bus Routing: [✓ Standard Vertical Bus] Compact Vertical Bus
I/O Floorplan: ☐ Free placement ☐ Fixed I/Os (IBIS) Load
☑ Limit row width to 8 µm

Compiler Status
Compiled cells : 0/7
Routed wires : 0
Pads : 0
No error

5) Layout:



Compiling complete, No error

CMOS 7nm - 8 Metal copper - FinFET (Core VDD=0.65V, IO VDD=1.00V)

Conclusion:

This internship project has provided a comprehensive introduction to the fundamentals of digital electronics and VLSI design. Through the use of DSCH software, I have gained hands-on experience in designing, simulating, and verifying digital circuits, including combinational and sequential logic. Additionally, the process of abstracting these designs to physical layouts using Microwind has deepened my understanding of the practical aspects of circuit implementation. The guidance and mentorship received during this internship have been invaluable, equipping me with essential skills and knowledge for a future career in VLSI design and digital electronics. This project has laid a solid foundation for further exploration and specialization in the field, and has significantly enhanced my proficiency in using industry-standard tools for digital circuit design and analysis.

Thank You