JSS MAHAVIDYAPHEETA

JSS ACADEMY OF TECHNICAL EDUCATION

Bengaluru

Department of Computer Science & Engineering

III SEMESTER

Digital Design and Computer Organization (BCS203)

Prepared By:

Dr. Manjunath B T	**Dr. Niranjan C Kundur**	**Mrs. Bhavani B**
Assoc. Professor	**Assoc. Professor**	**Asst. Professor**

H

INSTITUTION VISION

To be among the finest Institutions providing Engineering and Management Education empowered with research, innovation and entrepreneurship.

INSTITUTION MISSION

1. Strive towards Excellence in teaching–learning process and nurture personality development.
2. Encourage Research, Innovation & Entrepreneurship.
3. Train to uphold highest ethical standards in all activities.

DEPARTMENT VISION

To be a distinguished academic and research Department in the field of Computer Science and Engineering for enabling students to be highly competent professionals to meet global challenges.

DEPARTMENT MISSION

1. Impart quality education in Computer Science and Engineering through state-of-the art learning environment and committed faculty with research expertise.
2. Train students to become the most sought-after professionals in the field of Information Technology by providing them strong theoretical foundation with adequate practical training.
3. Provide a conducive environment for faculty and students to carry out research and innovation in collaboration with reputed research institutes and industry.
4. Inculcate human values and professional ethics among students to enable them to become good citizens and serve the society.

PROGRAM EDUCATIONAL OBJECTIVES

1. Graduates shall possess essential skills to adapt to emerging technologies & environment to solve real world problems.
2. Graduates shall have required technical competency for pursuing higher studies & Research.
3. Graduates shall have essential communication and managerial skills to become competent professionals and entrepreneurs.

PROGRAM SPECIFIC OUTCOMES

The Graduates will be able to:

1. Apply the principles of Basic Engineering Science to acquire the hardware and software aspects of Computer Science.
2. Solve the real-world problems using modelling for a specific Computer system and architecture.
3. Ability to design and develop applications using various software and hardware tools.
4. Exhibit the practical competence using broad range of programming languages.

PROGRAM OUTCOMES

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable

development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Course Outcomes (COs):

On successful completion of the course the student shall be able to

CO#	Description	RBT Level
C2O2.1	Apply K–Map techniques to simplify various Boolean expressions.	L3
C2O2.2	Construct different types of combinational and sequential circuits along with Verilog programs.	L3
C2O2.3	Examine the fundamentals of machine instructions, addressing modes and Processor performance.	L4
C2O2.4	Analyze the approaches involved in achieving communication between processor and I/O devices.	L4
C2O2.5	Analyze internal organization of memory and impact of cache/Pipelining on processor performance.	L4

CO-PO Mapping

		Pos											
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Digital Design & Computer Organization [BCS302]	**C202.1**	3	3	3	3	1	-	-	-	-	-	-	1
	C202.2	3	3	3	3	2	-	-	-	1	-	-	1
	C202.3	3	3	3	3	2	-	-	-	-	-	-	1
	C202.4	3	3	3	3	2	-	-	-	1	-	-	1
	C202.5	3	3	3	3	2	-	-	-	1	-	-	1
	All COs	**2**	**2**	**2**	**2**	**1**	**-**	**-**	**-**	**1**	**-**	**-**	**1**

Justification for CO-PO mapping:

CO#	PO#	Correlation	Justification
C202.1	**PO1**	**3**	To understand Karnaugh Map and Boolean Algebra, knowledge of Mathematics, Science, Engineering fundamentals is required
	PO2	**3**	Problem analysis is required to understand and solve problems using Karnaugh Map and Boolean Algebra.
	PO3	**3**	For designing and developing solutions, understanding Karnaugh map and Boolean Algebra is needed.
	PO4	**3**	Complex Problems can be investigated if we have the knowledge of Boolean Algebra and Karnaugh Map.
	PO5	**1**	Modern Simulation Tools can be used to understand Boolean Algebra and Karnaugh map.

	PO12	**1**	Lifelong learning is required to understand Boolean Algebra and Karnaugh map.
C202.2	**PO1**	**3**	Knowledge of Mathematics, Science, Engineering fundamentals and specialization is required to construct Combinational and Sequential Circuits.
	PO2	**3**	Problem Analysis is important to construct Combinational and Sequential Circuits.
	PO3	**3**	Solutions can be designed and developed using Combinational and Sequential Circuits.
	PO4	**3**	Complex Problems can be investigated using Combinational and Sequential Circuits.
	PO5	**2**	Modern Tools can be used to simulate Digital Circuits.
	PO9	**1**	Team work is required for constructing Digital Circuits.
	PO12	**1**	Lifelong learning is required to design and develop Digital Circuits.
C202.3	**PO1**	**3**	Knowledge of Mathematics, Science, Engineering fundamentals and specialization is required to examine the fundamentals of machine instructions, addressing modes and processor performance.
	PO2	**3**	Problem Analysis is important to examine the fundamentals of machine instructions, addressing modes and processor performance.
	PO3	**3**	Solutions can be designed and developed by examining the fundamentals of machine instructions, addressing modes and processor performance.
	PO4	**3**	Complex Problems can be investigated by examining the fundamentals of machine instructions, addressing modes and processor performance.
	PO5	**2**	Modern Tools can be used to examine the fundamentals of machine instructions, addressing modes and processor performance.
	PO12	**1**	Lifelong learning is required to examine the fundamentals of machine instructions, addressing modes and processor performance
C202.4	**PO1**	**3**	Analyzing the approaches involved in achieving communication between processor and I/O devices requires the knowledge of Mathematics, Science, Engineering Fundamentals and Specialization
	PO2	**3**	Problem Analysis is required for examining the approaches involved in achieving communication between processor and I/O devices.
	PO3	**3**	Design and development of solutions require analyzing the approaches involved in achieving communication between processor and I/O devices
	PO4	**3**	To conduct investigations of complex problems, analysis of the approaches involved in achieving communication between processor and I/O devices is required.
	PO5	**2**	Modern tools can be used for analyzing the approaches involved in achieving communication between processor and I/O devices.
	PO9	**1**	Team work is required for analyzing the approaches involved in achieving communication between processor and I/O devices.
	PO12	**1**	Lifelong learning is required for analyzing the approaches involved in achieving communication between processor and I/O devices.
C202.5	**PO1**	**3**	Knowledge of Mathematics, Science, Engineering fundamentals and specialization is required for analyzing

			internal organization of memory and impact of cache/pipelining on processor performance.
	PO2	**3**	Problem Analysis is required for examining internal organization of memory and impact of cache/pipelining on processor performance.
	PO3	**3**	Design and development of solution requires analyzing internal organization of memory and impact of cache/pipelining on processor performance.
	PO4	**3**	Complex Problem Investigation is required for analyzing internal organization of memory and impact of cache/pipelining on processor performance.
	PO5	**2**	Modern tools are required for required for analyzing internal organization of memory and impact of cache/pipelining on processor performance.
	PO9	**1**	Team work is required for analyzing internal organization of memory and impact of cache/pipelining on processor performance.
	PO12	**1**	Lifelong learning is required for analyzing internal organization of memory and impact of cache/pipelining on processor performance.

<u>CO-PSO Mapping</u>:

Digital Design and Computer Organization [BCS302]	**COs**	**PSOs**			
		1	2	3	4
	C202.1	3	3	2	-
	C202.2	3	3	2	-
	C202.3	3	3	2	-
	C202.4	3	3	2	-
	C202.5	3	3	2	-
	All COs	3	3	2	-

<u>CO-PSO Mapping:</u>

CO#	**PO#**	**Correlation**	**Justification**
C202.1	**PSO1**	**3**	Knowledge of Boolean Algebra and Simplification techniques is required to acquire the knowledge of software and hardware aspects of Computer Science.
	PSO2	**3**	Knowledge of Boolean Algebra and Simplification techniques leads to analysis of Real-world applications.
	PSO3	**2**	To design and develop solutions, knowledge of Boolean Algebra and Simplification techniques is required.
C202.2	**PSO1**	**3**	Principles of basic Engineering Science, hardware and software aspects of Computer Science is required to design and develop Digital Circuits.
	PSO2	**3**	Real world applications can be analyzed using Digital Circuits design and development
	PSO3	**2**	Design and development of solutions can be done by designing and developing Digital Circuits.
C202.3	**PSO1**	**3**	Principles of basic Engineering Science, hardware and software aspects of Computer Science is required to analyze fundamentals of machine instructions, addressing modes and processor performance
	PSO2	**3**	Real world applications can be analyzed using fundamentals of machine instructions, addressing modes and processor

			performance
	PSO3	**2**	Design and development of solutions can be done using fundamentals of machine instructions, addressing modes and processor performance
C202.4	**PSO1**	**3**	Knowledge of basic engineering science, hardware and software aspects of Computer Science is required to analyze the approaches involved in achieving communication between processor and I/O devices
	PSO2	**3**	Real world applications can be analyzed using the approaches involved in achieving communication between processor and I/O devices.
	PSO3	**2**	Design and development of Applications can be done by analyzing the approaches involved in achieving communication between processor and I/O devices
C202.5	**PSO1**	**3**	Knowledge of basic Engineering Science, hardware and software aspects of Computer Science is required to analyze internal organization of memory and impact of cache/pipelining on processor performance.
	PSO2	**3**	Real world applications can be analyzed by using internal organization of memory and impact of cache/pipelining on processor performance.
	PSO3	**2**	Design and Development of solutions can be done using internal organization of memory and impact of cache/pipelining on processor performance.

<u>PRACTICAL COMPONENT OF IPCC</u>

Sl. No.	**Experiments** **Simulation packages preferred: Multisim, Modelsim, PSpice or any other relevant**
1	Given a 4-variable logic expression, simplify it using appropriate technique and simulate the sameusing basic gates.
2	Design a 4-bit full adder and subtractor and simulate the same using basic gates.
3	Design Verilog HDL to implement simple circuits using structural, Data flow and Behavioral model.
4	Design Verilog HDL to implement Binary Adder-Subtractor – Half and Full Adder, Half and FullSubtractor.
5	Design Verilog HDL to implement Decimal adder.
6	Design Verilog program to implement Different types of multiplexer like 2:1, 4:1 and 8:1.
7	Design Verilog program to implement types of De-Multiplexer.
8	Design Verilog program for implementing various types of Flip-Flops such as SR, JK and D.
Course outcomes (Course Skill Set): At the end of the course, the student will be able to: CO1: Apply the K–Map techniques to simplify various Boolean expressions. CO2: Design different types of combinational and sequential circuits along with Verilog programs. CO3: Describe the fundamentals of machine instructions, addressing modes and Processor performance.	

CO4: Explain the approaches involved in achieving communication between processor and I/O devices.
CO5: Analyze internal Organization of Memory and Impact of cache/Pipelining on Processor Performance.

Assessment Details (both CIE and SEE)
The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

CIE for the theory component of the IPCC (maximum marks 50)

- IPCC means practical portion integrated with the theory of the course.
- CIE marks for the theory component are **25 marks** and that for the practical component is **25 marks**.
- 25 marks for the theory component are split into **15 marks** for two Internal Assessment Tests (Two Tests, each of 15 Marks with 01-hour duration, are to be conducted) and **10 marks** for other assessment methods mentioned in 22OB4.2. The first test at the end of 40-50% coverage of the syllabus and the second test after covering 85-90% of the syllabus.
- Scaled-down marks of the sum of two tests and other assessment methods will be CIE marks for thetheory component of IPCC (that is for **25 marks)**.
- The student has to secure 40% of 25 marks to qualify in the CIE of the theory component of IPCC.

CIE for the practical component of the IPCC

- **15 marks** for the conduction of the experiment and preparation of laboratory record, and **10 marks** for the test to be conducted after the completion of all the laboratory sessions.
- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to **15 marks**.
- The laboratory test **(duration 02/03 hours)** after completion of all the experiments shall be conducted for 50 marks and scaled down to **10 marks.**
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for **25 marks**.
- The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

SEE for IPCC
Theory SEE will be conducted by University as per the scheduled timetable, with common questionpapers for the course (**duration 03 hours**)

- The question paper will have ten questions. Each question is set for 20 marks.
- There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that module.
- The students have to answer 5 full questions, selecting one full question from each module.
- Marks scored by the student shall be proportionally scaled down to 50 Marks

- **The theory portion of the IPCC shall be for both CIE and SEE, whereas the practical portion will have a CIE component only. Questions mentioned in the SEE paper may include questions from the practical component**.

Suggested Learning Resources:

Books

1. M. Morris Mano & Michael D. Ciletti, Digital Design With an Introduction to Verilog Design, 5e,Pearson Education.

2. Carl Hamacher, ZvonkoVranesic, SafwatZaky, Computer Organization, 5th Edition, Tata McGrawHill.

Web links and Video Lectures (e-Resources):

https://cse11-iiith.vlabs.ac.in/

Activity Based Learning (Suggested Activities in Class)/ Practical Based learning

Assign the group task to Design the various types of counters and display the output accordingly

Assessment Methods

- Lab Assessment (25 Marks)
- GATE Based Aptitude Test

1) **Given a 4-variable logic expression, simplify it using appropriate technique and simulate the sameusing basic gates**

Verilog code for the Boolean Expression E=F(A,B, C,D)=∑m(2,6,8,9,10,11,14)
After simplifying this we get E=AB’ + CD’

```
module simple_circuit(a, b,c,d,e)
input a,b, c,d;
output e;
wire w1, w2, w3, w4;
not n1(w1,d0;
not n2(w2,b);
and a1(w3,c,w1);
and a2(w4,a,w2);
or o1(e, w3,w4);
endmodule
```

2) **Design a 4 bit full adder and subtractor and simulate the same using basic gates**

```
//define a 4-bit full adder
module fulladd4(a, b, c_in,sum, c_out);
 //i/o port declaration
 input [3:0] a, b;
 input c_in;
  output [3:0] sum;
 output c_out;
 //internal net
 wire c1, c2, c3;
 fulladd fa0(a[0], b[0], c_in, sum[0], c1);
 fulladd fa1(a[1], b[1], c1, sum[1], c2);
 fulladd fa2(a[2], b[2], c2, sum[2], c3);
 fulladd fa3(a[3], b[3], c3, sum[3], c_out);
endmodule


//define a 1 bit fulladder
module fulladd(a, b, c_in,sum, c_out);
  input a, b, c_in;
  output sum, c_out;
  wire s1, c1, c2;
  xor (s1,a, b);
  and ( c1,a, b);
  xor (sum,s1, c_in);
  and (c2,s1, c_in);
  or (c_out,c2, c1);
endmodule
```

3) **Design Verilog HDL to implement simple circuits using structural, Data flow and Behavioural model**.

```
/2X1 MUX in gate level modelling

module mux_2x1_gl(
input I0,I1,S,
output Y);
wire sb,a,b;
not(sb,S);
and(a,sb,I0);
and(b,S,I1);
or(Y,a,b);
endmodule

//2X1 MUX in data flow modelling

module mux_2x1_df(
input I0,I1,S,
output Y);
assign Y = (~S & I0)|(S & I1);
endmodule

  //Demux using behavioural Modelling

    module Demultiplexer_1_to_4_case (output reg [3:0] Y, input [1:0] A, input din);
    always @(Y, A) begin
        case (A)
            2'b00 : begin Y[0] = din; Y[3:1] = 0; end
            2'b01 : begin Y[1] = din; Y[0] = 0;   end
            2'b10 : begin Y[2] = din; Y[1:0] = 0; end
            2'b11 : begin Y[3] = din; Y[2:0] = 0; end
        endcase
    end
    endmodule
```

4) **Design Verilog HDL to implement Binary Adder-Subtractor – Half and Full Adder, Half and Full Subtractor**

```
  // Verilog code for Half adder, Full adder, Half subtractor, Full subtractor
  module half_add(a,b,s,c);
  input a,b;
  output s, c;
   xor(s, a, b);
  and(c,a,b);
 endmodule

module Full_add(a,b, c,s,cout);
  input a,b,c;
  output s, cout;
  assign s=a^b^c;
   assign cout=(a&b)| (b&c)|(a&c);
 endmodule

module half_sub(a,b, diff, borrow);
  input a,b,;
  output diff, borrow;
  assign s=a^b;
   assign cout=(~a&b);
 endmodule

module Full_sub(a, b, c, borrow, diff);
  input a,b,c;
  output borrow, diff;
  assign diff=a^b^c;
  assign cout=(~a&b)| (b&c)|(~a&c);
endmodule
```

5) **Design Verilog HDL to implement Decimal adder**.

```
//Verilog code for decimal adder
module bcd_adder(a,b,cin, sum,cout)
input [3:0] a,b;
input cin;
output [3:0] sum;
reg [4:0] sum_temp;
reg [3:0] sum;
reg carry;
always @)a,b,cin)
begin
sum_temp=a+b+cin;
if(sum_temp>0)
begin
sum_temp=sum_temp+6;
cout=1;
sum=sum_temp[3:0];end
else
begincarry=0;
sum=sum_temp[3:0];
end
end
endmodule
```

6) Design Verilog program to implement Different types of multiplexer like 2:1, 4:1 and 8:1.

```
//2X1 MUX using ternary operator

module mux_2x1_dt(
input I0,I1,S,
output Y);
assign Y = S?I1:I0;
endmodule

//2X1 MUX using if statement

module mux_2x1_bif(
input I0,I1,S,
output reg Y);
always @(*) begin
    if(S)
        Y = I1;
    else
        Y = I0;
end
endmodule


//2X1 MUX using case statement
  module mux_2x1_bcase(
  input I0,I1,S,
  output reg Y);
  always @(*) begin
```

```
    case(S)
    1'b0:
        Y = I0;
    1'b1:
        Y = I1;
    endcase
end
endmodule
```

//4X1 MUX using 2x1 MUX and ternary operator

```
module mux_2x1_b(
input I0,I1,I2,I3,S0,S1,
output Y);
assign Y = S1?(S0?I1:I0):(S0?I3:I2);
endmodule
```

//8X1 MUX using 2x1 MUX and ternary operator

```
module mux_2x1_b(
input I0,I1,I2,I3,I4,I5,I6,I7,S0,S1,S2,
output Y);
assign Y = S2?(S1?(S0?I1:I0):(S0?I3:I2)):(S1?(S0?I5:I4):(S0?I7:I6));
endmodule
```

7) Design Verilog program to implement types of De-Multiplexer.

//1-of-4 DEMUX using case statement

```
  module Demultiplexer_1_to_4_case (Y, A, din);
  output reg [3:0] Y;
  input [1:0] A;
  input din;

case (A)
2'b00 : begin
        Y[0] = din; Y[3:1] = 0;
       end
2'b01 : begin
        Y[1] = din; Y[0]   = 0;
       End
```

//Demux using behavioural Modelling

```
  module Demultiplexer_1_to_4_case (output reg [3:0] Y, input [1:0] A, input din);
  always @(Y, A) begin
     case (A)
        2'b00 : begin Y[0] = din; Y[3:1] = 0; end
        2'b01 : begin Y[1] = din; Y[0] = 0;   end
        2'b10 : begin Y[2] = din; Y[1:0] = 0; end
        2'b11 : begin Y[3] = din; Y[2:0] = 0; end
     endcase
  end
  endmodule
```

//Demux using assign keyword

```
module Demultiplexer_1_to_4_assign(output [3:0] Y, input [1:0] A, input din);
assign Y[0] = din & (~A[0]) & (~A[1]);
assign Y[1] = din & (~A[1]) & A[0];
assign Y[2] = din & A[1] & (~A[0]);
assign Y[3] = din & A[1] & A[0];
endmodule
```

8) **Design Verilog program for implementing various types of Flip-Flops such as SR, JK and D.**

```
//Verilog code for flipflops
///T flipflop
module tff(t,clk,rst, q,qb);
input t,clk,rst;
output q,qb;
reg q,qb;
reg temp=0;
always@(posedge clk,posedge rst)
begin
if (rst==0) begin
if(t==1) begin
temp=~ temp;
end
else
temp=temp;
end
q=temp;qb=~temp;
end
end module

//D flipflop
module dff(d,clk,rst,q,qb);
input d,clk,rst;
output q,qb;
reg q,qb;
reg temp=0;
always@(posedge clk,posedge rst)
begin
if (rst==0)
temp=d;
else
temp=temp;
q=temp;
qb=~ temp ;
end
end module

//SR Flipflop
module srff(s,r,clk,rst, q,qb);
input s,r,clk,rst;
output q,qb;
reg q,qb;
reg [1:0]sr;
always@(posedge clk,posedge rst)
begin
sr={s,r};
if(rst==0)
begin
case (sr)
2'd1:q=1'b0;
2'd2:q=1'b1;
2'd3:q=1'b1;
```

```
default: begin end
endcase
end
else
begin
q=1'b0;
end
qb=~q;
end
end module
```

//JK Flipflop

```
module jkff(j,k,clk,rst, q,qb);
input j,k,clk,rst;
output q,qb;
reg q,qb;
reg [1:0]jk;
always@(posedge clk,posedge rst)
begin
jk={j,k};
if(rst==0)
begin
case (jk)
2'd1:q=1'b0;
2'd2:q=1'b1;
2'd3:q=~q;
default: begin end
endcase
end
else
q=1'b0;
qb=~q;
end
end module
```