K S INSTITUTE OF TECHNOLOGY, BANGLORE
Department of Electronics & Communication Engineering

COURSE FILE

NAME OF THE STAFF	: SUNIL KUMAR G R
SUBJECT CODE/NAME	: 18EC56- VERILOG HDL
SEMESTER/YEAR	: V/III
ACADEMIC YEAR	: 2020 – 2021
BRANCH	: ECE (A & B)

Course in-Charge

Module Coordinator

HOD/ECE

K. S. INSTITUTE OF TECHNOLOGY

VISION

"To impart quality technical education with ethical values, employable skills and research to achieve excellence".

MISSION

- To attract and retain highly qualified, experienced & committed faculty.
- To create relevant infrastructure.
- Network with industry & premier institutions to encourage emergence of new ideas by providing research & development facilities to strive for academic excellence.
- To inculcate the professional & ethical values among young students with employable skills & knowledge acquired to transform the society.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

VISION

"To achieve excellence in academics and research in Electronics & Communication Engineering to meet societal need".

MISSION

- **To impart quality technical education with the relevant technologies to produce industry ready engineers with ethical values.**
- **To enrich experiential learning through active involvement in professional clubs & societies.**
- **To promote industry-institute collaborations for research & development.**

K.S. INSTITUTE OF TECHNOLOGY
Department: Electronics and Communication Engg.

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

- Excel in professional career by acquiring domain knowledge.
- Motivation to pursue higher Education & research by adopting technological innovations by continuous learning through professional bodies and clubs.
- To inculcate effective communication skills, team work, ethics and leadership qualities.

PROGRAM SPECIFIC OUTCOMES (PSO'S)

PSO1: Graduate should be able to understand the fundamentals in the field of Electronics & Communication and apply the same to various areas like Signal processing, embedded systems, Communication & Semiconductor technology.

PSO2: Graduate will demonstrate the ability to design, develop solutions for Problems in Electronics & Communication Engineering using hardware and software tools with social concerns.

K S INSTITUTE OF TECHNOLOGY

PROGRAM OUTCOMES (PO'S)

Engineering Graduates will be able to:

PO1 :Engineering knowledge: Apply the knowledge of mathematics, science, engineeringfundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2 : Problem analysis: Identify, formulate, review research literature, and analyze complexengineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 : Design/development of solutions: Design solutions for complex engineering problems anddesign 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.

PO4 : Conduct investigations of complex problems: Use research-based knowledge and researchmethods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 : Modern tool usage: Create, select, and apply appropriate techniques, resources, and modernengineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6 : The engineer and society: Apply reasoning informed by the contextual knowledge to assesssocietal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7 : Environment and sustainability: Understand the impact of the professional engineering solutionsin societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8 : Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms ofthe engineering practice.

PO9 :Individual and team work: Function effectively as an individual, and as a member or leader indiverse teams, and in multidisciplinary settings.

PO10 :Communication: Communicate effectively on complex engineering activities with the engineeringcommunity 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.

PO11 ;Project management and finance: Demonstrate knowledge and understanding of theengineering 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.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage inindependent and life-long learning in the broadest context of technological change.

K. S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-5600109

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CO-PO mapping: Verilog HDL

Course : Verilog HDL			
Course In charge : Sunil Kumar G R			
Type: core		**Course Code:18EC56**	
No of Hours per week			
Theory (Lecture Class)	Practical/Field Work/Allied Activities	Total/Week	Total teaching hours
3	0	3	40
Marks			
Internal Assessment	Examination	Total	Credits
40	60	100	3

Aim/Objective of the Course:

- Learn different Verilog HDL constructs.
- Familiarize the different levels of abstraction in Verilog.
- Understand Verilog Tasks, Functions and Directives.
- Understand timing and delay Simulation.
- Understand the concept of logic synthesis and its impact in verification

Course Learning Outcomes:

After completing the course, the students will be able to,

		Bloom's Level
CO1	**Write Verilog programs in gate, dataflow (RTL), behavioral and switch modeling levels of abstraction.**	K3(APPLYING)
CO2	**Design and verify the functionality of digital circuit/system using test benches.**	K3(APPLYING)
CO3	**Identify the suitable abstraction level for a particular digital design.**	K3(APPLYING)
CO4	**Write the programs more effectively using Verilog tasks, functions and directives.**	K3(APPLYING)
CO5	**Perform timing and delay Simulation and interpret the various constructs in logic synthesis.**	K3(APPLYING)

Syllabus Content:

Module 1: Overview of Digital Design with Verilog HDL Evolution of CAD, emergence of HDLs, typical HDL-flow, why Verilog HDL?, trends in HDLs. **Hierarchical Modeling Concepts** Top-down and bottom-up design methodology, differences between modules and module instances, parts of a simulation, design block, stimulus block. **LO: At the end of this session the student will be able to,** 1. Understand digital design methodologies and hierarchical modelling concepts	CO1, CO2
Module 2: Basic Concepts Lexical conventions, data types, system tasks, compiler directives.	CO3,CO4

Modules and Ports Module definition, port declaration, connecting ports, hierarchical name referencing. **LO: At the end of this session the student will be able to,** 1. Make use of system tasks and compiler directives. 2. Learn various data types available in Verilog and module definition.	
Module 3: Gate-Level Modeling Modeling using basic Verilog gate primitives, description of and/or and buf/not type gates, rise, fall and turn-off delays, min, max, and typical delays. **Dataflow Modeling** Continuous assignments, delay specification, expressions, operators, operands, operator types. **LO: At the end of this session the student will be able to,** 1. write verilog description for the design using gate level and dataflow modeling style 2. Understand modeling gate delays in verilog. Understand timing and delay Simulation	CO1,CO5
Module 4 :Behavioral Modeling Structured procedures, initial and always, blocking and non-blocking statements, delay control, generate statement, event control, conditional statements, Multiway branching, loops, sequential and parallel blocks. **Tasks and Functions:** Differences between tasks and functions, declaration, invocation, automatic tasks and functions. **LO: At the end of this session the student will be able to,** 1. Design digital systems in verilog using behavioral abstraction 2. Understand various constructs of verilog language.	CO3,CO4
Module 5: Useful Modeling Techniques: Procedural continuous assignments, overriding parameters, conditional compilation and execution, useful system tasks. **Logic Synthesis with Verilog:** Logic Synthesis, Impact of logic synthesis, Verilog HDL Synthesis, Synthesis design flow, Verification of Gate-Level Netlist. (Chapter 14 till 14.5 of Text). **LO: At the end of this session the student will be able to,** 1. Use better modeling techniques 2. Interpret the various constructs in logic synthesis.	CO4,CO5
Text Books: Samir Palnitkar, **"Verilog HDL: A Guide to Digital Design and Synthesis"**, Pearson Education, Second Edition.	
Reference Books: 1. Operating Donald E. Thomas, Philip R. Moorby, —The Verilog Hardware Description Languagel, Springer Science+Business Media, LLC, Fifth edition. 2. Michael D. Ciletti, —Advanced Digital Design with the Verilog HDLI Pearson (Prentice Hall), 2nd Ed. . 3. Padmanabhan, Tripura Sundari, —Design through Verilog HDLI, Wiley, 2016 or earlier.	
Useful Journals: https://ieeexplore.ieee.org/document/1708385/ **Useful Websites**: www.asic-world.com/verilog/veritut.html https://ieeexplore.ieee.org/iel5/10779/33945/01620780.pdf http://www.iuma.ulpgc.es/~nunez/clases-FdC/verilog/Verilog-IEEE-1364.pdf	
Teaching and Learning Methods: 1. Lecture class: 40 hrs. 2. Self-study: 5hrs. 3. Field visits/Group Discussions/Seminars: (0+0+0)hrs. Practical classes: 3 hrs./week - Laboratory	
Assessment: Type of test/examination: Written examination	

Continuous Internal Evaluation(CIE) : 40 marks (Average of best two of total three tests will be considered)

Semester End Exam(SEE) : 60 marks (students have to answer all main questions)
Test duration: 1 :30 hr
Examination duration: 3 hrs

CO - PO MAPPING

PO1: Science and engineering Knowledge **PO2:** Problem Analysis **PO3:** Design & Development **PO4:**Investigations of Complex Problems **PO5:** Modern Tool Usage **PO6:** Engineer & Society	**PO7:**Environment and Society **PO8:**Ethics **PO9:**Individual & Team Work **PO10:** Communication **PO11:**Project Mngmt & Finance **PO12:**Life long Learning

PSO1: Graduate should be able to understand the fundamentals in the field of Electronics & Communication and apply the same to various areas like Signal processing, embedded systems, Communication & Semiconductor technology.

PSO2: Graduate will demonstrate the ability to design, develop solutions for Problems in Electronics & Communication Engineering using hardware and software tools with social concerns.

CO 17EC53	Bloom's Level	PO1	PO2	PO3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O1	PS O2
CO1	K3	2	-	-	-	-	-	-	-	-	-	-	-	2	-
CO2	K3	3	3	3	-	-	-	-	-	-	-	-	1	2	3
CO3	K3	3	3	1	-	-	-	-	-	-	-	-	1	2	3
CO4	K3	3	3	1	-	-	-	-	-	-	-	-	1	2	3
CO5	K3	3	3	1	-	-	-	-	-	-	-	-	1	2	-
17EC53		2.8	3	1.5									1	2	3

Justification for CO-PO mapping

CO -Subject Code	Justification for PO mapping
CO1	PO1(2):students will be able to know the basics of Verilog HDL programming language
CO2	PO1(3):students will be able to identify the problem PO2(3):the identified problem can be solved by using solutions of Verilog HDL PO3(3):students will be able to design a Verilog code to solve simple problems

	PO12(1):problem solving techniques can be used life long
CO3	PO1(3):students will be able to identify the problem and define at different abstract levels PO2(3):the identified problem can be solved by using different abstraction levels of Verilog PO3(1):students will be able to design a Verilog code at different abstraction levels PO12(1):problem solving techniques using different abstraction levels can be used life long
CO4	PO1(3):students will be able to identify the problem and define at different abstract levels with timing simulation PO2(3):the identified problem can be solved by using different abstraction levels of Verilog with timing simulation PO3(1):students will be able to design a Verilog code at different abstraction levels with timing simulation PO12(1):problem solving techniques using different abstraction levels and with timing simulation can be used life long
CO5	PO1(3):students will be able to identify different modeling techniques PO2(3):the problems can be solved by better modeling techniques PO3 (1): students will be able to interpret the various constructs in logic synthesis. PO12(1):problem solving techniques using different modeling techniques can be used life long

CO PO mapping for the events conducted after gap identification

Sl. No.	Gap Identification	CO	Relevant PO Mapping
1.	Planning mini projects	CO1, CO3	PO8, PO9, PO10
2.	Quiz using ICT tools	CO1, CO2	PO9
3.	Trying to write HDL code for complex digital circuits	CO1, CO3	PO2, PO8, PO9, PO10, O11

Signature of Course in-Charge **Signature of Module Coordinator** **Signature of HOD**

K. S INSTITUTE OF TECHNOLOGY, BENGALURU-560109

TENTATIVE CALENDAR OF EVENTS: ODD SEMESTER (2020-2021)

SESSION: SEP 2020 - JAN 2021

Week No.	Month	Day						Days	Activities
		Mon	Tue	Wed	Thu	Fri	Sat		
1	SEP		1*	2	3	4	5	5	1*-Commencement of Higher Semester
2	SEP	7	8	9	10	11	12	6	
3	SEP	14	15	16	17 H	18	19	5	17- Mahalaya Amavasya
4	SEP	21	22	23	24	25	26TA	6	
5	SEP / OCT	28 T1	29 T1	30 T1	1	2 H	3	5	2- Mahatma Gandhi Jayanthi
6	OCT	5	6BV	7ASD	8	9	10	6	5-10 First Feed Back
7	OCT	12	13	14	15	16	17 DH	5	
8	OCT	19	20	21	22	23	24	6	24 - Monday Time Table
9	OCT	24 H	27	28	29	30 H	31 H	3	26- Vijayadashami 30- Eid-Milad 31- Maharishi Valmiki Jayanti
10	NOV	2	3	4	5	6	7 TA	6	7 - Wednesday Time Table
11	NOV	9	10	11	12	13	14 DH	5	
12	NOV	16 H	17 T2	18 T2	19 T2	20	21	5	16 - Balipadyami Deepavalli 18 - 21 Second Feed Back 21 - Friday Time Table
13	NOV	23	24BV	25ASD	26	27	28 DH	5	
14	NOV /DEC	30	1	2	H	4	5	5	3- Kanakadasa Jayanti 5 - Monday Time Table
15	DEC	7	8	9	10	11	12 DH	5	
16	DEC	14	15	16	17	18	19	6	19- Monday Time Table
17	DEC	21	22	23	24	25 H	26 DH	4	25-Christmas
18	DEC/ JAN	28	29	30	31	1 TA	2	6	2Thursday Time Table
19	JAN	4 LT	5 LT	6 LT	7 LT	8	9 DH	5	
20	JAN	11 T3	12 T3	13 T3	H	15	16 *	5	14- Makara sankaranthi 16* -Last Working Day
Total No of Working Days : 106									

Total Number of working days (Excluding holidays and Tests)=87

H	Holiday
BV	Blue Book Verification
T1,T2, T3	Tests 1,2, 3
ASD	Attendance & Sessional Display
DH	Declared Holiday
LT	Lab Test
TA	Test attendance

Monday	16
Tuesday	16
Wednesday	16
Thursday	16
Friday	17
Saturday	6
Total	87

PRINCIPAL
K.S. INSTITUTE OF TECHNOLOGY
BENGALURU - 560 109.

K.S INSTITUT E OF TECHNOLOGY, Bengaluru-109

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

TENTATIVE CALENDAR OF EVENTS: ODD SEMESTER (2020-2021)

SESSION: SEP 2020 – JAN 2021

Week No.	Month	Day						Days	Activities	Department Activities Tentative Dates
		Mon	Tue	Wed	Thu	Fri	Sat			
1	SEP		1*	2	3	4	5	5	1*-Commencement of Higher Semester	
2	SEP	7	8	9	10	11	12	6		10 - Technical Talk IEEE, ASH
3	SEP	14	15	16		18	19	5	17- Mahalaya Amavasya	15- Engineers Day IEEE 19 - Guest Lecture
4	SEP	21	22	23	24	25	26 TA	6		21 - Technical Talk 23 - Valedictory IEEE
5	SEP / OCT	28 T1	29 T1	30 T1	1		3	5	2- Mahatma Gandhi Jayanthi	
6	OCT	5	6 BV	7 ASD	8	9	10	6	5-10 First Feed Back	5 - IEEE Day 10 - Guest Lecture
7	OCT	12	13	14	15	16		5		15 - Technical Talk
8	OCT	19	20	21	22	23	24	6	24 Monday Time table	19 - Technical Talk 20 - Technical Talk
9	OCT		27	28	29			3	26- Vijayadashami 30- Eid-Milad 31- Maharishi Valmiki	
10	NOV	2	3	4	5	6	7TA	6	7-Wednesday Time Table	
11	NOV	9	10	11	12	13		5		12 - Technical Talk
12	NOV		17 T2	18 T2	19 T2	20	21	5	16 - Balipadyami Deepavalli 18-21 Second Feed Back 21- Friday Time Table	20 - Simulation
13	NOV	23	24 BV	25 ASD	26	27		5		
14	NOV /DEC	30	1	2		4	5	5	3- Kanakadasa Jayanti 5- Monday Time Table	
15	DEC	7	8	9	10	11		5		
16	DEC	14	15	16	17	18	19	6	19- Monday Time Table	
17	DEC	21	22	23	24			4	25- Chrisumas	
18	DEC/JAN	28	29	30	31	1TA	2	6	2- Tuesday Time Table	
19	JAN	4LT	5LT	6LT	7LT	8		5		
20	JAN	11T3	12T3	13T3		15	16	5	14- Makara Sankranthi 16* -Last Working Day	
Total No of Working Days : 106										

Total Number of working days (Excluding holidays and Tests)=87

H	Holiday
BV	Blue Book Verification
T1, T2, T3	Tests 1,2, 3
ASD	Attendance & Sessional Display
DH	Declared Holiday
LT	Lab Test
TA	Test attendance

Monday	16
Tuesday	16
Wednesday	16
Thursday	16
Friday	17
Saturday	6
Total	**87**

30/11/2020

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE -109

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

V SEMESTER TIME TABLE FOR THE YEAR 2020 (ODD SEMESTER)

W.E.F. : 30/11/2020 — **OFFLINE TIME TABLE** — CLASS TEACHER : Mrs. SANGEETHA V

SEC : 'A'

PERIOD	1	2	11.00 AM 11.15 AM	3	4	1.15 PM 1.45 PM	5	6
TIME / DAY	9.00 AM 10.00 AM	10.00 AM 11.00 AM		11.15 AM 12.15 PM	12.15 PM 1.15 PM		1.45 PM 2.45 PM	2.45 PM 3.45 PM
MON	DSP(18EC52)	M&E(18ES51)		← PLACEMENT →			M&E(18ES51)	ITC(18EC54)
TUE	← HDL LAB (18ECL58)/ A-1 DSP LAB (18ECL57) A-2 →			← HDL LAB (18ECL58)/ A-3 DSP LAB (18ECL57) A-1 →			← HDL LAB (18ECL58)/ A-2 DSP LAB (18ECL57) A-3 →	
WED	EMW(18EC55)	ITC(18EC54)		DSP(18EC52)	VERILOG HDL (18EC56)		ITC(18EC54)	PCS(18EC53)
THU	M&E(18ES51)	ITC(18EC54)		EMW(18EC55)	VERILOG HDL (18EC56)		PCS(18EC53)	DSP(18EC52)
FRI	EMW(18EC55)	M&E(18ES51)		PCS(18EC53)	VERILOG HDL (18EC56		EVS(18CIV59)	EMW(18EC55)
SAT	VERILOG HDL (18EC56)	PCS(18EC53)		DSP(18EC52)	TUTORIAL			

BREAK — LUNCH BREAK

Sub-Code	Subject Name	Faculty Name	Login ID	Password
18ES51	Technological Innovation Management And Entrepreneurship	Dr. Sangappa.S.B	Team Link ID : 1919474122	
18EC52	Digital Signal Processing	Mrs.Sangeetha.V	Zoom ID : 703-865-7441	2020
18EC53	Principles of Communication Systems	Mrs. Pooja S	Zoom ID : 77470838311	3vJW1F
18EC54	Information Theory & Coding	Mrs. Aruna Rao B.P	Zoom ID : 6891013731	itc2920
18EC55	Electromagnetic Waves	Mrs. Jayasudha B.S.K	Zoom ID : 5414395615	2020
18EC56	Verilog HDL	Mr. Sunil Kumar.G.R	Zoom ID : 83893954485	8j2x6i
18ECL57	Digital Signal Processing Laboratory	Mrs. Jayasudha B.S.K, Mrs. Sangeetha V	Zoom ID : 703-865-7441	2020
18ECL58	HDL Laboratory	Dr. Surekha B, Mr. Sunil Kumar G R	Zoom ID : 89702477459	2CkhJz
18CIV59	Environmental Studies	Mrs. Shylaja K R		

Time Table Co-ordinator — **HOD** — **Principal**

HEAD OF THE DEPARTMENT
Dept. of Electronics & Communication Engg.
K.S. Institute of Technology
Bangalore-560 109

K.S. INSTITUTE OF TECHNOLOGY
BENGALURU-560 109

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE -109

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

V SEMESTER TIME TABLE FOR THE YEAR 2020 (ODD SEMESTER)

W.E.F. : 30/11/2020 — **OFFLINE TIME TABLE** — CLASS TEACHER : Mrs.ARUNA RAO

SEC : 'B'

PERIOD	1	2	11.00 AM 11.15 AM	3	4	1.15 PM 1.45 PM	5	6
TIME / DAY	9.00 AM 10.00 AM	10.00 AM 11.00 AM		11.15 AM 12.15 PM	12.15 PM 1.15 PM		1.45 PM 2.45 PM	2.45 PM 3.45 PM
MON	PCS(18EC53)	M&E(18ES51)	BREAK	← PLACEMENT →		LUNCH BREAK	M&E(18ES51)	VERILOG HDL (18EC56)
TUE	DSP(18EC52)	ITC(18EC54)		PCS(18EC53)	VERILOG HDL (18EC56)		EMW(18EC55)	PCS(18EC53)
WED	← HDL LAB (18ECL58)/ B-1 DSP LAB (18ECL57) B-2 →			← HDL LAB (18ECL58)/ B-1 DSP LAB (18ECL57) B-2 →			← HDL LAB (18ECL58)/ B-1 DSP LAB (18ECL57) B-2 →	
THU	M&E(18ES51)	VERILOG HDL (18EC56)		PCS(18EC53)	DSP(18EC52)		ITC(18EC54)	VERILOG HDL (18EC56)
FRI	ITC(18EC54)	M&E(18ES51)		DSP(18EC52)	EMW(18EC55)		EVS(18CIV59)	ITC(18EC54)
SAT	EMW(18EC55)	DSP(18EC52)		EMW(18EC55)	TUTORIAL			

Sub-Code	Subject Name	Faculty Name	Login ID	Password
18ES51	Technological Innovation Management And Entrepreneurship	Dr. Sangappa.S.B	Team Link ID : 1919474122	
18EC52	Digital Signal Processing	Mrs.Sangeetha.V	Zoom ID : 703-865-7441	2020
18EC53	Principles of Communication Systems	Mrs. Pooja S	Zoom ID : 77470838311	3vJW1F
18EC54	Information Theory & Coding	Mrs. Aruna Rao B.P	Zoom ID : 6891013731	itc2920
18EC55	Electromagnetic Waves	Mrs. Jayasudha B.S.K	Zoom ID : 5414395615	2020
18EC56	Verilog HDL	Mr. Sunil Kumar.G.R	Zoom ID : 81654633243	3XGArv
18ECL57	Digital Signal Processing Laboratory	Mrs. Jayasudha B.S.K, Mrs. Sangeetha V	Zoom ID : 5414395615	2020
18ECL58	HDL Laboratory	Dr. Surekha B, Mr. Sunil Kumar G R	Zoom ID : 89702477459	2CkhJz
18CIV59	Environmental Studies	Mrs. Shylaja K R		

Time Table Co-ordinator **HOD** **Principal**

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE -109

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

INDIVIDUAL TIME TABLE FOR THE YEAR - 2020 (ODD SEMESTER)

W.E.F. : 1/9/2020

NAME OF THE FACULTY : SUNIL KUMAR G.R.

DESIGNATION: ASSISTANT PROFESSOR

PERIOD	1	2	11.00 AM 11.15 AM	3	4	1.15 PM 1.45 PM	5	6
TIME / DAY	9.00 AM 10.00 AM	10.00 AM 11.00 AM		11.15 AM 12.15 PM	12.15 PM 1.15 PM		1.45 PM 2.45 PM	2.45 PM 3.45 PM
MON			TEA BREAK			LUNCH BREAK	← HDL LAB - B (18ECL58) →	
TUE	Verilog HDL-A (18EC56)				Verilog HDL-B (18EC56)		← HDL LAB - A (18ECL58) →	
WED								
THU		Verilog HDL-B (18EC56)		Verilog HDL-A (18EC56)				
FRI		Verilog HDL-A (18EC56)			Verilog HDL-B (18EC56)			PEDAGOGY Verilog HDL-A & B (18EC56)
SAT	← PROJECT WORK PHASE - I (17ECP78) →							

	Subject Code	Subject Name	Sem	Section	Work Load
Subject 1	18EC56	Verilog HDL	V	A&B	8
Lab -1	18ECL58	HDL Laboratory	V	A&B	9
Project	17ECP78	Project Work Phase–I + Project work Seminar	VII		2
ADDITIONAL WORK: MENTORING AND OTHERS					
TOTAL LOAD = 19 Hrs/Week					

Time Table Co-ordinator

HOD

Principal

B. E. (EC / TC)
Choice Based Credit System (CBCS) and Outcome Based Education (OBE)
SEMESTER – V

Verilog HDL

Course Code	**18EC56**	**IA Marks**	**40**
Number of Lecture Hours/Week	**03**	**Exam Marks**	**60**
Total Number of Lecture Hours	**40 (08 Hours per Module)**	**Exam Hours**	**03**

CREDITS– 03

Course Learning Objectives:

- Learn different Verilog HDL constructs.
- Familiarize the different levels of abstraction in Verilog.
- Understand Verilog Tasks, Functions and Directives.
- Understand timing and delay Simulation.
- Understand the concept of logic synthesis and its impact in verification

Module 1	**RBT Level**
Overview of Digital Design with Verilog HDL: Evolution of CAD, emergence of HDLs, typical HDL-flow, why Verilog HDL?, trends in HDLs. **Hierarchical Modeling Concepts:** Top-down and bottom-up design methodology, differences between modules and module instances, parts of a simulation, design block, stimulus block.	L1,L2,L3
Module 2	
Basic Concepts: Lexical conventions, data types, system tasks, compiler directives. **Modules and Ports:** Module definition, port declaration, connecting ports, hierarchical name referencing.	L1,L2,L3
Module 3	
Gate-Level Modeling: Modeling using basic Verilog gate primitives, description of and/or and buf/not type gates, rise, fall and turn-off delays, min, max, and typical delays. **Dataflow Modeling:** Continuous assignments, delay specification, expressions, operators, operands, operator types.	L1,L2,L3
Module 4	
Behavioral Modeling: Structured procedures, initial and always, blocking and non-blocking statements, delay control, generate statement, event control, conditional statements, Multiway branching, loops, sequential and parallel blocks. **Tasks and Functions:** Differences between tasks and functions, declaration, invocation, automatic tasks and functions.	L1,L2,L3
Module 5	
Useful Modeling Techniques: Procedural continuous assignments, overriding parameters, conditional compilation and execution, useful system tasks. **Logic Synthesis with Verilog:** Logic Synthesis, Impact of logic synthesis, Verilog HDL Synthesis, Synthesis design flow, Verification of Gate-Level Netlist. **(Chapter 14 till 14.5 of Text).**	L1,L2,L3

Course Outcomes: At the end of this course, students should be able to

- Write Verilog programs in gate, dataflow (RTL), behavioral and switch modeling levels of Abstraction.
- Design and verify the functionality of digital circuit/system using test benches.
- Identify the suitable Abstraction level for a particular digital design.
- Write the programs more effectively using Verilog tasks, functions and directives.
- Perform timing and delay Simulation
- Interpret the various constructs in logic synthesis.

Question paper pattern:

- Examination will be conducted for 100 marks with question paper containing 10 full questions, each of 20 marks.
- Each full question can have a maximum of 4 sub questions.
- There will be 2 full questions from each module covering all the topics of the module.
- Students will have to answer 5 full questions, selecting one full question from each module.

- The total marks will be proportionally reduced to 60 marks as SEE marks is 60.

Text Book:

Samir Palnitkar, **"Verilog HDL: A Guide to Digital Design and Synthesis"**, Pearson Education, Second Edition.

Reference Books:

1. Donald E. Thomas, Philip R. Moorby, "The Verilog Hardware Description Language", Springer Science+Business Media, LLC, Fifth edition.
2. Michael D. Ciletti, "Advanced Digital Design with the Verilog HDL" Pearson (Prentice Hall), Second edition.
3. Padmanabhan, Tripura Sundari, "Design through Verilog HDL", Wiley, 2016 or earlier.

KS INSTITUTE OF TECHNOLOGY BANGALORE
DEPARTMENT OF ELECRONICS & COMMUNICATION ENGINEERING

NAME OF THE STAFF : **Sunil Kumar G R**
SUBJECT CODE/NAME : **18EC56/VERILOG HDL**
SEMESTER/YEAR : **V / III (A sec)**
ACADEMIC YEAR : **2020-2021**

Sl. No.	Topic to be covered	Mode of Delivery	Teaching Aid	No. of Periods	Cumulative No. of Periods	Proposed Date
MODULE 1:Overview of Digital Design with Verilog HDL &Hierarchical Modeling Concepts						
1.	Evolution of CAD, emergence of HDLs	L+D	PPT	1	1	1/9/2020
2.	Typical HDL-flow	L+D	PPT	1	2	3/9/2020
3.	why Verilog HDL?, Trends in HDLs	L+D	PPT	1	3	4/9/2020
4.	Top-down and bottom-up design methodology	L+D	PPT	2	5	5/9/2020 8/9/2020
5.	Differences between modules and module instances	L+D	PPT	1	6	10/9/2020
6.	Parts of a simulation, Design block	L+D	PPT	1	7	11/9/2020
7.	Stimulus block., Examples	L+D	PPT	1	8	12/9/2020
MODULE 2:Basic Concepts, Modules and Ports						
8.	Lexical conventions	L+ D	PPT	1	9	15/9/2020
9.	Data types	L+D	PPT	1	10	18/9/2020
10.	Data types	L+D	PPT	1	11	19/9/2020
11.	System tasks	L+D	PPT	1	12	22/9/2020
12.	Compiler directives	L+D	PPT	1	13	24/9/2020
13.	Compiler directives, examples	L+D	PPT	1	14	25/9/2020
14.	Module definition	L+D	PPT	1	15	26/9/2020

15.	Port declaration	L+D	PPT	1	16	29/9/2020
16.	Connecting ports	L+D	PPT	1	17	1/10/2020
17.	Hierarchical name referencing	L+D	PPT	1	18	3/10/2020
MODULE 3: Gate-Level Modeling & Dataflow Modeling						
18.	Modeling using basic Verilog gate primitives	L+D	PPT	1	19	8/10/2020
19.	Description of and/or and buf/not type Gates	L+D	PPT	1	20	9/10/2020
20.	Description of and/or and buf/not type Gates	L+D	PPT	1	21	10/10/2020
21.	Rise, Fall and Turn-off delays	L+D	PPT	1	22	13/10/2020
22.	min, max and typical delays	L+D	PPT	1	23	15/10/2020
23.	Continuous assignments	L+D	PPT	1	24	16/10/2020
24.	Delay specification, Expressions	L+D	PPT	1	25	17/10/2020
25.	Operators, Operands, Operator types.	L+D	PPT	1	26	20/10/2020
26.	Examples	L+D	PPT	1	27	22/10/2020
MODULE 4: Behavioral Modeling						
27.	Structured procedure, initial statement	L+D	PPT	1	28	23/10/2020
28.	always statement	L+D	PPT	1	29	24/10/2020
29.	blocking and non-blocking statements	L+D	PPT	1	30	27/10/2020
30.	delay control, generate statement	L+D	PPT	1	31	29/10/2020
31.	conditional statements,multiway branching	L+D	PPT	1	32	5/11/2020
32.	loops-while loop, for loop	L+D	PPT	1	33	6/11/2020
33.	loops-Repeat, forever	L+D	PPT	1	34	7/11/2020
34.	sequential and parallel blocks	L+D	PPT	1	35	10/11/2020
35.	Examples	L+D	PPT	1	36	12/11/2020
MODULE 5:Useful Modeling Techniques:						
36.	Procedural continuous assignments	L+D	PPT	1	37	13/11/2020
37.	overriding parameters	L+D	PPT	1	38	14/11/2020
38.	conditional compilation and execution	L+D	PPT	1	39	17/11/2020
39.	useful system tasks	L+D	PPT	1	40	19/11/2020

40	**Logic Synthesis with Verilog:** Logic Synthesis	LCD	PPT	1	41	20/11/2020
41	Impact of logic synthesis	LCD	PPT	1	42	21/11/2020
42	Verilog HDL Synthesis,	LCD	PPT	1	43	24/11/2020
43	Synthesis design flow	LCD	PPT	1	44	26/11/2020
44	Verification of Gate Level Netlist	LCD	PPT	1	45	27/11/2020

Text Books:

1. Samir Palnitkar, —Verilog HDL: A Guide to Digital Design and Synthesis", Pearson Education, Second Edition.

Reference Books:

1. Donald E. Thomas, Philip R. Moorby, —The Verilog Hardware Description Language‖, Springer Science+Business Media, LLC, Fifth edition.
2. Michael D. Ciletti, —Advanced Digital Design with the Verilog HDL‖ Pearson (Prentice Hall), Second edition.
3. Padmanabhan, Tripura Sundari, —Design through Verilog HDL‖, Wiley, 2016 or earlier.

WEB MATERIALS:

W1: https://www.youtube.com/watch?v=wiNDn19GpRU
W2:https://www.youtube.com/watch?v=PybxgAroozA
W3:https://www.youtube.com/watch?v=GRR6VMj9-hI

Details for the teaching Aids

1. Whatsapp usage for all communications
2. Zoom and Microsoft-Teams tools usage for online classes

Signature of Course In charge **Signature of Module Coordinator** **Signature of HOD**

KS INSTITUTE OF TECHNOLOGY BANGALORE
DEPARTMENT OF ELECRONICS & COMMUNICATION ENGINEERING

NAME OF THE STAFF : **Sunil Kumar G R**
SUBJECT CODE/NAME : **18EC56/VERILOG HDL**
SEMESTER/YEAR : **V / III (B sec)**
ACADEMIC YEAR : **2020-2021**

Sl. No.	Topic to be covered	Mode of Delivery	Teaching Aid	No. of Periods	Cumulative No. of Periods	Proposed Date
MODULE 1:Overview of Digital Design with Verilog HDL &Hierarchical Modeling Concepts						
1.	Evolution of CAD, emergence of HDLs	L+D	PPT	1	1	1/9/2020
2.	Typical HDL-flow	L+D	PPT	1	2	3/9/2020
3.	why Verilog HDL?, Trends in HDLs	L+D	PPT	1	3	4/9/2020
4.	Top-down and bottom-up design methodology	L+D	PPT	2	5	5/9/2020 8/9/2020
5.	Differences between modules and module instances	L+D	PPT	1	6	10/9/2020
6.	Parts of a simulation, Design block	L+D	PPT	1	7	11/9/2020
7.	Stimulus block., Examples	L+D	PPT	1	8	12/9/2020
MODULE 2:Basic Concepts, Modules and Ports						
8.	Lexical conventions	L+ D	PPT	1	9	15/9/2020
9.	Data types	L+D	PPT	1	10	18/9/2020
10.	Data types	L+D	PPT	1	11	19/9/2020
11.	System tasks	L+D	PPT	1	12	22/9/2020
12.	Compiler directives	L+D	PPT	1	13	24/9/2020
13.	Compiler directives, examples	L+D	PPT	1	14	25/9/2020
14.	Module definition	L+D	PPT	1	15	26/9/2020

15.	Port declaration	L+D	PPT	1	16	29/9/2020
16.	Connecting ports	L+D	PPT	1	17	1/10/2020
17.	Hierarchical name referencing	L+D	PPT	1	18	3/10/2020
MODULE 3: Gate-Level Modeling & Dataflow Modeling						
18.	Modeling using basic Verilog gate primitives	L+D	PPT	1	19	8/10/2020
19.	Description of and/or and buf/not type Gates	L+D	PPT	1	20	9/10/2020
20.	Description of and/or and buf/not type Gates	L+D	PPT	1	21	10/10/2020
21.	Rise, Fall and Turn-off delays	L+D	PPT	1	22	13/10/2020
22.	min, max and typical delays	L+D	PPT	1	23	15/10/2020
23.	Continuous assignments	L+D	PPT	1	24	16/10/2020
24.	Delay specification, Expressions	L+D	PPT	1	25	17/10/2020
25.	Operators, Operands, Operator types.	L+D	PPT	1	26	20/10/2020
26.	Examples	L+D	PPT	1	27	22/10/2020
MODULE 4: Behavioral Modeling						
27.	Structured procedure, initial statement	L+D	PPT	1	28	23/10/2020
28.	always statement	L+D	PPT	1	29	24/10/2020
29.	blocking and non-blocking statements	L+D	PPT	1	30	27/10/2020
30.	delay control, generate statement	L+D	PPT	1	31	29/10/2020
31.	conditional statements,multiway branching	L+D	PPT	1	32	5/11/2020
32.	loops-while loop, for loop	L+D	PPT	1	33	6/11/2020
33.	loops-Repeat, forever	L+D	PPT	1	34	7/11/2020
34.	sequential and parallel blocks	L+D	PPT	1	35	10/11/2020
35.	Examples	L+D	PPT	1	36	12/11/2020
MODULE 5:Useful Modeling Techniques:						
36.	Procedural continuous assignments	L+D	PPT	1	37	13/11/2020
37.	overriding parameters	L+D	PPT	1	38	14/11/2020
38.	conditional compilation and execution	L+D	PPT	1	39	17/11/2020
39.	useful system tasks	L+D	PPT	1	40	19/11/2020

40.	**Logic Synthesis with Verilog:** Logic Synthesis	L+D	PPT	I	41	20/11/2020
41.	Impact of logic synthesis	L+D	PPT	I	42	21/11/2020
42.	Verilog HDLSynthesis,	L+D	PPT	I	43	24/11/2020
43.	Synthesis design flow	L+D	PPT	I	44	26/11/2020
44.	Verification of Gate-Level Netlist	L+D	PPT	I	45	27/11/2020

Text Books:

1. Samir Palnitkar, —Verilog HDL: A Guide to Digital Design and Synthesis", Pearson Education, Second Edition.

Reference Books:

1. Donald E. Thomas, Philip R. Moorby, —The Verilog Hardware Description Language||, Springer Science+Business Media, LLC, Fifth edition.
2. Michael D. Ciletti, —Advanced Digital Design with the Verilog HDL|| Pearson (Prentice Hall), Second edition.
3. Padmanabhan, Tripura Sundari, —Design through Verilog HDL||, Wiley, 2016 or earlier.

WEB MATERIALS:

W1: https://www.youtube.com/watch?v=wiNDn19GpRU
W2:https://www.youtube.com/watch?v=PybxgAroozA
W3:https://www.youtube.com/watch?v=GRR6VMj9-hI

Details for the teaching Aids

1. Whatsapp usage for all communications
2. Zoom and Microsoft-Teams tools usage for online classes

Signature of Course In charge **Signature of Module Coordinator** **Signature of HOD**

K. S. INSTITUTE OF TECHNOLOGY, BANGALORE – 560109

Department of Electronics and Communication

Assignment – 1

Course Title : VERILOG HDL
Course Code : 18EC56

Date: 26-09-2020
Marks: 10

Note: Write the Answers for the following questions and submit it before 3-10-2020

Q No.	Question	Marks	CO Mapping	K-Level
1	Explain a typical design flow for designing VLSI IC circuits using the block diagram.	1	CO1	K3 Applying
2	Explain the different levels of Abstraction used for programming in Verilog.	1	CO1	K3 Applying
3	Explain a top-down design methodology and a bottom-up design methodology.	1	CO1	K3 Applying
4	Explain the factors that have made Verilog HDL popular	1	CO1	K3 Applying
5	Explain the trends in HDLs.	1	CO1	K3 Applying
6	Explain $display and $monitor tasks with examples.	1	CO1	K3 Applying
7	Write a note on i) Comments ii) Number Specification iii) X and Z values and iv) Identifiers and Keywords with suitable examples.	1	CO2	K3 Applying
8	Explain the following data types with an example in Verilog module: (i) Nets, (ii) Register, (iii) Integers, (iv) Real, (v) Time Register	1	CO2	K3 Applying
9	Explain Compiler Directives 'define and 'include with examples.	1	CO2	K3 Applying
10	Explain the following data types with an example in Verilog module: (i) Arrays, (ii) Memories, (iii) Parameters, (iv) Strings.	1	CO2	K3 Applying

Course In-charge

Module Coordinator

HOD-ECE

K. S. INSTITUTE OF TECHNOLOGY, BANGALORE – 560109

Department of Electronics and Communication

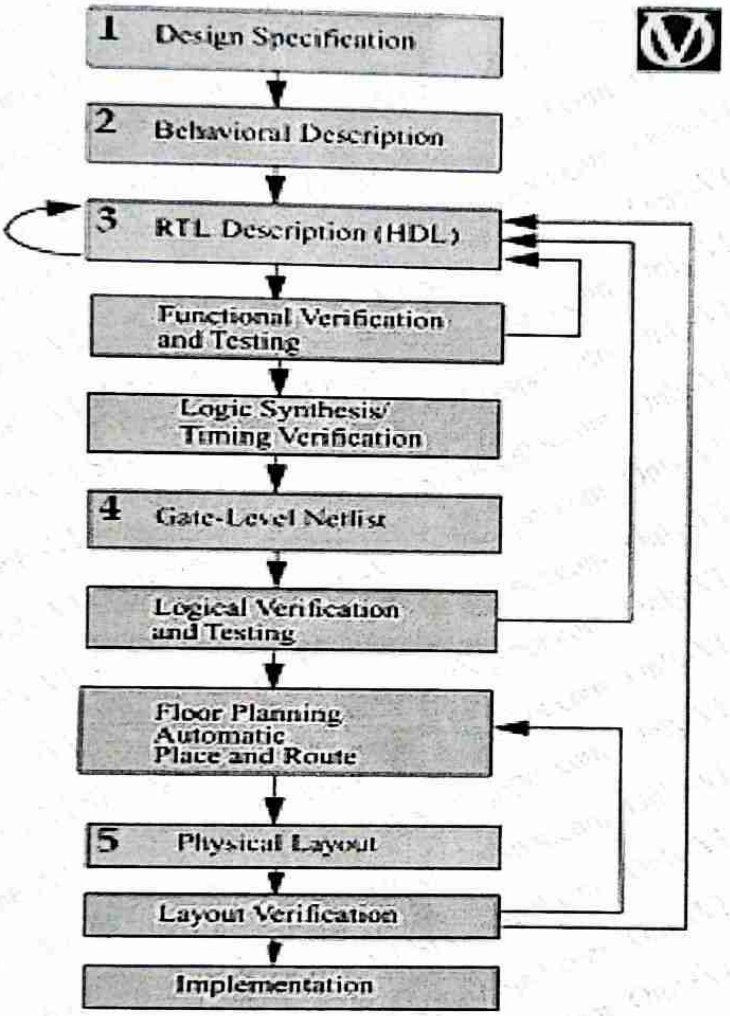
Assignment –1 Question and Scheme

Course Title : VERILOG HDL
Course Code : 18EC56 (AY-2020-21)

Date: 26-09-2020
Marks: 10

Note: Write the Answers for the following questions and submit it before 3-10-2020

1. **Explain a typical design flow for designing VLSI IC circuits using the block diagram.**



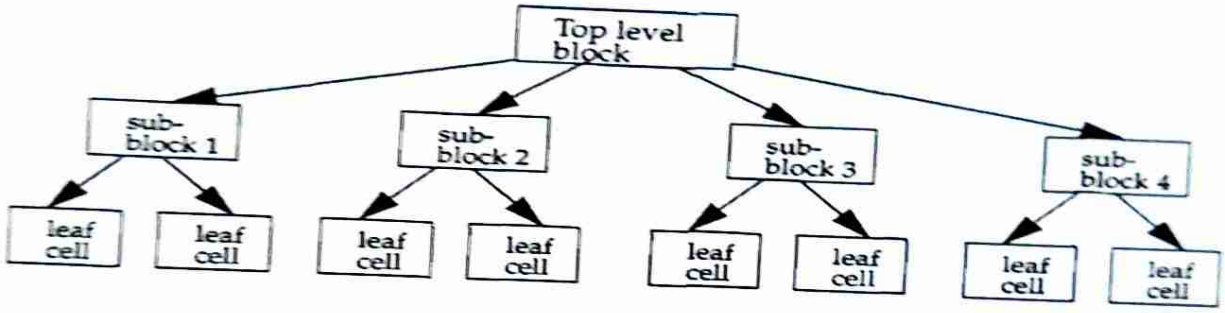
- Designs can be described at a very abstract level by use of HDLs
- Designers can write their RTL description without choosing a specific fabrication technology
- If a new technology emerges, designers do not need to redesign their circuit
- The logic synthesis tool will optimize the circuit in area and timing for the new technology
- Functional verification of the design can be done early in the design cycle
- Designing with HDLs is analogous to computer programming

2. **Explain the different levels of Abstraction used for programming in Verilog.**

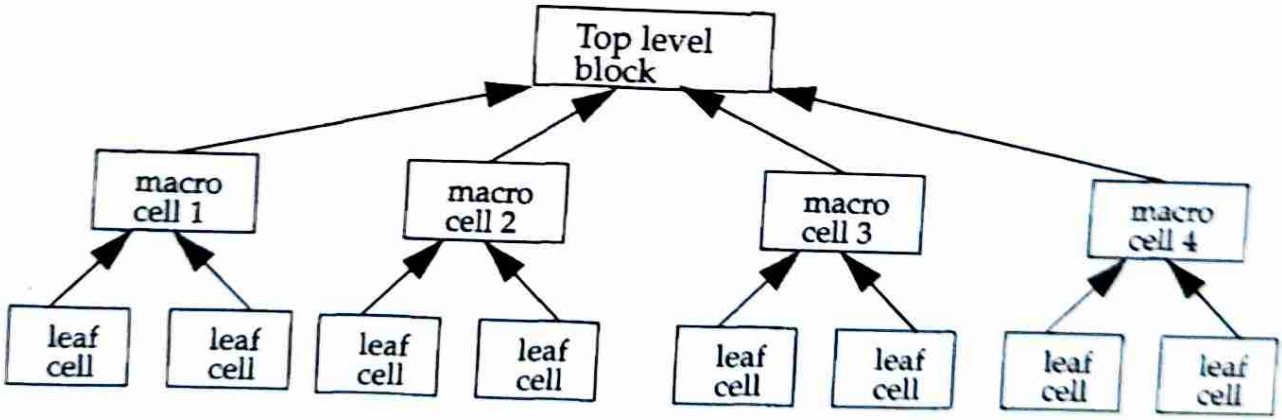
Behavioral or Algorithmic	• Highest level of Abstraction • Implemented in terms of the desired design algorithm without concern for the H/W implementation details
Dataflow	• The designer should be aware of how data flows b/w H/W registers • And how the data is processed in the design
Gate Level	• Implemented in terms of logic gates and interconnection between these gates • Similar to describing a design in terms of a gate level logic diagram
Switch level	• Lowest level of abstraction provided by Verilog • Implemented in terms of switches, storage nodes and the interconnection between them

3. **Explain a top-down design methodology and a bottom-up design methodology.**

Top-Down Design Methodology:



Bottom-Up Design Methodology:



4. **Explain the factors that have made Verilog HDL popular.**

- It is a general purpose language: easy to learn & easy to use– similar in syntax to the C
- It allows different levels of abstraction to be mixed in the same model: gate, RTL, or Behavioral design
- Most popular logic synthesis tools support Verilog HDL– making HDL popular among designers
- It allows the widest choice of vendors because all fabrication vendors provide Verilog HDL libraries for post-logic synthesis simulation
- Designers can customize a Verilog HDL simulator to their needs with the Programming Language Interface (PLI).

5. **Explain the trends in HDLs.**

- The speed and complexity of digital circuits has increased rapidly
- The most popular trend currently is to design in HDL at an RTL level, because logic synthesis tools can create gate-level net lists from RTL level design
- *Formal verification* techniques are also appearing on the horizon
- For very high speed and timing-critical circuits like microprocessors, designers often mix gate-level description directly into the RTL description to achieve optimum results
- A trend that is emerging for system-level design is a mixed bottom-up methodology

6. Explain $display and $monitor tasks with examples.

$display:

- ***$display*** is the main system task for displaying values of variables or strings or expressions
- This is one of the most useful tasks in Verilog
- ***Usage: $display(p1, p2, p3 ,....., pn); p1, p2, p3, ..., pn*** can be quoted strings or variables or expressions
- The format of ***$display*** is very similar to ***printf*** in *C*
- A ***$display*** inserts a newline at the end of the string by default
- ***A $display*** without any arguments produces a newline
- Strings can be formatted by using the format specifications

$monitor:

- Verilog provides a mechanism to monitor a signal when its value changes
- This facility is provided by the $monitor task
- Usage: $monitor(p1 ,p2,p3 ,...., pn);
- The parameters p1, p2, ... , pn can be variables, signal names, or quoted strings
- **A** format similar to the **$display** task is used in the $monitor task
- $monitor continuously monitors the values of the variables or signals specified in the parameter list and displays all parameters in the list whenever the value of any one variable or signal changes
- Unlike $display, $monitor needs to be invoked only once
- Only one monitoring list can be active at a time
- If there is more than one **$monitor** statement in your simulation, the last **$monitor** statement will be the active statement
- The earlier $monitor statements will be overridden

7. Write a note on i) Comments ii) Number Specification iii) X and Z values and iv) Identifiers and Keywords with suitable examples.

i) Comments:

- Comments can be inserted in the code for readability and documentation
- A one-line comment starts with "//". Verilog skips from that point to the end of line
- A multiple-line comment starts with "/*" and ends with "*/"
- Multiple-line comments cannot be nested

ii) Number Specification:

- Sized Numbers
- Unsized Numbers
- X or Z Values
- Negative Numbers

iii) X and Z values:

- Verilog has two symbols for unknown and high impedance values
- These values are very important for modeling real circuits
- An unknown value is denoted by an X
- A high impedance value is denoted by Z

- An X or Z sets four bits for a number in the hexadecimal base, three bits for a number in the oct base, and one bit for a number in the binary base.
- If the most significant bit of a number is of X, or Z, the number is automatically extended to fill the most significant bits, respectively, with 0, X, or Z.

iv) Identifiers and Keywords

- reg value; // reg is a keyword; value is an identifier
- **input clk; // input is a keyword, clk is an identifier**

8. Explain the following data types with an example in Verilog module: (i) Nets, (ii) Register, (iii) Integers, (iv) Real, (v) Time Register

Nets:

```
wire a; // Declare net a for the above circuit
wire b,c; // Declare two wires b, c for the above circuit
wire d = 1'bO; // Net d is fixed to logic value 0 at declaration.
```

Registers:

```
reg reset; /* declare a variable reset that can hold its value*/
initial // this construct will be discussed later
begin
reset = 1'b1; /*initialize reset to 1 to reset the digital circuit*/
#100 reset = 1'b0; /*after 100 time units reset is deasserted.*/
end
```

Integer:

```
integer counter; /*general purpose variable used as a counter*/
initial
counter = -1; // A negative one is stored in the counter
```

Real

```
real delta; // Define a real variable called delta
initial
begin
. delta = 4e10; // delta is assigned in scientific notation
delta = 2.13; // delta is assigned a value 2.13
end
integer i; // Define an integer i
initial
i = delta; // i gets the value 2 (rounded value of 2.13)
```

9. Explain Compiler Directives 'define and 'include with examples.

'define

- The ***'define*** directive is used to define text macros in Verilog
- This is similar to the *#define* construct in *C*
- The defined constants or text macros are used in the Verilog code by preceding them with a ' (back tick)
- The Verilog compiler substitutes the text of the macro wherever it encounters a ' <*macro-name*>.

```
//define a text macro that defines default word size
//Used as 'WORD_SIZE in the code
'define WORD_SIZE 32
/*define an alias. A $stop will be substituted wherever 'S appears*/
'define S $stop.
//define a frequently used text string
'define WORD_REG reg [31:0]
// you can then define a 32-bit register as 'WORD_REG reg32*/
```

'include

- The ' **include** directive allows you to include entire contents of a Verilog source file in another Verilog file during compilation
- This works similarly to the *#include* in the *C* programming language
- This directive is typically used to include header files, which typically contain global or commonly used definitions

```
/* Include the file header.v, which contains declarations in the main Verilog file design.v */
'include header.v
...
...
<Verilog code in file design.v>
...
...
```

10. Explain the following data types with an example in Verilog module: (i) Arrays, (ii) Memories, (iii) Parameters, (iv) Strings.

Arrays:

```
integer count[0:7]; //an array of 8 count variables
reg bool[31:0]; //Array of 32 one-bit boolean register variables
time chk_qoint[1:1OO]; //Array of 100 time checkpoint variables
reg [4 : 0] port_id[O : 7] ; /*Array of 8 port-ids; each port-id is 5 bits wide*/
integer matrix[4:0][0:255]; // two dimensional array of integers
reg [63:0] array_4d [15:0][7:0][7:0][255:0]; /* 4 dimensional array*/
Wire [7:0] w_array2 [5:0]; /*declare an array of 8 bit vector wire*/
Wire w_array1 [7:0] [5:0];/*declare an array of single bit wire*/
```

Memories:

```
reg mem1bit[0:1023]; //memory mem1bit with 1K 1-bit words
reg [7 : 0] membyte [O : 1023]; /* memory membyte with 1K 8-bit words (bytes)*/
membyte[511] //Fetches 1 byte word whose address is 511
```

Parameters:

```
parameter port_id = 5;//Defines a constant port_id
parameter cache_line_width=256;//constant defines width of cache line
Parameter signed [15:0] WIDTH; /*Fixed sign and range for parameter WIDTH*/
```

Strings:

```
reg [8*18:1] string_value; /*Declare a variable that is 18 bytes wide*/
initial
        string-value = "Hello Verilog World"; /*String can be stored in variable*/
```

Course In-charge | Module Coordinator | HOD-ECE

K. S. INSTITUTE OF TECHNOLOGY, BANGALORE – 560109

Department of Electronics and Communication

Assignment – II

Course Title : VERILOG HDL
Course Code : 18EC56

Date: 06-11-2020
Marks: 10

Note: Write the Answers for the following questions and submit it before 16-10-2020

Q No.	Question	Marks	CO Mapping	K-Level
1	Explain the port connection rules.	1	CO2	K3 Applying
2	Explain the two methods of connecting ports to external signals with an example.	1	CO2	K3 Applying
3	Discuss different types of Gates with respect to logic symbols, gate instantiation and truth tables.	1	CO3	K3 Applying
4	What are rise, fall and turnoff delays? How they are specified in Verilog.	1	CO3	K3 Applying
5	Write a Verilog code and stimulus in gate level description for 4-bit full Adder.	1	CO3	K3 Applying
6	Explain regular assignment delay, implicit assignment delay, net declaration delay for continuous assignment statements.	1	CO3	K3 Applying
7	Write the Verilog code for 4-to-1Multiplexer using (i) Logic Equations and (ii) Conditional Operator.	1	CO3	K3 Applying
8	Write a Verilog dataflow description for 4-bit Full Adder with carry look ahead.	1	CO3	K3 Applying
9	What would be the output of the following a= 4'b0111, and b= 4'b1001, (i) &b, (ii) a<<2, (iii) {a,b}, (iv) {2{b},a}, (v) a^b, (vi) a|b	1	CO4	K3 Applying
10	Write a Verilog dataflow description for 8-to-3 encoder with priority.	1	CO4	K3 Applying

Course In-charge

Module Coordinator

HOD-ECE

K. S. INSTITUTE OF TECHNOLOGY, BANGALORE – 560109

Department of Electronics and Communication

Assignment – II Question and Scheme

Course Title : VERILOG HDL
Course Code : 18EC56 (AY-2020-21)

Date: 06-11-2020
Marks: 10

Note: Write the Answers for the following questions and submit it before 16-10-2020

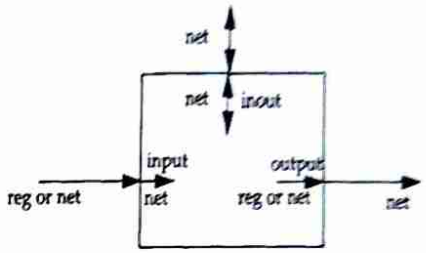
1. Explain the port connection rules.

Inputs

- Internally, input ports must always be of the type *net.*
- Externally, the inputs can be connected to a variable which is a reg or a *net.*

Outputs

- Internally, outputs ports can be of the type reg or *net*
- Externally, outputs must always be connected to a *net*
- They cannot be connected to a reg



Inouts

- Internally, inout ports must always be of the type *net*
- Externally, inout ports must always be connected to a *net*

Width matching

- It is legal to connect internal and external items of different sizes when making inter-module port connections
- However, a warning is typically issued that the widths do not match.

Unconnected ports

- Verilog allows ports to remain unconnected
- For example, certain output ports might be simply for debugging, and you might not be interested in connecting them to the external signals
- You can let a port remain unconnected by instantiating a module as shown below

2. Explain the two methods of connecting ports to external signals with an example.

Connecting by ordered list

```
module Top;
//Declare connection variables
reg [3:0]A,B;
reg C_IN;
wire [3:0] SUM;
wire C_OUT;
//Instantiate fulladd4, call it fa_ordered.
//Signals are connected to ports in order (by position)
fulladd4 fa_ordered(SUM, C_OUT, A, B, C_IN);
...
<stimulus>
...
Endmodule

module fulladd4 (sum, c_out, a, b, c_in) ;
output [3 : 0] sum;
output c_out;
input [3:0] a, b;
input c_in;
...
```

<module internals>
...
endmodule

Connecting ports by name

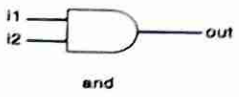
/* Instantiate module fa_byname and connect signals to ports by name*/
fulladd4 fa_byname(.c_out(C_OUT), .sum(SUM), .b(B), .c_in(C_IN), .a(A) ,);

/*Instantiate module fa_byname and connect signals to ports by name*/
fulladd4 fa_byname(. sum(SUM) , .b(B) , .c_in (C_IN) , .a (A) ,) ;

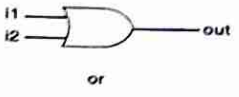
3. Discuss different types of Gates with respect to logic symbols, gate instantiation and truth tables.

And/or Gates:

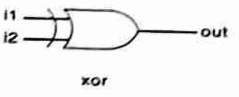
and	or	xor
nand	nor	xnor



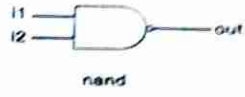
and



or

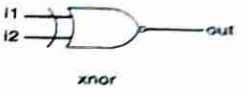


xor



nand

nor



xnor

or	i1 0	i1 1	i1 x	i1 z
i2 0	0	1	x	x
i2 1	1	1	1	1
i2 x	x	1	x	x
i2 z	x	1	x	x

nor	i1 0	i1 1	i1 x	i1 z
i2 0	1	0	x	x
i2 1	0	0	0	0
i2 x	x	0	x	x
i2 z	x	0	x	x

xor	i1 0	i1 1	i1 x	i1 z
i2 0	0	1	x	x
i2 1	1	0	x	x
i2 x	x	x	x	x
i2 z	x	x	x	x

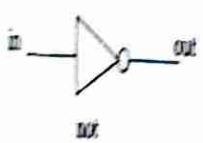
xnor	i1 0	i1 1	i1 x	i1 z
i2 0	1	0	x	x
i2 1	0	1	x	x
i2 x	x	x	x	x
i2 z	x	x	x	x

and	i1 0	i1 1	i1 x	i1 z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

nand	i1 0	i1 1	i1 x	i1 z
i2 0	1	1	1	1
i2 1	1	0	x	x
i2 x	1	x	x	x
i2 z	1	x	x	x

```
wire OUT, IN1, IN2;

// basic gate instantiations.
and a1(OUT, IN1, IN2);
nand na1(OUT, IN1, IN2);
or or1(OUT, IN1, IN2);
nor nor1(OUT, IN1, IN2);
xor x1(OUT, IN1, IN2);
xnor nx1(OUT, IN1, IN2);

// More than two inputs; 3 input nand g
nand na1_3inp(OUT, IN1, IN2, IN3);

// gate instantiation without instance
and (OUT, IN1, IN2); // legal gate ins
```

Buf /Not Gates:



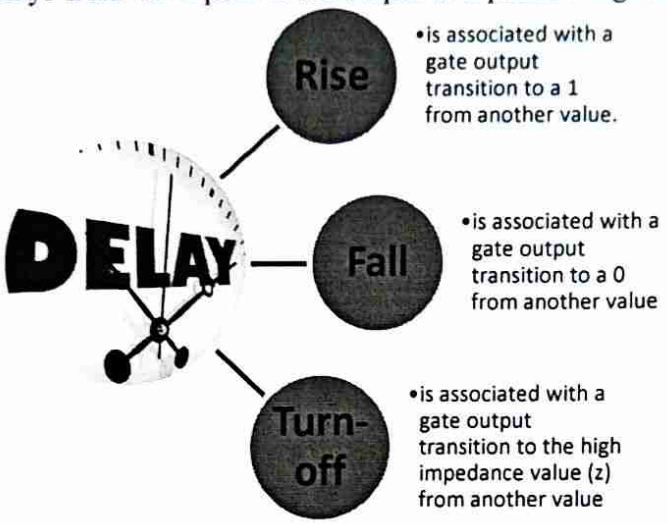
buf	in	out
	0	0
	1	1
	x	x
	z	x

not	in	out
	0	1
	1	0
	x	x
	z	x

```
// basic gate instantiations.
buf b1(OUT1, IN);
not n1(OUT1, IN);

// More than two outputs
buf b1_2out(OUT1, OUT2, IN);

// gate instantiation without instance name
not (OUT1, IN); // legal gate instantiation
```

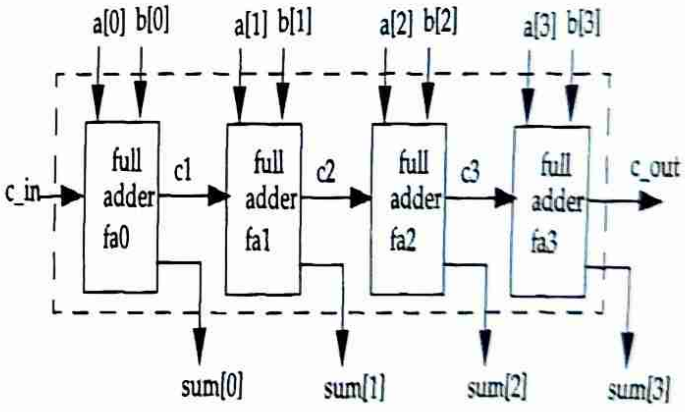
4. What are rise, fall and turnoff delays? How they are specified in Verilog.

There are three types of delays from the inputs to the output of a primitive gate



5. Write a Verilog code and stimulus in gate level description for 4-bit full Adder.

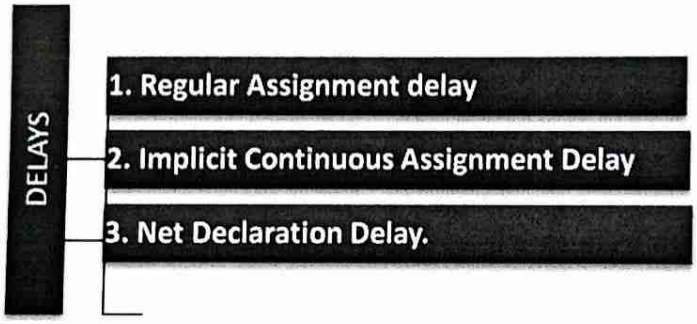
```
// Define a 4-bit full adder
module fulladd4(sum, c_out, a, b, c_in);

// I/O port declarations
output [3:0] sum;
output c_out;
input[3:0] a, b;
input c_in;

// Internal nets
wire c1, c2, c3;

// Instantiate four 1-bit full adders.
fulladd fa0(sum[0], c1, a[0], b[0], c_in);
fulladd fa1(sum[1], c2, a[1], b[1], c1);
fulladd fa2(sum[2], c3, a[2], b[2], c2);
fulladd fa3(sum[3], c_out, a[3], b[3], c3);

endmodule
```



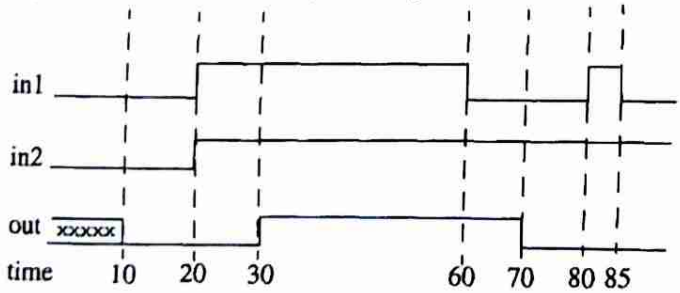

```
// Define the stimulus (top level module)
module stimulus;

// Set up variables
reg [3:0] A, B;
reg C_IN;
wire [3:0] SUM;
wire C_OUT;

// Instantiate the 4-bit full adder. call it FA1_4
fulladd4 FA1_4(SUM, C_OUT, A, B, C_IN);


// Setup the monitoring for the signal values
```

```
initial
begin
  $monitor($time," A= %b, B=%b, C_IN= %b, --- C_OUT= %b, SUM= %b\n",
                                A, B, C_IN, C_OUT, SUM);

end

// Stimulate inputs
initial
begin
  A = 4'd0; B = 4'd0; C_IN = 1'b0;

  #5 A = 4'd3; B = 4'd4;

  #5 A = 4'd2; B = 4'd5;

  #5 A = 4'd9; B = 4'd9;

  #5 A = 4'd10; B = 4'd15;

  #5 A = 4'd10; B = 4'd5; C_IN = 1'b1;
end

endmodule
```

6. **Explain regular assignment delay, implicit assignment delay, net declaration delay for continuous assignment statements.**



assign $10 out = in1 & in2; // Delay in a continuous assign

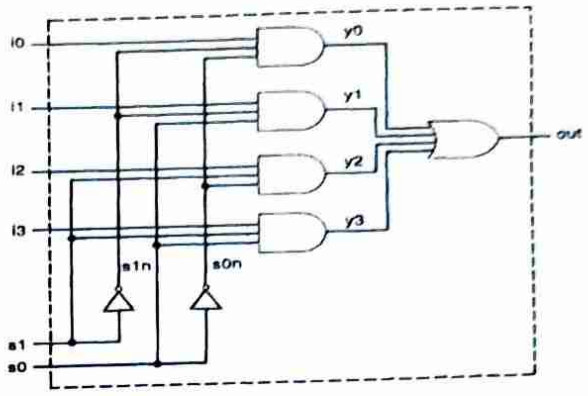


//implicit continuous assignment delay
wire #10 out = in1 & in2;
//same as
wire out;
assign #10 out = in1 & in2;

//Net Delays
wire # 10 out;
assign out = in1 & in2;
//The above statement has the same effect as the following
wire out;
assign #10 out = in1 & in2;

7. **Write the Verilog code for 4-to-1Multiplexer using (i) Logic Equations and (ii) Conditional Operator.**

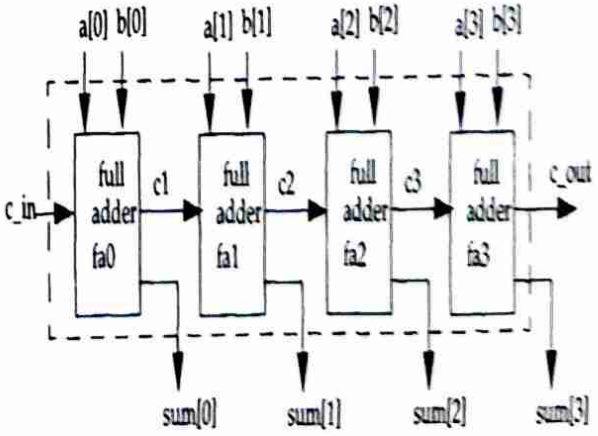
```
module mux4-to-1 (out, i0, i1, i2, i3, s1, s0);
output out;
input i0, i1, i2, i3;
input s1, S0;
wire s1n, s0n;
wire y0, y1, y2, y3;
not (s1n, s1) ;
not (s0n, s0);
and (y0, i0, s1n, s0n);
and (y1, i1, s1n, s0);
and (y2, i2, s1, s0n);
and (y3, i3, s1, s0);
or (out, y0, y1, y2, y3);
endmodule
```




```
// Define the stimulus module (no ports)

// Stimulate the inputs
initial
begin
  // set input lines
  IN0 = 1; IN1 = 0; IN2 = 1; IN3 = 0;
  #1 $display("IN0= %b, IN1= %b, IN2= %b, IN3=
%b\n",IN0,IN1,IN2,IN3);

  // choose IN0
  S1 = 0; S0 = 0;
  #1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUT

  // choose IN1
  S1 = 0; S0 = 1;
  #1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUT

  // choose IN2
  S1 = 1; S0 = 0;
  #1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUT

  // choose IN3
  S1 = 1; S0 = 1;
  #1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUT
end

endmodule
```

```
// Define the stimulus module (no ports)
module stimulus;

// Declare variables to be connected
// to inputs
reg IN0, IN1, IN2, IN3;
reg S1, S0;

// Declare output wire
wire OUTPUT;

// Instantiate the multiplexer
mux4_to_1 mymux(OUTPUT, IN0, IN1, IN2, IN3, S1, S0
```

8. **Write a Verilog dataflow description for 4-bit Full Adder with carry look ahead.**



```
// Define a 4-bit full adder
module fulladd4(sum, c_out, a, b, c_in);

// I/O port declarations
output [3:0] sum;
output c_out;
input[3:0] a, b;
input c_in;

// Internal nets
wire c1, c2, c3;

// Instantiate four 1-bit full adders.
fulladd fa0(sum[0], c1, a[0], b[0], c_in);
fulladd fa1(sum[1], c2, a[1], b[1], c1);
fulladd fa2(sum[2], c3, a[2], b[2], c2);
fulladd fa3(sum[3], c_out, a[3], b[3], c3);

endmodule
```

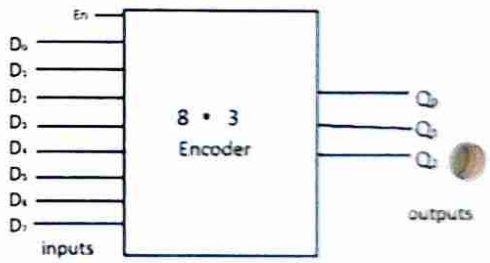
9. What would be the output of the following a= 4'b0111, and b= 4'b1001, (i) &b, (ii) a<<2, (iii) {a,b}, (iv) {2{b},a}, (v) a^b, (vi) a|b

(i) &b= 0; (ii) A<<2= 1100; (iii) {a,b}= 8'b01111001, (iv) {2{b},a}= 12'b100110010111, (v) a^b= 1110; (vi) a | b= 1111

10. Write a Verilog dataflow description for 8-to-3 encoder with priority.

8-to-3 Encoder without priority:

```
module encoder(D, Q);
   input [7:0] D;
   output [2:0] Q;
       reg [2:0] Q;
       always @ (D)
       begin
       case(D)
       8'b00000001:Q<=3'b000;
       8'b00000010:Q<=3'b001;
      8'b00000100:Q<=3'b010;
      8'b00001000:Q<=3'b011;
      8'b00010000:Q<=3'b100;
      8'b00100000:Q<=3'b101;
      8'b01000000:Q<=3'b110;
      8'b10000000:Q<=3'b111;
      default: Q<=3'bXXX;
      endcase
      end
endmodule
```



Course In-charge

Module Coordinator

HOD-ECE

K. S. INSTITUTE OF TECHNOLOGY, BANGALORE – 560109

Department of Electronics and Communication

Assignment – III

Course Title : VERILOG HDL
Course Code : 18EC56

Date: 02-1-2021
Marks: 10

Q No.	Question	Marks	CO Mapping	K-Level
1	Explain sequential blocks and parallel blocks with examples.	1	CO4	K3 Applying
2	Chart out the differences between tasks and functions.	1	CO4	K3 Applying
3	Discuss special features of block statements.	1	CO4	K3 Applying
4	Illustrate Tasks and Functions with respect to: (i) Usage condition, (ii) Declaration and Invocation, and (iii) Examples	1	CO4	K3 Applying
5	Explain conditional compilation and conditional execution with examples.	1	CO5	K3 Applying
6	Discuss the two ways to override parameter values with examples.	1	CO5	K3 Applying
7	What is logic synthesis? Illustrate the designer's mind as the logic synthesis tool.	1	CO5	K3 Applying
8	Discuss at least six limitations of design process and how automation of logic synthesis tool has addressed these limitations.	1	CO5	K3 Applying
9	Interpret the following statements into Verilog constructs: (i) assign out = (a & b) | c; (ii) assign {c_out, sum} = a+b+c_in; (iii) assign out= (s)? i1 : i0;	1	CO5	K3 Applying
10	Discuss the logic synthesis flow from RTL to Gates with a neat diagram.	1	CO5	K3 Applying

Course In-charge

Module Coordinator

HOD-ECE

K. S. INSTITUTE OF TECHNOLOGY, BANGALORE – 560109

Department of Electronics and Communication

Assignment – III Question and Scheme

Course Title : VERILOG HDL
Course Code : 18EC56 (AY-2020-21)

Date: 02-1-2021
Marks: 10

Note: Write the Answers for the following questions and submit it before 15-12-2020

1. Explain sequential blocks and parallel blocks with examples.

Sequential blocks:

```
//Illustration 1: Sequential block without delay
reg x, y;
reg [1:0] z, w;

initial
begin
        x = 1'b0;
        y = 1'b1;
        z = {x, y};
        w = {y, x};
end

//Illustration 2: Sequential blocks with delay.
reg x, y;
reg [1:0] z, w;

initial
begin
        x = 1'b0; //completes at simulation time 0
        #5 y = 1'b1; //completes at simulation time 5
        #10 z = {x, y}; //completes at simulation time 15
        #20 w = {y, x}; //completes at simulation time 35
end
```

Parallel blocks:

```
//Example 1: Parallel blocks with delay.
reg x, y;
reg [1:0] z, w;

initial
fork
        x = 1'b0; //completes at simulation time 0
        #5 y = 1'b1; //completes at simulation time 5
        #10 z = {x, y}; //completes at simulation time 10
        #20 w = {y, x}; //completes at simulation time 20
join
```

2. **Chart out the differences between tasks and functions.**

Functions	Tasks
A function can enable another function but not another task.	A task can enable other tasks and functions.
Functions always execute in 0 simulation time.	Tasks may execute in non-zero simulation time.
Functions must not contain any delay, event, or timing control statements.	Tasks may contain delay, event, or timing control statements.
Functions must have at least one **input** argument. They can have more than one **input**.	Tasks may have zero or more arguments of type **input, output or inout**.
Functions always return a single value. They cannot have **output** or **inout** arguments.	Tasks do not return with a value but can pass multiple values through **output** and **inout** arguments.

3. **Discuss special features of block statements.**

Nested blocks,

```
//Nested blocks
initial
begin
        x = 1'b0;
        fork
                #5 y = 1'b1;
                #10 z = {x, y};
        join
        #20 w = {y, x};
end
```

Named blocks,

```
//Named blocks
module top;

initial
begin: block1 //sequential block named block1
integer i; //integer i is static and local to block1
          // can be accessed by hierarchical name, top.block1.i
…
…
end

initial
fork: block2 //parallel block named block2
reg i; // register i is static and local to block2
                // can be accessed by hierarchical name, top.block2.i
…
…
join
```

Disabling of named blocks:

```
initial
begin
  flag = 16'b 0010_0000_0000_0000;
  i = 0;
  begin: block1 //The main block inside while is named block1
  while(i < 16)
   begin
       if (flag[i])
       begin
           $display("Encountered a TRUE bit at element number %d", i);
           disable block1; //disable block1 because you found true bit.
       end
       i = i + 1;
   end
  end
end
```

4. **Illustrate Tasks and Functions with respect to: (i) Usage condition, (ii) Declaration and Invocation, and (iii) Examples**

Tasks:

- There are delay, timing, or event control constructs in the procedure.
- The procedure has zero or more than one output arguments.
- The procedure has no input arguments.

Functions:

- There are no delay, timing, or event control constructs in the procedure.
- The procedure returns a single value.
- There is at least one input argument.

5. **Explain conditional compilation and conditional execution with examples.**

Conditional Compilation: Conditional compilation can be accomplished by using compiler directives 'ifdef, 'else, and 'endif.

```
//Conditional Compilation

//Example 1
`ifdef TEST //compile module test only if text macro TEST is defined
module test;
…
…
endmodule
`else //compile the module stimulus as default
module stimulus;
…
…
endmodule
`endif //completion of `ifdef statement

//Example 2
module top;

bus_master b1(); //instantiate module unconditionally
`ifdef ADD_B2
    bus_master b2(); //b2 is instantiated conditionally if text macro
                     //ADD_B2 is defined
`endif

endmodule
```

Conditional Execution:

```
//Conditional execution
module test;
...
...
initial
begin
    if($test$plusargs("DISPLAY_VAR"))
        $display("Display = %b ", {a,b,c} ); //display only if flag is set
    else
        $display("No Display"); //otherwise no display
end
```

6. Discuss the two ways to override parameter values with examples.

defparam Statement:

Example 9-2 *Defparam Statement*

```
//Define a module hello_world
module hello_world;
parameter id_num = 0; //define a module identification number = 0

initial //display the module identification number
        $display("Displaying hello_world id number = %d", id_num);
endmodule


//define top-level module
module top;
//change parameter values in the instantiated modules
//Use defparam statement
defparam w1.id_num = 1, w2.id_num = 2;

//instantiate two hello_world modules
hello_world w1();
hello_world w2();

endmodule
```

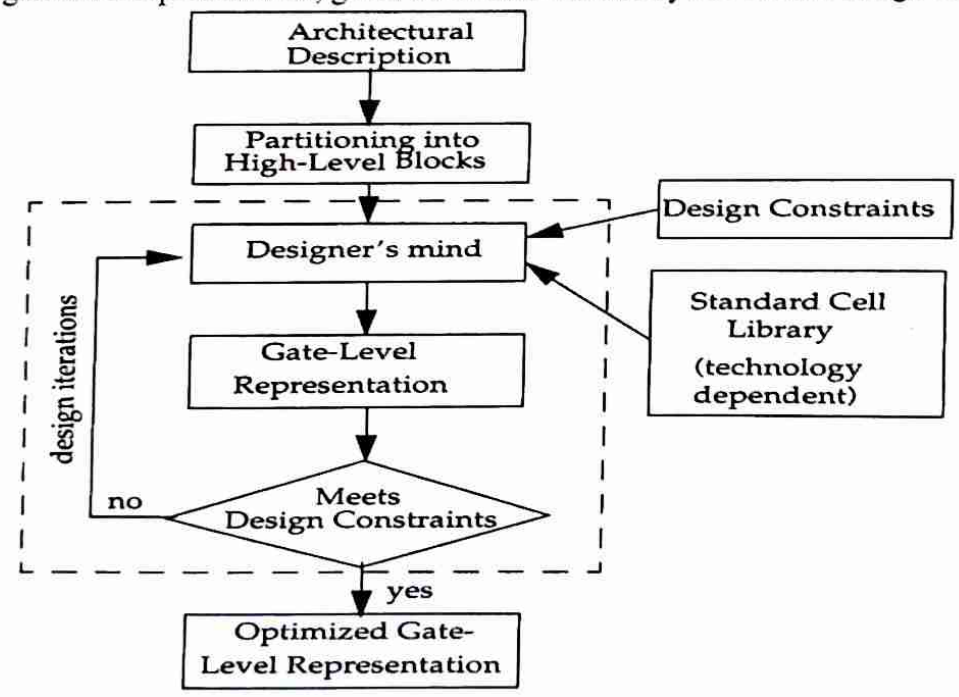
Module-Instance Parameter Values:

Example 9-3 *Module Instance Parameter Values*

```
//define module with delays
module bus_master;
parameter delay1 = 2;
parameter delay2 = 3;
parameter delay3 = 7;
…
<module internals>
…
endmodule

//top-level module; instantiates two bus_master modules
module top;

//Instantiate the modules with new delay values
bus_master #(4, 5, 6) b1(); //b1: delay1 = 4, delay2 = 5, delay3 = 6
bus_master #(9,4) b2(); //b2: delay1 = 9, delay2 = 4, delay3 = 7(default)

endmodule
```

7. What is logic synthesis? Illustrate the designer's mind as the logic synthesis tool.

Logic synthesis is the process of converting a high-level description of the design into an optimized gate-level representation, given a standard cell library and certain design constraints.



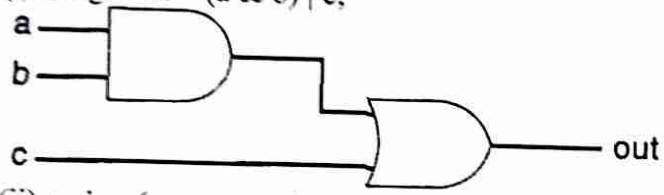
8. **Discuss at least six limitations of design process and how automation of logic synthesis tool has addressed these limitations.**

Limitations: human error, never be sure that the design constraints were going to be met, turnaround time, What-if scenarios were hard to verify, little consistency in design styles, a bug, Timing, area, and power dissipation in library cells, Design reuse was not possible.

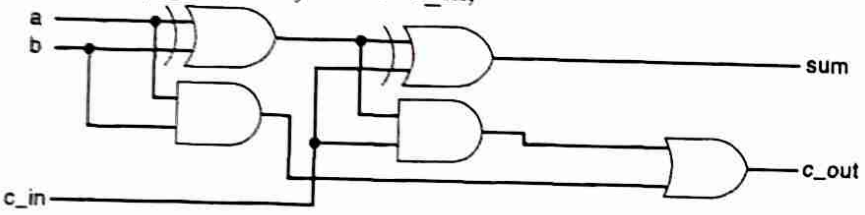
Automated logic synthesis: less prone to human error, ensure that all constraints have been met, Conversion from high-level design to gates is fast, Turnaround time for redesign of blocks is shorter, `What-if` scenarios are easy to verify, Logic synthesis tools optimize the design as a whole, the bug is eliminated, Logic synthesis tools allow `technology-independent` design, Design reuse is possible.

9. **Interpret the following statements into Verilog constructs: (i) assign out = (a & b) | c; (ii) assign {c_out, sum} = a+b+c_in; (iii) assign out= (s)? i1 : i0;**

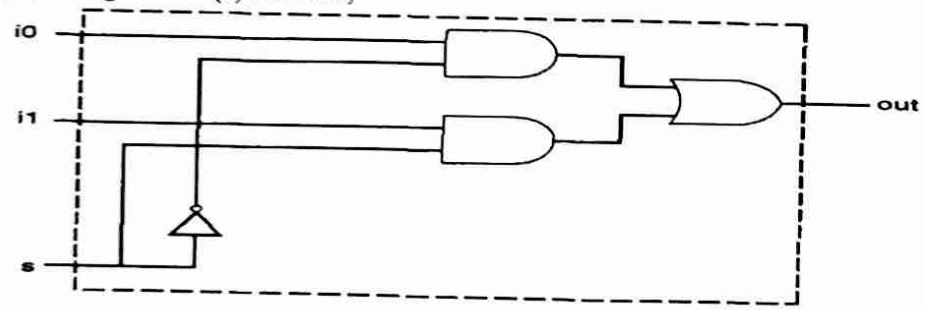
(i) assign out = (a & b) | c;



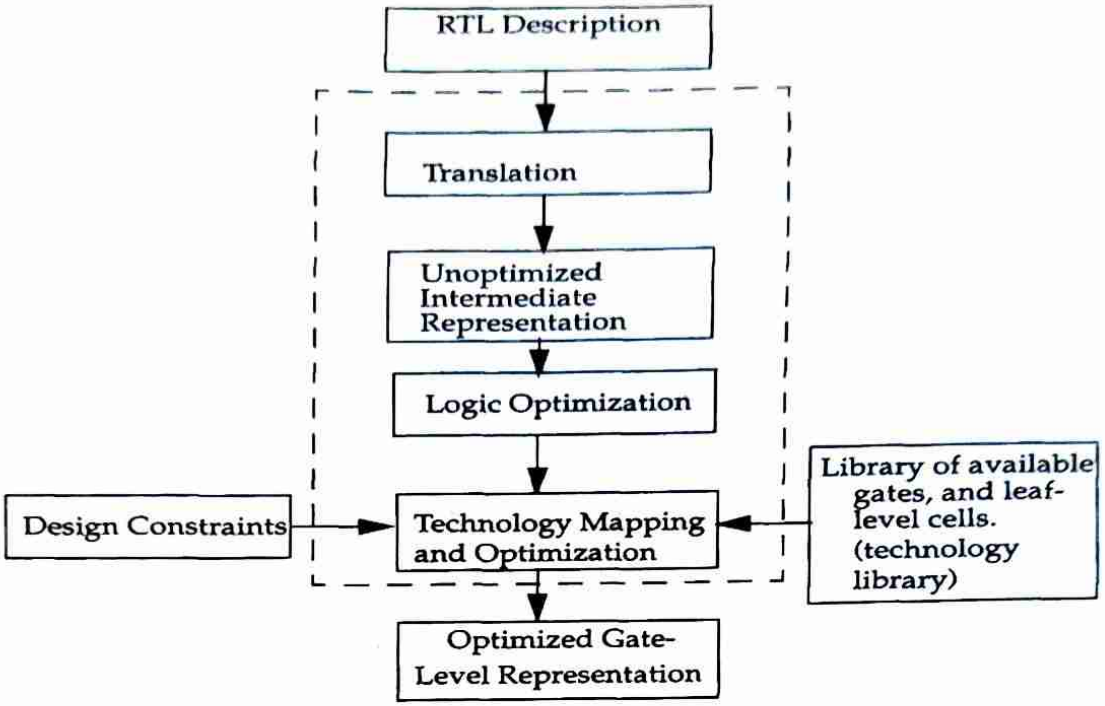
(ii) assign {c_out, sum} = a+b+c_in;



(iii) assign out= (s)? i1 : i0;



10. Discuss the logic synthesis flow from RTL to Gates with a neat diagram.



Course In-charge

Module Coordinator

HOD-ECE

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

I SESSIONAL TEST QUESTION PAPER 2020 – 21ODD SEMESTER

Set A

USN | | | | | | | | | | |

Degree	: B.E	Semester	: V A & B
Branch	: Electronics & Communication Egg.	CourseCode	: 18EC56
Course Title	: VERILOG HDL	Date	: 7-10-2020
Duration	: 90Minutes	Max Marks	: 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	CO mapping	K-Level
	PART-A			
1(a)	Demonstrate Verilog Code in Behavioral, Data Flow and Gate level abstractions for a Half Adder.	6	CO1	K-3 Applying
(b)	Classify a top-down design methodology and a bottom-up design methodology.	6	CO1	K-3 Applying
(c)	Identify the trends in HDLs.	6	CO1	K-3 Applying
	OR			
2(a)	Classify the different levels of Abstraction used for programming in Verilog.	6	CO1	K-3 Applying
(b)	Write a note on i) Instances and ii) Simulation	6	CO1	K-3 Applying
(c)	Discover the factors that have made Verilog HDL popular.	6	CO1	K-3 Applying
	PART-B			
3(a)	Write a note on i) Comments ii) Number Specification iii) X and Z values and iv) Identifiers and Keywords with suitable examples.	6	CO2	K-3 Applying
(b)	Explore Compiler Directives 'define and 'include with examples.	6	CO2	K-3 Applying
	OR			
4(a)	Explain the following data types with an example in Verilog module: (i) Arrays, (ii) Memories, (iii) Parameters, (iv) Strings.	6	CO2	K-3 Applying
(b)	Explore $display and $monitor tasks with examples.	6	CO2	K-3 Applying

Course in charge

Module Coordinator

HOD/ECE

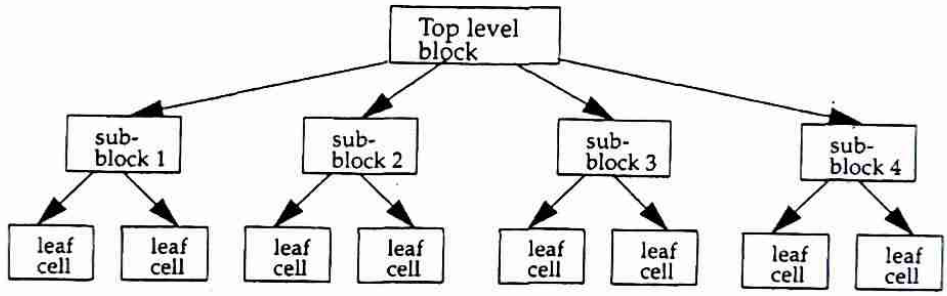
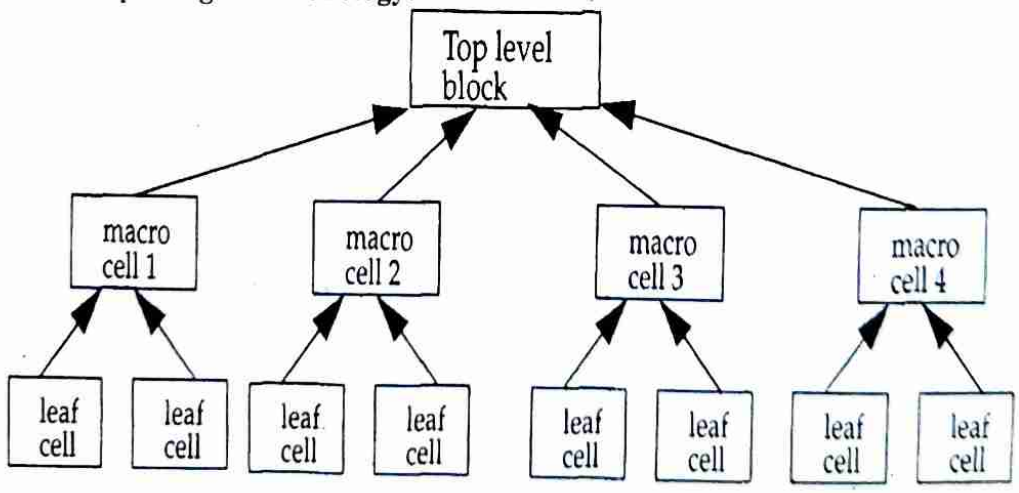
K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

I SESSIONAL TEST scheme 2020 - 21 ODD SEMESTER

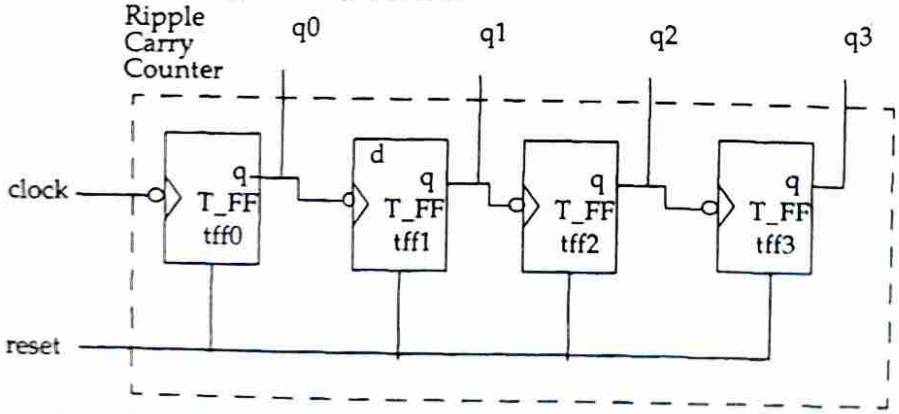
SET – A

Degree	:	B.E	Semester	:	V A & B
Branch	:	Electronics and Communication Engg	Course Code	:	18EC56
Course Title	:	Verilog HDL	Date	:	07/10/20
Duration	:	90 Minutes	Max Marks	:	30

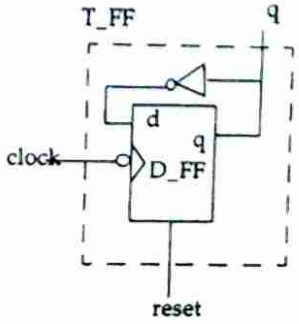
Note: Answer ONE full question from each part.

Q No.	Question	Marks
1(a)	DESIGN MODULE: module half_adder(s,c,a,b); input a,b; output s,c; assign s=a^b; assign c=a&b; endmodule	6
(b)	**Top-Down Design Methodology:**  **Bottom-Up Design Methodology:**  	6

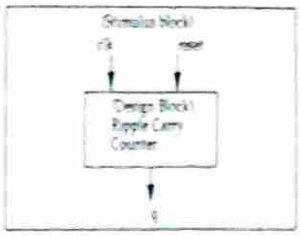
(C)	**Trends in HDLs:** o The speed and complexity of digital circuits has increased rapidly o The most popular trend currently is to design in HDL at an RTL level, because logic synthesis tools can create gate-level net lists from RTL level design o *Formal verification* techniques are also appearing on the horizon o For very high speed and timing-critical circuits like microprocessors, designers often mix gate-level description directly into the RTL description to achieve optimum results o A trend that is emerging for system-level design is a mixed bottom-up methodology	6
2(a)	**Four Levels of Abstraction:** **Behavioral or Algorithmic** – • Highest level of Abstraction • Implemented in terms of the desired design algorithm without concern for the H/W implementation details **Dataflow** – • The designer should be aware of how data flows b/w H/W registers • And how the data is processed in the design **Gate Level** – • Implemented in terms of logic gates and interconnection between these gates • Similar to describing a design in terms of a gate level logic diagram **Switch level** – • Lowest level of abstraction provided by Verilog • Implemented in terms of switches, storage nodes and the interconnection between them	6
(b)	**Instances:** o The process of creating objects from a module template is called *instantiation* o And the objects are called *instances* o Each *instances* must be given a unique name o Example: Ripple Carry Counter	6




```
module ripple_carry_counter(q, clk, reset);
output [3:0] q;
input clk, reset;
T-FF tff0 (q[0], clk, reset); // 4 instances
T-FF tff1 (q[1], q[0], reset);// unique names
T-FF tff2 (q[2] ,q[1] , reset) ;
T-FF tff3(q[3] ,q[2], reset) ;
endmodule

module T_FF(q, clk, reset) ;
output q;
```



input clk, reset;
wire d;
D-FF dff0 (q, d, clk, reset) ; // Instantiate
not n1(d, q); // not gate is a Verilog primitive
endmodule



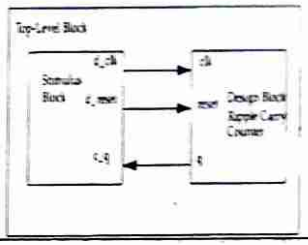
Simulation:

1st Style: Stimulus Block Instantiates the Design Block

- Stimulus block drives the signals in the design block
- Stimulus block becomes the top-level block
- Stimulus manipulates signal clk & reset, and it checks and displays output signal q

2nd Style: Stimulus and Design Blocks Instantiated in a Dummy Top-Level Module

- Stimulus block interacts with the design block only through the interface
- Stimulus module drives the signals d-clk and d-reset, which are connected to the signals clk and reset in the design block
- Stimulus also checks and displays signal c-q, which is connected to the signal q in the design block
- The function of top-level block is simply to instantiate the design and stimulus blocks



(C)

Popularity of Verilog HDL:

- It is a general purpose language: easy to learn & easy to use– similar in syntax to the C
- It allows different levels of abstraction to be mixed in the same model: gate, RTL, or Behavioral design
- Most popular logic synthesis tools support Verilog HDL– making HDL popular among designers
- It allows the widest choice of vendors because all fabrication vendors provide Verilog HDL libraries for post-logic synthesis simulation
- Designers can customize a Verilog HDL simulator to their needs with the Programming Language Interface (PLI).

6

3(a)

i) Comments:

- Comments can be inserted in the code for readability and documentation
- A one-line comment starts with "//". Verilog skips from that point to the end of line
- A multiple-line comment starts with "/*" and ends with "*/"
- Multiple-line comments cannot be nested

ii) Number Specification:

- Sized Numbers
- Unsized Numbers
- X or Z Values
- Negative Numbers

iii) X and Z values:

- Verilog has two symbols for unknown and high impedance values
- These values are very important for modeling real circuits
- An unknown value is denoted by an X
- A high impedance value is denoted by Z
- An X or Z sets four bits for a number in the hexadecimal base, three bits for a number in the octal base, and one bit for a number in the binary base.
- If the most significant bit of a number is of X, or Z, the number is automatically extended to fill the most significant bits, respectively, with 0, X, or Z.

6

	iv) Identifiers and Keywords reg value; // reg is a keyword; value is an identifier input clk; // input is a keyword, clk is an identifier	
(b)	*1. 'define* - The ***'define*** directive is used to define text macros in Verilog - This is similar to the *#define* construct in *C* - The defined constants or text macros are used in the Verilog code by preceding them with a ' (back tick) - The Verilog compiler substitutes the text of the macro wherever it encounters a ' *<macro-name>*. //define a text macro that defines default word size //Used as 'WORD_SIZE in the code 'define WORD_SIZE 32 /*define an alias. A $stop will be substituted wherever 'S appears*/ 'define S $stop. //define a frequently used text string 'define WORD_REG reg [31:0] // you can then define a 32-bit register as 'WORD_REG reg32*/ **2. 'include** - The ' **include** directive allows you to include entire contents of a Verilog source file in another Verilog file during compilation - This works similarly to the *#include* in the *C* programming language - This directive is typically used to include header files, which typically contain global or commonly used definitions /* Include the file header.v, which contains declarations in the main Verilog file design.v */ 'include header.v ... ... *<Verilog code in file design.v>*	6
4(a)	**(i) Arrays:** integer count[0:7]; //an array of 8 count variables reg bool[31:0]; //Array of 32 one-bit boolean register variables time chk_qoint[1:1OO]; //Array of 100 time checkpoint variables reg [4 : 0] port_id[O : 7] ; /*Array of 8 port-ids; each port-id is 5 bits wide*/ integer matrix[4:0][0:255]; // two dimensional array of integers reg [63:0] array_4d [15:0][7:0][7:0][255:0]; /* 4 dimensional array*/ Wire [7:0] w_array2 [5:0]; /*declare an array of 8 bit vector wire*/ Wire w_array1 [7:0] [5:0];/*declare an array of single bit wire*/ **(ii) Memories:** reg mem1bit[0:1023]; //memory mem1bit with 1K 1-bit words reg [7 : 0] membyte [O : 1023]; /* memory membyte with 1K 8-bit words (bytes)*/ membyte[511] //Fetches 1 byte word whose address is 511 **(iii)Parameters:** parameter port_id = 5;//Defines a constant port_id parameter cache_line_width=256;//constant defines width of cache line Parameter signed [15:0] WIDTH; /*Fixed sign and range for parameter WIDTH*/ **(iv)Strings:** reg [8*18:1] string_value; /*Declare a variable that is 18 bytes wide*/ initial string-value = "Hello Verilog World"; /*String can be stored in variable*/	6

$display:

- ***$display*** is the main system task for displaying values of variables or strings or expressions
- This is one of the most useful tasks in Verilog
- ***Usage: $display(p1, p2, p3 ,....., pn); p1, p2, p3, ... pn*** can be quoted strings or variables or expressions
- The format of ***$display*** is very similar to ***printf*** in *C*
- A ***$display*** inserts a newline at the end of the string by default
- *A* ***$display*** without any arguments produces a newline
- Strings can be formatted by using the format specifications

(b) **$monitor:** 6

- Verilog provides a mechanism to monitor a signal when its value changes
- This facility is provided by the $monitor task
- Usage: $monitor(p1 ,p2,p3 ,...., pn);
- The parameters p1, p2, ... , pn can be variables, signal names, or quoted strings
- **A** format similar to the **$display** task is used in the $monitor task
- $monitor continuously monitors the values of the variables or signals specified in the parameter list and displays all parameters in the list whenever the value of any one variable or signal changes
- Unlike $display, $monitor needs to be invoked only once
- Only one monitoring list can be active at a time
- If there is more than one **$monitor** statement in your simulation, the last **$monitor** statement will be the active statement
- The earlier **$monitor** statements will be overridden

Course in charge **Module Coordinator** **HOD/ECE**

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

I SESSIONAL TEST QUESTION PAPER 2020 - 21ODD SEMESTER

Set B

USN

Degree	: B.E	Semester	: V A& B
Branch	: Electronics & Communication Egg.	CourseCode	: 18EC56
Course Title	: VERILOG HDL	Date	: 7-10-2020
Duration	: 90 Minutes	Max Marks	: 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	CO mapping	K-Level
	PART-A			
1(a)	Explain a typical design flow for designing VLSI IC circuits using the block diagram.	6	CO1	K-3 Applying
(b)	Classify a top-down design methodology and a bottom-up design methodology.	6	CO1	K-3 Applying
(c)	Identify the trends in HDLs.	6	CO1	K-3 Applying
	OR			
2(a)	Demonstrate Verilog code in four different abstraction levels for an Inverter gate.	6	CO1	K-3 Applying
(b)	Write a note on i) Instances and ii) Simulation	6	CO1	K-3 Applying
(c)	Discover the factors that have made Verilog HDL popular.	6	CO1	K-3 Applying
	PART-B			
3(a)	Write a note on i) Comments ii) Number Specification iii) Strings and iv) Identifiers and Keywords with suitable examples.	6	CO2	K-3 Applying
(b)	Explore Compiler Directives 'define and 'include with examples.	6	CO2	K-3 Applying
	OR			
4(a)	Explain the following data types with an example in Verilog module: (i) Nets, (ii) Registers, (iii) Integer, (iv) Real.	6	CO2	K-3 Applying
(b)	Explore $display and $monitor tasks with examples.	6	CO2	K-3 Applying

Course in charge

Module Coordinator

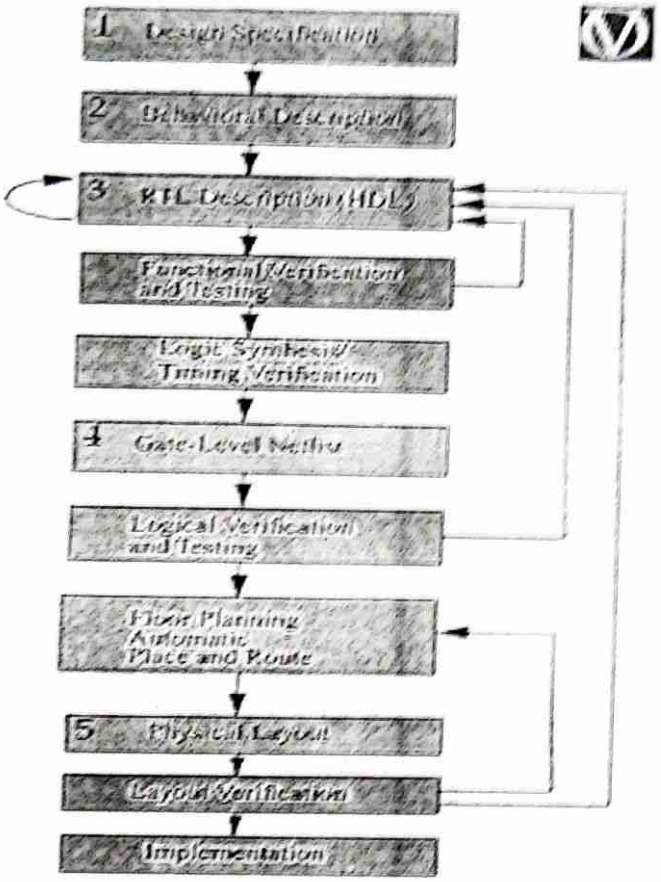
HOD/ECE

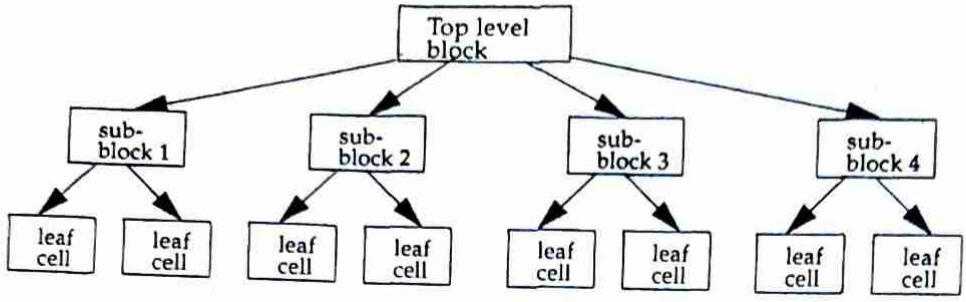
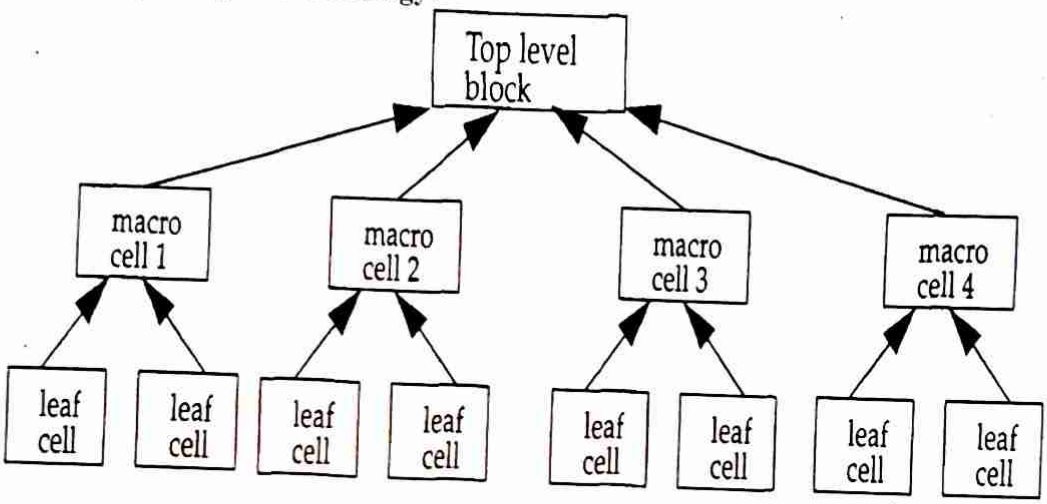
K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
I SESSIONAL TEST scheme 2020 - 21 ODD SEMESTER

SET – B

Degree	:	B.E	Semester	:	V A & B
Branch	:	Electronics and Communication Engg	Course Code	:	18EC56
Course Title	:	Verilog HDL	Date	:	07/10/20
Duration	:	90 Minutes	Max Marks	:	30

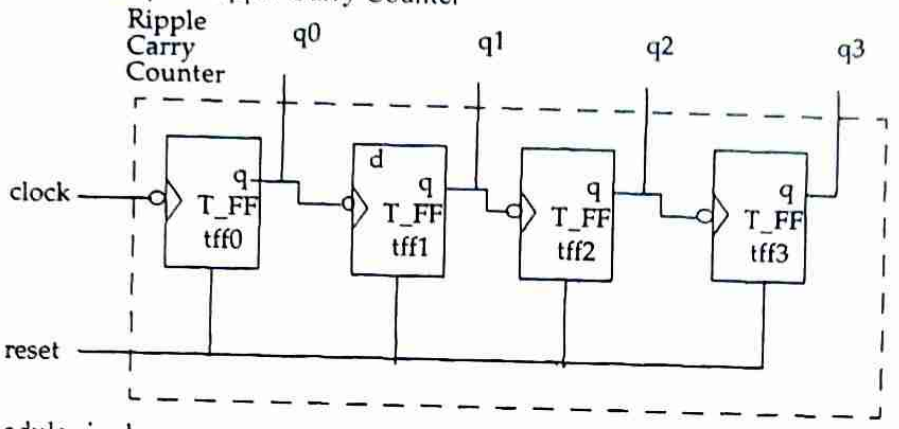
Note: Answer ONE full question from each part.

Q No.	Question	Marks
1(a)	Typical Design Flow:  • Designs can be described at a very abstract level by use of HDLs • Designers can write their RTL description without choosing a specific fabrication technology • If a new technology emerges, designers do not need to redesign their circuit • The logic synthesis tool will optimize the circuit in area and timing for the new technology • Functional verification of the design can be done early in the design cycle • Designing with HDLs is analogous to computer programming	6

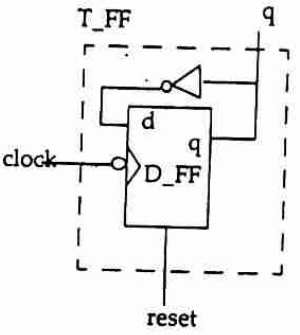
(b)	**Top-Down Design Methodology:**  **Bottom-Up Design Methodology:**  	6
(C)	**Trends in HDLs:** o The speed and complexity of digital circuits has increased rapidly o The most popular trend currently is to design in HDL at an RTL level, because logic synthesis tools can create gate-level net lists from RTL level design o *Formal verification* techniques are also appearing on the horizon o For very high speed and timing-critical circuits like microprocessors, designers often mix gate-level description directly into the RTL description to achieve optimum results o A trend that is emerging for system-level design is a mixed bottom-up methodology	6
2 (a)	**Four Levels of Abstraction:** **Behavioral or Algorithmic** • Highest level of Abstraction • Implemented in terms of the desired design algorithm without concern for the H/W implementation details **Dataflow** • The designer should be aware of how data flows b/w H/W registers • And how the data is processed in the design **Gate Level** • Implemented in terms of logic gates and interconnection between these gates • Similar to describing a design in terms of a gate level logic diagram **Switch level** • Lowest level of abstraction provided by Verilog • Implemented in terms of switches, storage nodes and the interconnection between them	6

Instances:

- The process of creating objects from a module template is called *instantiation*
- And the objects are called *instances*
- Each *instances* must be given a unique name
- Example: Ripple Carry Counter




```
module ripple_carry_counter(q, clk, reset);
output [3:0] q;
input clk, reset;
T-FF tff0 (q[0], clk, reset); // 4 instances
T-FF tff1 (q[1], q[0], reset);// unique names
T-FF tff2 (q[2] ,q[1] , reset) ;
T-FF tff3(q[3] ,q[2], reset) ;
endmodule
```



(b)

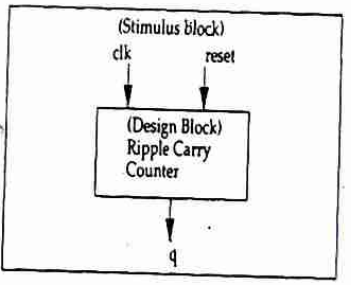
```
module T_FF(q, clk, reset) ;
output q;
input clk, reset;
wire d;
D-FF dff0 (q, d, clk, reset) ; // Instantiate
not n1(d, q); // not gate is a Verilog primitive
endmodule
```

6

Simulation:

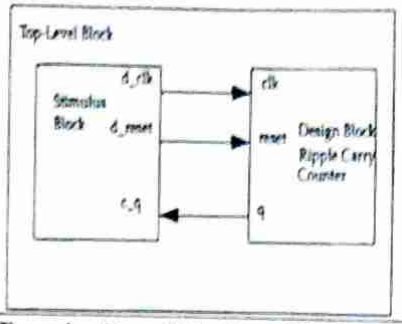
1st Style: Stimulus Block Instantiates the Design Block

- Stimulus block drives the signals in the design block
- Stimulus block becomes the top-level block
- Stimulus manipulates signal clk & reset, and it checks and displays output signal q



2nd Style: Stimulus and Design Blocks Instantiated in a Dummy Top-Level Module

- Stimulus block interacts with the design block only through the interface
- Stimulus module drives the signals d-clk and d-reset, which are connected to the signals clk and reset in the design block
- Stimulus also checks and displays signal c-q, which is connected to the signal q in the design block
- The function of top-level block is simply to instantiate the design and stimulus blocks

		
(C)	**Popularity of Verilog HDL:** o It is a general purpose language: easy to learn & easy to use– similar in syntax to the C o It allows different levels of abstraction to be mixed in the same model: gate, RTL, or Behavioral design o Most popular logic synthesis tools support Verilog HDL– making HDL popular among designers o It allows the widest choice of vendors because all fabrication vendors provide Verilog HDL libraries for post-logic synthesis simulation o Designers can customize a Verilog HDL simulator to their needs with the Programming Language Interface (PLI).	6
3 (a)	**i) Comments:** ➢ Comments can be inserted in the code for readability and documentation ➢ A one-line comment starts with "//". Verilog skips from that point to the end of line ➢ A multiple-line comment starts with "/*" and ends with "*/" ➢ Multiple-line comments cannot be nested **ii) Number Specification:** ➢ Sized Numbers ➢ Unsized Numbers ➢ X or Z Values ➢ Negative Numbers **iii) Strings** i) A string is a sequence of characters that are enclosed by double quotes ii) The restriction on a string is that it must be contained on a single line, that is, without a carriage return iii) It cannot be on multiple lines iv) Strings are treated as a sequence of one-byte ASCII values "Hello Verilog World" // is a string "a / b" // is a string **iv) Identifiers and Keywords** reg value; // reg is a keyword; value is an identifier input clk; // input is a keyword, clk is an identifier	6
(b)	*1. 'define* o The *'define* directive is used to define text macros in Verilog o This is similar to the *#define* construct in *C* o The defined constants or text macros are used in the Verilog code by preceding them with a ' (back tick) o The Verilog compiler substitutes the text of the macro wherever it encounters a '*<macro-name>*. //define a text macro that defines default word size //Used as 'WORD_SIZE in the code 'define WORD_SIZE 32 /*define an alias. A $stop will be substituted wherever 'S appears*/ 'define S $stop.	6

	//define a frequently used text string 'define WORD_REG reg [31:0] // you can then define a 32-bit register as 'WORD_REG reg32*/ **2. 'include** o The ' **include** directive allows you to include entire contents of a Verilog source file in another Verilog file during compilation o This works similarly to the *#include* in the *C* programming language o This directive is typically used to include header files, which typically contain global or commonly used definitions /* Include the file header.v, which contains declarations in the main Verilog file design.v */ 'include header.v ... ... *<Verilog code in file design.v>*	
4 (a)	**Nets:** wire a; // Declare net a for the above circuit wire b,c; // Declare two wires b, c for the above circuit wire d = 1'bO; // Net d is fixed to logic value 0 at declaration. **Registers:** reg reset; /* declare a variable reset that can hold its value*/ initial // this construct will be discussed later begin reset = 1'b1; /*initialize reset to 1 to reset the digital circuit*/ #100 reset = 1'b0; /*after 100 time units reset is deasserted.*/ end **Integer:** integer counter; /*general purpose variable used as a counter*/ initial counter = -1; // A negative one is stored in the counter **Real** real delta; // Define a real variable called delta initial begin . delta = 4e10; // delta is assigned in scientific notation delta = 2.13; // delta is assigned a value 2.13 end integer i; // Define an integer i initial i = delta; // i gets the value 2 (rounded value of 2.13)	6
(b)	**Sdisplay:** ➢ ***$display*** is the main system task for displaying values of variables or strings or expressions ➢ This is one of the most useful tasks in Verilog ➢ ***Usage: $display(p1, p2, p3 ,....., pn); p1, p2, p3, ..., pn*** can be quoted strings or variables or expressions ➢ The format of ***$display*** is very similar to ***printf*** in *C* ➢ A ***$display*** inserts a newline at the end of the string by default ➢ ***A $display*** without any arguments produces a newline ➢ Strings can be formatted by using the format specifications	6

$monitor:

- Verilog provides a mechanism to monitor a signal when its value changes
- This facility is provided by the $monitor task
- Usage: $monitor(p1 ,p2,p3 ....., pn);
- The parameters p1, p2, ... , pn can be variables, signal names, or quoted strings
- A format similar to the **$display** task is used in the $monitor task
- $monitor continuously monitors the values of the variables or signals specified in the parameter list and displays all parameters in the list whenever the value of any one variable or signal changes
- Unlike $display, $monitor needs to be invoked only once
- Only one monitoring list can be active at a time
- If there is more than one **$monitor** statement in your simulation, the last **$monitor** statement will be the active statement
- The earlier **$monitor** statements will be overridden

Course in charge **Module Coordinator** **HOD/ECE**

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
II SESSIONAL TEST QUESTION PAPER 2020 - 21 ODD SEMESTER

SET - A

USN | | | | | | | | | |

Degree	: B.E	**Semester**	: V A& B
Branch	: Electronics & Communication Engg.	**Course Code**	: 18EC56
Course Title	: Verilog HDL	**Date**	: 19/11/2020
Duration	: 90 Minutes	**Max Marks**	: 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	CO mapping	K-Level
	PART-A			
1(a)	Construct a gate level digital circuit for 4-to-1 multiplexer with the help of circuit diagram and verify its functionality with stimulus.	6	CO3	Applying K-3
(b)	Articulate rise, fall and turnoff delays? How they are specified in Verilog.	6	CO3	Applying K-3
(C)	Calculate the output of the following a= 4'b0111, and b= 4'b1001, (i) &b, (ii) a<<2, (iii) {a,b}, (iv) {2{b},a}, (v) a^b, (vi) a|b	6	CO3	Applying K-3
	OR			
2(a)	Construct a Verilog code in dataflow level description for 4-to-1 Multiplexer using (i) Logic Equations and (ii) Conditional Operator	6	CO3	Applying K-3
(b)	Explain regular assignment delay, implicit assignment delay, net declaration delay for continuous assignment statements with examples.	6	CO3	Applying K-3
(C)	Discuss different types of Gates with respect to logic symbols, gate instantiation and truth tables.	6	CO3	Applying K-3
	PART-B			
3(a)	Explain the port connection rules.	6	CO2	Applying K-3
(b)	Construct a Verilog program for 8-to-1 Multiplexer using case statements.	6	CO4	Applying K-3
	OR			
4(a)	Explain the two methods of connecting ports to external signals with examples.	6	CO2	Applying K-3
(b)	Construct a Verilog program for 8-to-3 Encoder without priority.	6	CO4	Applying K-3

Course In charge **Module Coordinator** **HOD-ECE**

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

II SESSIONAL TEST scheme 2020 - 21 ODD SEMESTER

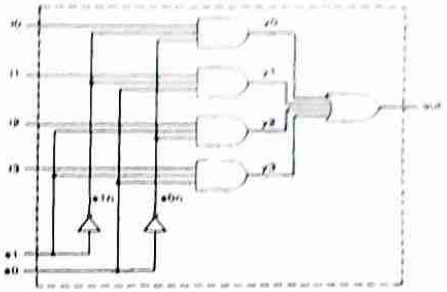
SET - A

Degree	:	**B.E**	**Semester**	:	**V A & B**
Branch	:	**Electronics and Communication Engg**	**Course Code**	:	**18EC56**
Course Title	:	**Verilog HDL**	**Date**	:	**19/11/2020**
Duration	:	**90 Minutes**	**Max Marks**	:	**30**

Note: Answer ONE full question from each part.

<table>
<tr><th>Q No.</th><th>Question</th><th>Marks</th></tr>
<tr><td>1(a)</td><td>
module mux4-to-1 (out, i0, i1, i2, i3, s1, s0);

output out;
input i0, i1, i2, i3;
input s1, S0;
wire s1n, s0n;
wire y0, y1, y2, y3;
not (s1n, s1) ;
not (s0n, s0);
and (y0, i0, s1n, s0n);
and (y1, i1, s1n, s0);
and (y2, i2, s1, s0n);
and (y3, i3, s1, s0);
or (out, y0, y1, y2, y3);
endmodule



// Define the stimulus module (no ports)
module stimulus;

// Declare variables to be connected
// to inputs
reg IN0, IN1, IN2, IN3;
reg S1, S0;

// Declare output wire
wire OUTPUT;

// Instantiate the multiplexer
mux4_to_1 mymux(OUTPUT, IN0, IN1, IN2, IN3, S1, S0);

// Define the stimulus module (no ports)

// Stimulate the inputs
initial
begin
// set input lines
IN0 = 1; IN1 = 0; IN2 = 1; IN3 = 0;
#1 $display("IN0= %b, IN1= %b, IN2= %b, IN3= %b\n",IN0,IN1,IN2,IN3);

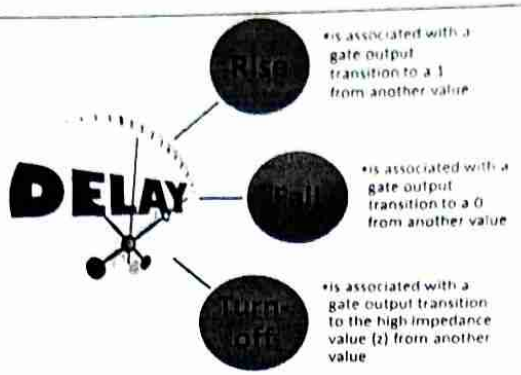
// choose IN0
S1 = 0; S0 = 0;
#1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUTPUT);

// choose IN1
S1 = 0; S0 = 1;
#1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUTPUT);

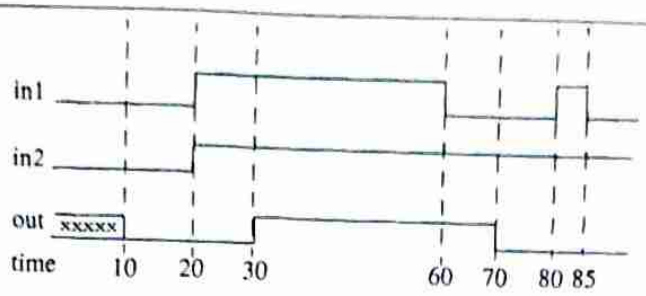
// choose IN2
S1 = 1; S0 = 0;
#1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUTPUT);

// choose IN3
S1 = 1; S0 = 1;
#1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUTPUT);
end

endmodule
</td><td>6</td></tr>
<tr><td>(b)</td><td>There are three types of delays from the inputs to the output of a primitive gate</td><td>6</td></tr>
</table>



(C)	(i) &b= 0; (ii) A<<2= 1100; (iii) {a,b}= 8'b01111001, (iv) {2{b},a}= 12'b100110010111, (v) a^b= 1110; (vi) a | b= 1111	6
2(a)	**Method 1: logic equation** `// Module 4-to-1 multiplexer using data flow. logic equation` `// Compare to gate-level model` `module mux4_to_1 (out, i0, i1, i2, i3, s1, s0);` `// Port declarations from the I/O diagram` `output out;` `input i0, i1, i2, i3;` `input s1, s0;` `//Logic equation for out` `assign out = (~s1 & ~s0 & i0)|` `(~s1 & s0 & i1) |` `(s1 & ~s0 & i2) |` `(s1 & s0 & i3) ;` `endmodule` **Method 2: conditional operator:** `// Module 4-to-1 multiplexer using data flow. Conditional operator.` `// Compare to gate-level model` `module multiplexer4_to_1 (out, i0, i1, i2, i3, s1, s0);` `// Port declarations from the I/O diagram` `output out;` `input i0, i1, i2, i3;` `input s1, s0;` `// Use nested conditional operator` `assign out = s1 ? ( s0 ? i3 : i2) : (s0 ? i1 : i0) ;` `endmodule`	6
(b)	DELAYS 1. Regular Assignment delay 2. Implicit Continuous Assignment Delay 3. Net Declaration Delay. assign $10 out = in1 & in2; // Delay in a continuous assign	6



//implicit continuous assignment delay

```
wire #10 out = in1 & in2;
//same as
wire out;
assign #10 out = in1 & in2;
```

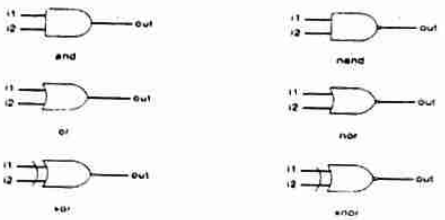
//Net Delays

```
wire # 10 out;
assign out = in1 & in2;
//The above statement has the same effect as the following
wire out;
assign #10 out = in1 & in2;
```

(C)

And/or Gates:

and	or	xor
nand	nor	xnor



or	i1 0	i1 1	i1 x	i1 z
i2 0	0	1	x	x
i2 1	1	1	1	1
i2 x	x	1	x	x
i2 z	x	1	x	x

nor	i1 0	i1 1	i1 x	i1 z
i2 0	1	0	x	x
i2 1	0	0	0	0
i2 x	x	0	x	x
i2 z	x	0	x	x

xor	i1 0	i1 1	i1 x	i1 z
i2 0	0	1	x	x
i2 1	1	0	x	x
i2 x	x	x	x	x
i2 z	x	x	x	x

xnor	i1 0	i1 1	i1 x	i1 z
i2 0	1	0	x	x
i2 1	0	1	x	x
i2 x	x	x	x	x
i2 z	x	x	x	x

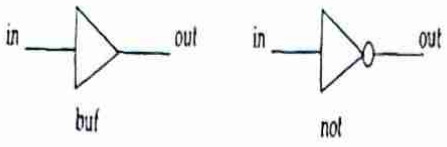
and	i1 0	i1 1	i1 x	i1 z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

nand	i1 0	i1 1	i1 x	i1 z
i2 0	1	1	1	1
i2 1	1	0	x	x
i2 x	1	x	x	x
i2 z	1	x	x	x

```
wire OUT, IN1, IN2;

// basic gate instantiations.
and a1(OUT, IN1, IN2);
nand na1(OUT, IN1, IN2);
or or1(OUT, IN1, IN2);
nor nor1(OUT, IN1, IN2);
xor x1(OUT, IN1, IN2);
xnor nx1(OUT, IN1, IN2);

// More than two inputs; 3 input nand gate
nand na1_3inp(OUT, IN1, IN2, IN3);

// gate instantiation without instance name
and (OUT, IN1, IN2); // legal gate instantiation
```

6

Buf/Not Gates:



buf	in	out
	0	0
	1	1
	x	x
	z	x

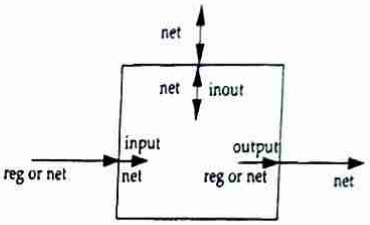
not	in	out
	0	1
	1	0
	x	x
	z	x

```
// basic gate instantiations.
buf b1(OUT1, IN);
not n1(OUT1, IN);

// More than two outputs
buf b1_2out(OUT1, OUT2, IN);

// gate instantiation without instance name
not (OUT1, IN); // legal gate instantiation
```

3(a)

Inputs

- Internally, input ports must always be of the type *net*.
- Externally, the inputs can be connected to a variable which is a reg or a *net*.



Outputs

- Internally, outputs ports can be of the type reg or *net*
- Externally, outputs must always be connected to a *net*
- They cannot be connected to a reg

Inouts

- Internally, inout ports must always be of the type *net*
- Externally, inout ports must always be connected to a *net*

Width matching

- It is legal to connect internal and external items of different sizes when making inter-module port connections
- However, a warning is typically issued that the widths do not match.

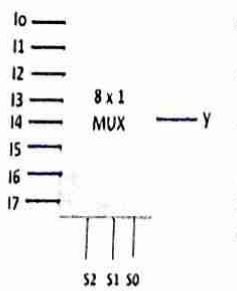
Unconnected ports

- Verilog allows ports to remain unconnected
- For example, certain output ports might be simply for debugging, and you might not be interested in connecting them to the external signals
- You can let a port remain unconnected by instantiating a module as shown below

6

(b)

```
module mux_case(I0, I1, I2, I3, I4, I5, I6, I7, sel, y);
input I0, I1, I2, I3, I4, I5, I6, I7;
input [2:0]sel;
output y;
reg y;
always @(sel)
begin
case(sel)
3'b000:y=I0;
3'b001:y=I1;
3'b010:y=I2;
3'b011:y=I3;
3'b100:y=I4;
3'b101:y=I5;
3'b110:y=I6;
3'b111:y=I7;
default: y=1'b0;
```



S2	S1	S0	Y
0	0	0	I0
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	0	0	I4
1	0	1	I5
1	1	0	I6
1	1	1	I7

6

endcase
end
endmodule

4(a) — 6

a) Connecting by ordered list

```
module Top;
//Declare connection variables
reg [3:0]A,B;
reg C_IN;
wire [3:0] SUM;
wire C_OUT;
//Instantiate fulladd4, call it fa_ordered.
//Signals are connected to ports in order (by position)
fulladd4 fa_ordered(SUM, C_OUT, A, B, C_IN);
...
<stimulus>
...
Endmodule
```

```
module fulladd4 (sum, c_out, a, b, c_in) ;
output [3 : 0] sum;
output c_out;
input [3:0] a, b;
input c_in;
...
<module internals>
...
endmodule
```

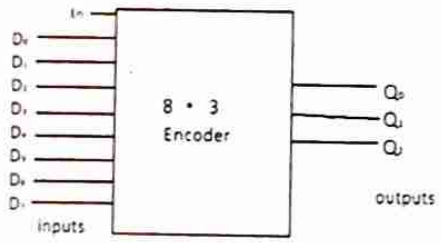
b) Connecting ports by name

```
/* Instantiate module fa_byname and connect signals to ports by name*/
fulladd4 fa_byname(.c_out(C_OUT), .sum(SUM), .b(B), .c_in(C_IN), .a(A) .);

/*Instantiate module fa_byname and connect signals to ports by name*/
fulladd4 fa_byname( . sum(SUM) , .b(B) , .c_in (C_IN) , .a (A) , ) ;
```

(b) — 6

8-to-3 Encoder without priority:

```
module encoder(D, Q);
  input [7:0] D;
  output [2:0] Q;
      reg [2:0] Q;
      always @ (D)
      begin
      case(D)
      8'b00000001:Q<=3'b000;
      8'b00000010:Q<=3'b001;
      8'b00000100:Q<=3'b010;
      8'b00001000:Q<=3'b011;
      8'b00010000:Q<=3'b100;
      8'b00100000:Q<=3'b101;
      8'b01000000:Q<=3'b110;
      8'b10000000:Q<=3'b111;
      default: Q<=3'bXXX;
      endcase
      end
endmodule
```



Course in charge | Module Coordinator | HOD-ECE

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

II SESSIONAL TEST QUESTION PAPER 2020 - 21 ODD SEMESTER

SET - B

USN | | | | | | | | | | |

Degree	: B.E	Semester	: V A& B
Branch	: Electronics & Communication Engg.	Course Code	: 18EC56
Course Title	: Verilog HDL	Date	: 19/11/2020
Duration	: 90 Minutes	Max Marks	: 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	CO mapping	K-Level
	PART-A			
1(a)	Construct a Verilog code and stimulus in gate level description for 4-bit full Adder.	6	CO3	Applying K-3
(b)	Articulate rise, fall and turnoff delays? How they are specified in Verilog.	6	CO3	Applying K-3
(C)	Calculate the output of the following: a= 4'b0111, and b= 4'b1001, (i) &b, (ii) a<<2, (iii) {a,b}, (iv) {2{b},a}, (v) a^b, (vi) a|b	6	CO3	Applying K-3
	OR			
2(a)	Construct a Verilog code in dataflow level description for 4-to-1 Multiplexer using (i) Logic Equations and (ii) Conditional Operator.	6	CO3	Applying K-3
(b)	Explain regular assignment delay, implicit assignment delay, net declaration delay for continuous assignment statements with examples.	6	CO3	Applying K-3
(C)	Discuss different types of Gates with respect to logic symbols, gate instantiation and truth tables.	6	CO3	Applying K-3
	PART-B			
3(a)	Illustrate the components of Verilog module with the help of neat diagram.	6	CO2	Applying K-3
(b)	Construct a Verilog program for 8-to-1 Multiplexer using case statements.	6	CO4	Applying K-3
	OR			
4(a)	Explain the two methods of connecting ports to external signals with examples.	6	CO2	Applying K-3
(b)	Construct a Verilog program for 8-to-3 Encoder with priority.	6	CO4	Applying K-3

Course In charge | Module Coordinator | HOD-ECE

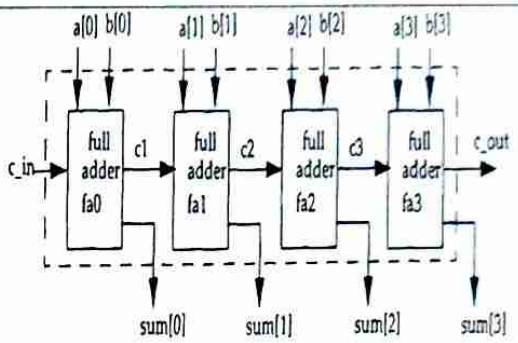
K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

II SESSIONAL TEST scheme 2020 - 21 ODD SEMESTER

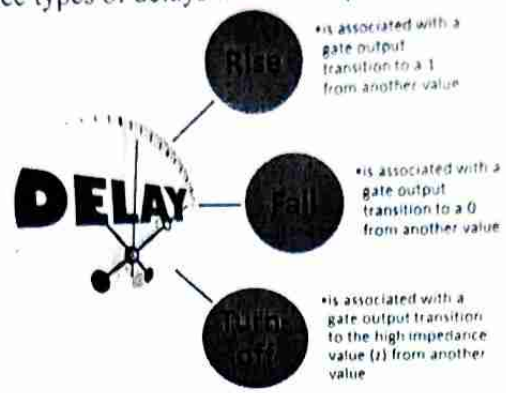
SET – B

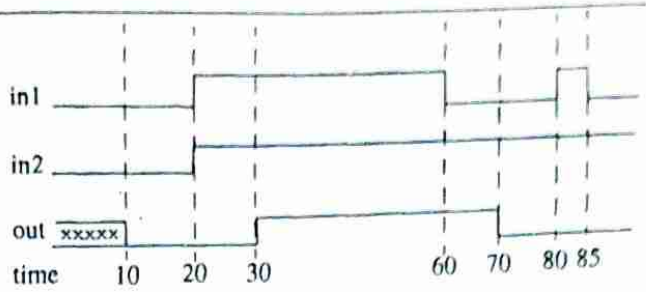
Degree	: B.E	Semester	: V A & B
Branch	: Electronics and Communication Engg	Course Code	: 18EC56
Course Title	: Verilog HDL	Date	: 19/11/2020
Duration	: 90 Minutes	Max Marks	: 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks
1(a)		6

```
// Define a 4-bit full adder
module fulladd4(sum, c_out, a, b, c_in);

// I/O port declarations
output [3:0] sum;
output c_out;
input[3:0] a, b;
input c_in;

// Internal nets
wire c1, c2, c3;

// Instantiate four 1-bit full adders.
fulladd fa0(sum[0], c1, a[0], b[0], c_in);
fulladd fa1(sum[1], c2, a[1], b[1], c1);
fulladd fa2(sum[2], c3, a[2], b[2], c2);
fulladd fa3(sum[3], c_out, a[3], b[3], c3);

endmodule
// Define the stimulus (top level module)
module stimulus;

// Set up variables
reg [3:0] A, B;
reg C_IN;
wire [3:0] SUM;
wire C_OUT;

// Instantiate the 4-bit full adder. call it FA1_4
fulladd4 FA1_4(SUM, C_OUT, A, B, C_IN);


// Setup the monitoring for the signal values
initial
begin
  $monitor($time," A= %b, B=%b, C_IN= %b, --- C_OUT= %b, SUM= %b\n",
                          A, B, C_IN, C_OUT, SUM);
end

// Stimulate inputs
initial
begin
  A = 4'd0; B = 4'd0; C_IN = 1'b0;

  #5 A = 4'd3; B = 4'd4;

  #5 A = 4'd2; B = 4'd5;

  #5 A = 4'd9; B = 4'd9;

  #5 A = 4'd10; B = 4'd15;

  #5 A = 4'd10; B = 4'd5; C_IN = 1'b1;
end

endmodule
```

(b)	There are three types of delays from the inputs to the output of a primitive gate  	6
(C)	(i) &b= 0; (ii) A<<2= 1100; (iii) {a,b}= 8'b01111001, (iv) {2{b},a}= 12'b100110010111, (v) a^b= 1110; (vi) a | b= 1111	6
2(a)	**Method 1: logic equation** `// Module 4-to-1 multiplexer using data flow. logic equation` `// Compare to gate-level model` `module mux4_to_1 (out, i0, i1, i2, i3, s1, s0);` `// Port declarations from the I/O diagram` `output out;` `input i0, i1, i2, i3;` `input s1, s0;` `//Logic equation for out` `assign out = (~s1 & ~s0 & i0)|` `(~s1 & s0 & i1) |` `(s1 & ~s0 & i2) |` `(s1 & s0 & i3) ;` `endmodule` **Method 2: conditional operator:** `// Module 4-to-1 multiplexer using data flow. Conditional operator.` `// Compare to gate-level model` `module multiplexer4_to_1 (out, i0, i1, i2, i3, s1, s0);` `// Port declarations from the I/O diagram` `output out;` `input i0, i1, i2, i3;` `input s1, s0;` `// Use nested conditional operator` `assign out = s1 ? ( s0 ? i3 : i2) : (s0 ? i1 : i0) ;` `endmodule`	6
(b)	DELAYS 1. Regular Assignment delay 2. Implicit Continuous Assignment Delay 3. Net Declaration Delay. assign #10 out = in1 & in2; // Delay in a continuous assign	6



//implicit continuous assignment delay

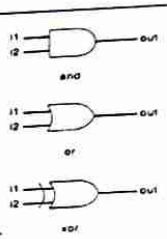
```
wire #10 out = in1 & in2;
//same as
wire out;
assign #10 out = in1 & in2;
```

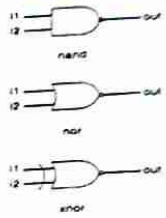
//Net Delays

```
wire # 10 out;
assign out = in1 & in2;
//The above statement has the same effect as the following
wire out;
assign #10 out = in1 & in2;
```

And/or Gates:

and	or	xor
nand	nor	xnor





or	0	1	x	z
0	0	1	x	x
1	1	1	1	1
x	x	1	x	x
z	x	1	x	x

nor	0	1	x	z
0	1	0	x	x
1	0	0	0	0
x	x	0	x	x
z	x	0	x	x

and	0	1	x	z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

nand	0	1	x	z
0	1	1	1	1
1	1	0	x	x
x	1	x	x	x
z	1	x	x	x

xor	0	1	x	z
0	0	1	x	x
1	1	0	x	x
x	x	x	x	x
z	x	x	x	x

xnor	0	1	x	z
0	1	0	x	x
1	0	1	x	x
x	x	x	x	x
z	x	x	x	x

(C)

```
wire OUT, IN1, IN2;

// basic gate instantiations.
and a1(OUT, IN1, IN2);
nand na1(OUT, IN1, IN2);
or or1(OUT, IN1, IN2);
nor nor1(OUT, IN1, IN2);
xor x1(OUT, IN1, IN2);
xnor nx1(OUT, IN1, IN2);

// More than two inputs; 3 input nand gate
nand na1_3inp(OUT, IN1, IN2, IN3);

// gate instantiation without instance name
and (OUT, IN1, IN2); // legal gate instantiation
```

Buf/Not Gates:

6

buf	in	out
	0	0
	1	1
	x	x
	z	x

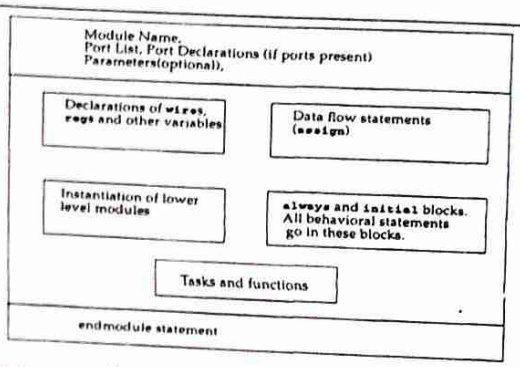
not	in	out
	0	1
	1	0
	x	x
	z	x

```
// basic gate instantiations.
buf b1(OUT1, IN);
not n1(OUT1, IN);

// More than two outputs
buf b1_2out(OUT1, OUT2, IN);

// gate instantiation without instance name
not (OUT1, IN); // legal gate instantiation
```

3(a) — 6

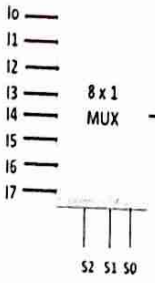
Components of a Verilog Module:

- The five components within a module are - variable declarations, dataflow statements, instantiation of lower modules, behavioral blocks, and tasks or functions
- These components can be in any order and at any place in the module definition
- The endmodule statement must always come last in a module definition
- All components except module, module name, and endmodule are optional and can be mixed and matched as per design needs
- Verilog allows multiple modules to be defined in a single file
- The modules can be defined in any order in the file.



(b) — 6

```
module mux_case(I0, I1, I2, I3, I4, I5, I6, I7, sel, y);
input I0, I1, I2, I3, I4, I5, I6, I7;
input [2:0]sel;
output y;
reg y;
always @(sel)
begin
case(sel)
3'b000:y=I0;
3'b001:y=I1;
3'b010:y=I2;
3'b011:y=I3;
3'b100:y=I4;
3'b101:y=I5;
3'b110:y=I6;
3'b111:y=I7;
default: y=1'b0;
endcase
end
endmodule
```



S2	S1	S0	Y
0	0	0	I0
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	0	0	I4
1	0	1	I5
1	1	0	I6
1	1	1	I7

4(a) — 6

a) Connecting by ordered list

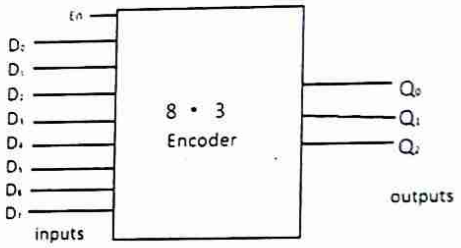
```
module Top;
//Declare connection variables
reg [3:0]A,B;
reg C_IN;
wire [3:0] SUM;
```

```
wire C_OUT;
//Instantiate fulladd4, call it fa_ordered.
//Signals are connected to ports in order (by position)
fulladd4 fa_ordered(SUM, C_OUT, A, B, C_IN);
...
<stimulus>
...
Endmodule


module fulladd4 (sum, c_out, a, b, c_in) ;
output [3 : 0] sum;
output c_out;
input [3:0] a, b;
input c_in;
...
<module internals>
...
endmodule
```

b) Connecting ports by name

```
/* Instantiate module fa_byname and connect signals to ports by name*/
fulladd4 fa_byname(.c_out(C_OUT), .sum(SUM), .b(B), .c_in(C_IN), .a(A) ,);

/*Instantiate module fa_byname and connect signals to ports by name*/
fulladd4 fa_byname( . sum(SUM) , .b(B) , .c_in (C_IN) , .a (A) , ) ;
```

(b) **8-to-3 Encoder with priority:**




```
module encoder_with (y, a);
input [7:0] y;
output [2:0] a;
reg [2:0] a;
always @(y)
begin
case(y)
8'b00000001:a<=3'd0;
8'b0000001X:a<=3'd1;
8'b000001XX:a<=3'd2;
8'b00001XXX:a<=3'd3;
8'b0001XXXX:a<=3'd4;
8'b001XXXXX:a<=3'd5;
8'b01XXXXXX:a<=3'd6;
8'b1XXXXXXX:a<=3'd7;
default: a<=3'dX;
endcase
end
endmodule
```

6

Course in charge | Module Coordinator | HOD-ECE

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
III SESSIONAL TEST QUESTION PAPER 2020 – 21 ODD SEMESTER

SET – A

USN

Degree : B.E
Branch : Electronics & Communication Engg.
Course Title : Verilog HDL
Duration : 90 Minutes

Semester : V A& B
Course Code : 18EC56
Date : 9/1/2021
Max Marks : 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	CO mapping	K-Level
	PART-A			
1(a)	Discuss the logic synthesis flow from RTL to Gates with a neat diagram.	6	CO5	Applying K-3
(b)	Explain conditional compilation and conditional execution with examples.	6	CO5	Applying K-3
(C)	Discuss the two ways to override parameter values with examples.	6	CO5	Applying K-3
	OR			
2(a)	Discuss at least six limitations of design process and how automation of logic synthesis tool has addressed these limitations.	6	CO5	Applying K-3
(b)	Interpret the following statements into Verilog constructs: (i) assign out = (a & b) | c; (ii) assign {c_out, sum} = a+b+c_in; (iii) assign out= (s)? i1 : i0;	6	CO5	Applying K-3
(C)	What is logic synthesis? Illustrate the designer's mind as the logic synthesis tool'	6	CO5	Applying K-3
	PART-B			
3(a)	Explain sequential blocks and parallel blocks with examples.	6	CO4	Applying K-3
(b)	Construct a 4-to-1 multiplexer with the behavioral case statement; test its functionality by writing a test bench.	6	CO4	Applying K-3
	OR			
4(a)	Chart out the differences between tasks and functions.	6	CO4	Applying K-3
(b)	Explain three types of delay control for procedural assignments.	6	CO4	Applying K-3

Course in charge

Module Coordinator

HOD/ECE

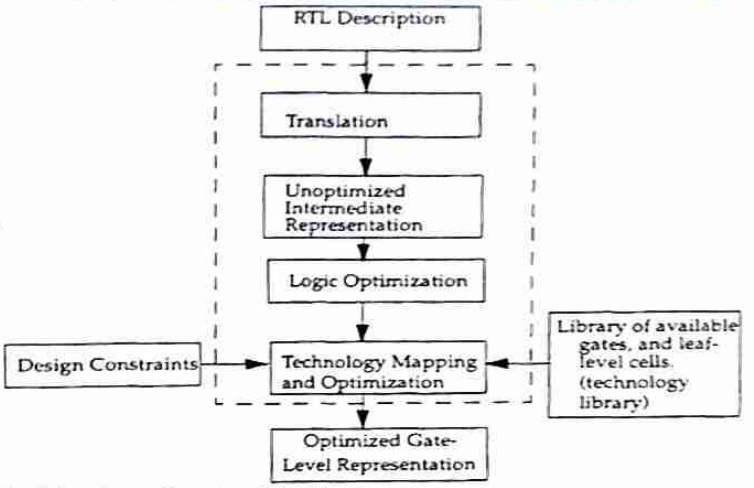
K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

III SESSIONAL TEST scheme 2020 - 21 ODD SEMESTER

SET – A

Degree	: B.E	Semester	: V A & B
Branch	: Electronics and Communication Engg	Course Code	: 18EC56
Course Title	: Verilog HDL	Date	: 9/1/2021
Duration	: 90 Minutes	Max Marks	: 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks
1(a)	 Explaination of each of the blocks.	6
(b)	**Conditional Compilation:** Conditional compilation can be accomplished by using compiler directives 'ifdef, 'else, and 'endif. `//Conditional Compilation` `//Example 1` `'ifdef TEST //compile module test only if text macro TEST is defined` `module test;` `…` `…` `endmodule` `'else //compile the module stimulus as default` `module stimulus;` `…` `…` `endmodule` `'endif //completion of 'ifdef statement` `//Example 2` `module top;` `bus_master b1(); //instantiate module unconditionally` `'ifdef ADD_B2` `bus_master b2(); //b2 is instantiated conditionally if text macro //ADD_B2 is defined` `'endif` `endmodule` **Conditional Execution:**	6

```
//Conditional execution
module test;
…
…
initial
begin
    if($test$plusargs("DISPLAY_VAR"))
        $display("Display = %b ", {a,b,c} ); //display only if flag is set
    else
        $display("No Display"); //otherwise no display
end
```

defparam Statement:

Example 9-2 *Defparam Statement*

```
//Define a module hello_world
module hello_world;
parameter id_num = 0; //define a module identification number = 0

initial //display the module identification number
        $display("Displaying hello_world id number = %d", id_num);
endmodule


//define top-level module
module top;
//change parameter values in the instantiated modules
//Use defparam statement
defparam w1.id_num = 1, w2.id_num = 2;

//instantiate two hello_world modules
hello_world w1();
hello_world w2();

endmodule
```

(C)

Module-Instance Parameter Values:

Example 9-3 *Module Instance Parameter Values*

```
//define module with delays
module bus_master;
parameter delay1 = 2;
parameter delay2 = 3;
parameter delay3 = 7;
…
<module internals>
…
endmodule

//top-level module; instantiates two bus_master modules
module top;

//Instantiate the modules with new delay values
bus_master #(4, 5, 6) b1(); //b1: delay1 = 4, delay2 = 5, delay3 = 6
bus_master #(9,4) b2(); //b2: delay1 = 9, .delay2 = 4, delay3 = 7(default)

endmodule
```

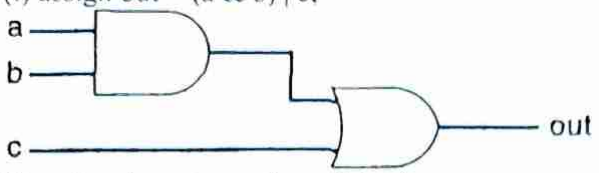
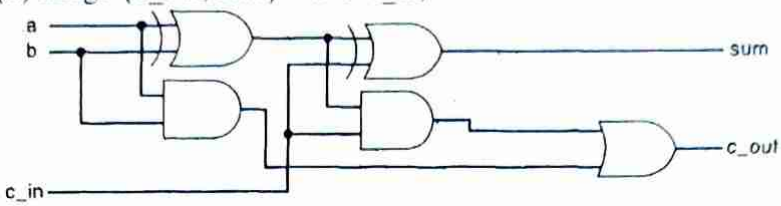
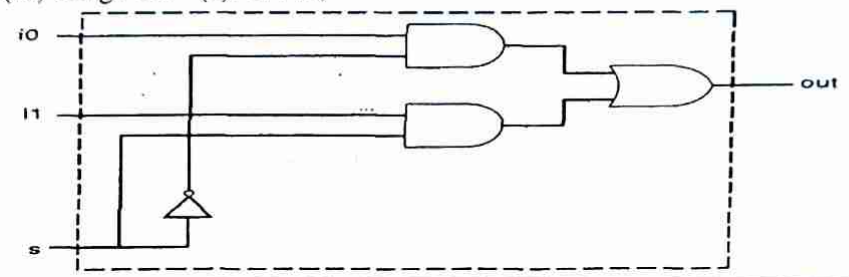
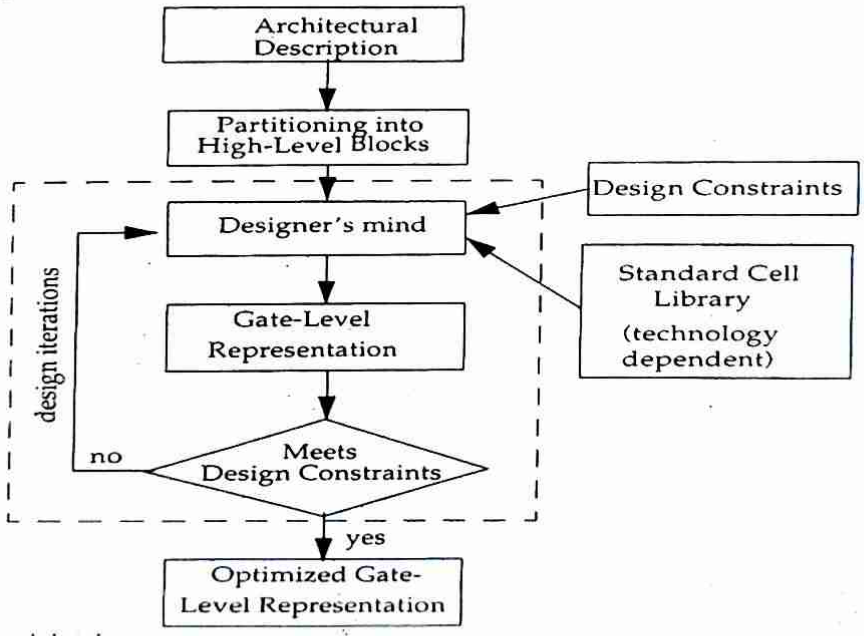
6

2(a)

Limitations: human error, never be sure that the design constraints were going to be met, turnaround time, What-if scenarios were hard to verify, little consistency in design styles, a bug, Timing, area, and power dissipation in library cells, Design reuse was not possible.

Automated logic synthesis: less prone to human error, ensure that all constraints have been met, Conversion from high-level design to gates is fast, Turnaround time for redesign of blocks is shorter, ***What-if*** scenarios are easy to verify, Logic synthesis tools optimize the design as a whole, the bug is eliminated, Logic synthesis tools allow ***technology-independent***

6

	design, Design reuse is possible.	
(b)	(i) assign out = (a & b) | c;  (ii) assign {c_out, sum} = a+b+c_in;  (iii) assign out= (s)? i1 : i0;  	6
(C)	Logic synthesis is the process of converting a high-level description of the design into an optimized gate-level representation, given a standard cell library and certain design constraints.  Explaination.	6
3(a)	**Sequential blocks:**	6

```
//Illustration 1: Sequential block without delay
reg x, y;
reg [1:0] z, w;

initial
begin
        x = 1'b0;
        y = 1'b1;
        z = {x, y};
        w = {y, x};
end

//Illustration 2: Sequential blocks with delay.
reg x, y;
reg [1:0] z, w;

initial
begin
        x = 1'b0; //completes at simulation time 0
        #5 y = 1'b1; //completes at simulation time 5
        #10 z = {x, y}; //completes at simulation time 15
        #20 w = {y, x}; //completes at simulation time 35
end
```

Parallel blocks:

```
//Example 1: Parallel blocks with delay.
reg x, y;
reg [1:0] z, w;

initial
fork
        x = 1'b0; //completes at simulation time 0
        #5 y = 1'b1; //completes at simulation time 5
        #10 z = {x, y}; //completes at simulation time 10
        #20 w = {y, x}; //completes at simulation time 20
join
```

(b)

```
// Port declarations from the I/O diagram
output out;
input i0, i1, i2, i3;
input s1, s0;
//output declared as register
reg out;

//recompute the signal out if any input signal changes.
//All input signals that cause a recomputation of out to
//occur must go into the always @(...)  sensitivity list.
always @(s1 or s0 or i0 or i1 or i2 or i3)
begin
  case ({s1, s0})
  2'b00: out = i0;
  2'b01: out = i1;
  2'b10: out = i2;
  2'b11: out = i3;
  default: out = 1'bx;
  endcase
end

endmodule
```

6

4(a) — 6

Functions	Tasks
A function can enable another function but not another task.	A task can enable other tasks and functions.
Functions always execute in 0 simulation time.	Tasks may execute in non-zero simulation time.
Functions must not contain any delay, event, or timing control statements.	Tasks may contain delay, event, or timing control statements.
Functions must have at least one **input** argument. They can have more than one **input**.	Tasks may have zero or more arguments of type **input, output or inout**.
Functions always return a single value. They cannot have **output** or **inout** arguments.	Tasks do not return with a value but can pass multiple values through **output** and **inout** arguments.

(b) — 6

Delay-Based Timing Control

```
<delay>
    ::= #<NUMBER>
    |= #<identifier>
    |= #(<mintypmax_expression> <,<mintypmax_expression>>*)
```

Regular delay control

```
//define parameters
parameter latency = 20;
parameter delta = 2;
//define register variables
reg x, y, z, p, q;

initial
begin
        x = 0; // no delay control
       #10 y = 1; // delay control with a number. Delay execution of
                                  // y = 1 by 10 units

      #latency z = 0; //Delay control with identifier.Delay of 20 units
        #(latency + delta) p = 1; // Delay control with expression

       #y x = x + 1; // Delay control with identifier. Take value of y.

        #(4:5:6) q = 0; // Minimum, typical and maximum delay values.
                        //Discussed in gate-level modeling chapter.
end
```

Intra-assignment delay control:

```
//define register variables
reg x, y, z;

//intra assignment delays
initial
begin
        x = 0; z = 0;
        y = #5 x + z; //Take value of x and z at the time=0, evaluate
                      //x + z and then wait 5 time units to assign value
                     //to y.

end

//Equivalent method with temporary variables and regular delay control
initial
begin
        x = 0; z = 0;
        temp_xz = x + z;
       #5 y = temp_xz; //Take value of x + z at the current time and
              //store it in a temporary variable. Even though x and z
                 //might change between 0 and 5,
                 //the value assigned to y at time 5 is unaffected.
end
```

Course in charge

Module Coordinator

HOD/ECE

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
III SESSIONAL TEST QUESTION PAPER 2020 - 21 ODD SEMESTER

SET – B

USN | | | | | | | | | | |

Degree	:	**B.E**	**Semester**	:	**V A& B**
Branch	:	**Electronics & Communication Engg.**	**Course Code**	:	**18EC56**
Course Title	:	**Verilog HDL**	**Date**	:	**9/1/2021**
Duration	:	**90 Minutes**	**Max Marks**	:	**30**

Note: Answer ONE full question from each part.

Q No.	Question	Marks	CO mapping	K-Level
	PART-A			
1(a)	Discuss the following Procedural Continuous Assignments: (i) assign and deassign, (ii) force and release	6	CO5	Applying K-3
(b)	Illustrate the automated logic synthesis tool with a neat flow diagram.	6	CO5	Applying K-3
(C)	Justify how automated logic synthesis tools have addressed the problems faced in manual synthesis.	6	CO5	Applying K-3
	OR			
2(a)	Explain any three useful system tasks.	6	CO5	Applying K-3
(b)	Discuss the limitations of manual synthesis.	6	CO5	Applying K-3
(C)	Discuss the following: (i) Technology Mapping and Optimization, (ii) Technology Library, (iii) Design Constraints.	6	CO5	Applying K-3
	PART-B			
3(a)	Illustrate Tasks and Functions with respect to: (i) Usage condition, (ii) Declaration and Invocation, and (iii) Examples	6	CO4	Applying K-3
(b)	Construct an 8-to-1 multiplexer with the behavioral case statement; test its functionality by writing a test bench.	6	CO4	Applying K-3
	OR			
4(a)	Illustrate with example four types of looping statements in Verilog.	6	CO4	Applying K-3
(b)	Discuss special features of block statements.	6	CO4	Applying K-3

Course in charge **Module Coordinator** **HOD/ECE**

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

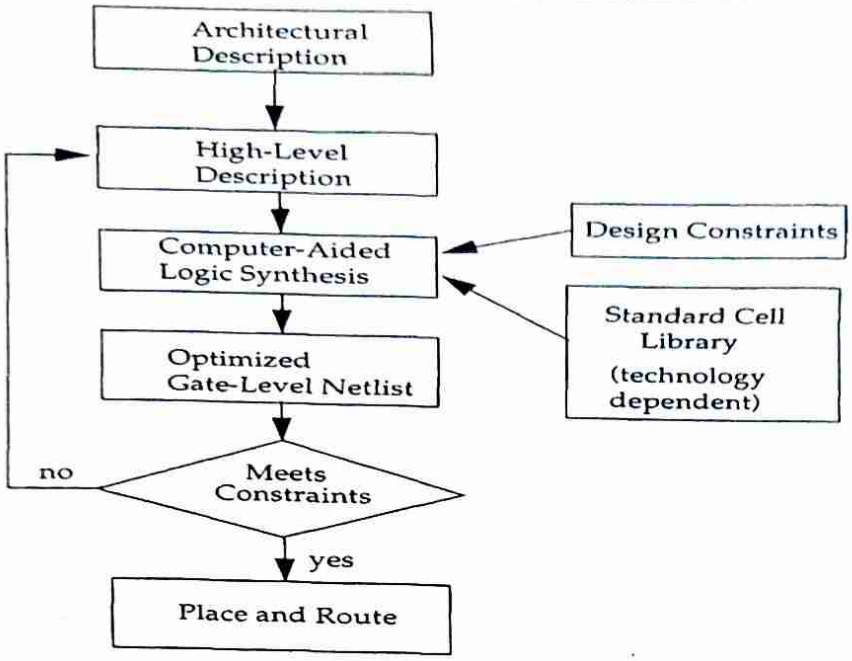
III SESSIONAL TEST scheme 2020 – 21 ODD SEMESTER

SET – B

Degree	: B.E	Semester	: V A & B
Branch	: Electronics and Communication Engg	Course Code	: 18EC56
Course Title	: Verilog HDL	Date	: 9/1/2021
Duration	: 90 Minutes	Max Marks	: 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks
1(a)	**assign and deassign:** `// Negative edge-triggered D-flipflop with asynchronous reset` `module edge_dff(q, qbar, d, clk, reset);` `// Inputs and outputs` `output q,qbar;` `input d, clk, reset;` `reg q, qbar; //declare q and qbar are registers` `always @(negedge clk) //assign value of q & qbar at active edge of clock.` `begin` `q = d;` `qbar = ~d;` `end` `always @(reset) //Override the regular assignments to q and qbar` `//whenever reset goes high. Use of procedural continuous` `//assignments.` `if(reset)` `begin //if reset is high,override regular assignments to q with` `//the new values, using procedural continuous assignment.` `assign q = 1'b0;` `assign qbar = 1'b1;` `end` `else` `begin //If reset goes low, remove the overriding values by` `//deassigning the registers. After this the regular` `//assignments q = d and qbar = ~d will be able to change` `//the registers on the next negative edge of clock.` `deassign q;` `deassign qbar;` `end` `endmodule` **force and release:** `module stimulus;` `…` `…` `//instantiate the d-flipflop` `edge_dff dff(Q, Qbar, D, CLK, RESET);` `…` `…` `initial` `begin` `//these statements force value of 1 on dff.q between time 50 and` `//100, regardless of the actual output of the edge_dff.` `#50 force dff.q = 1'b1; //force value of q to 1 at time 50.` `#50 release dff.q; //release the value of q at time 100.` `end` `…` `…` `endmodule`	6

(b)	 Explanation	6
(C)	**Limitations:** human error, never be sure that the design constraints were going to be met, turnaround time, What-if scenarios were hard to verify, little consistency in design styles, a bug, Timing, area, and power dissipation in library cells, Design reuse was not possible. **Automated logic synthesis:** less prone to human error, ensure that all constraints have been met, Conversion from high-level design to gates is fast, Turnaround time for redesign of blocks is shorter, `What-if` scenarios are easy to verify, Logic synthesis tools optimize the design as a whole, the bug is eliminated, Logic synthesis tools allow `technology-independent` design, Design reuse is possible.	6
2(a)	System tasks for *file output, displaying hierarchy, strobing, random number generation, memory initialization,* and *value change dump.*	6
(b)	**Limitations:** human error, never be sure that the design constraints were going to be met, turnaround time, What-if scenarios were hard to verify, little consistency in design styles, a bug, Timing, area, and power dissipation in library cells, Design reuse was not possible.	6
(C)	**Technology mapping and optimization** Until this step, the design description is independent of a specific *target technology.* In this step, the synthesis tool takes the internal representation and implements the representation in gates, using the cells provided in the technology library. In other words, the design is *mapped* to the desired *target technology.* Suppose you want to get your IC chip fabricated at *ABC Inc.* ABC Inc. has 0.65 micron CMOS technology, which they call *abc_100* technology. Then, abc_100 becomes the target technology. You must therefore implement your internal design representation in gates, using the cells provided in abc_100 technology library. This is called *technology mapping*. Also, the implementation should satisfy design constraints such as timing, area, and power. Some local optimizations are done to achieve the best results for the target technology. This is called *technology optimization* or *technology-dependent optimization.*	6

	Technology library The *technology library* contains *library cells* provided by ABC Inc. The term *standard cell library* used earlier in the chapter and the term *technology library* are identical and are used interchangeably. To build a technology library, ABC Inc. decides the range of functionality to provide in its library cells. As discussed earlier, library cells can be basic logic gates or macro cells such as adders, ALUs, multiplexers, and special flip-flops. The library cells are the basic building blocks that ABC Inc. will use for IC fabrication. Physical layout of library cells is done first. Then, the area of each cell is computed from the cell layout. Then, modeling techniques are used to estimate the timing and power characteristics of each library cell. This process is called *cell characterization*. Finally, the each cell is described in a format that is understood by the synthesis tool. The cell description contains information about the following: • Functionality of the cell • Area of the cell layout • Timing information about the cell • Power information about the cell **Design constraints** Design constraints typically include the following: • *Timing*—The circuit must meet certain timing requirements. An internal static timing analyzer checks timing. • *Area*—The area of the final layout must not exceed a limit. • *Power*—The power dissipation in the circuit must not exceed a threshold.	
3(a)	**Tasks:** • There are delay, timing, or event control constructs in the procedure. • The procedure has zero or more than one output arguments. • The procedure has no input arguments. **Functions:** • There are no delay, timing, or event control constructs in the procedure. • The procedure returns a single value. • There is at least one input argument.	6
(b)	`// Port declarations from the I/O diagram` `output out;` `input i0, i1, i2, i3;` `input s1, s0;` `//output declared as register` `reg out;` `//recompute the signal out if any input signal changes.` `//All input signals that cause a recomputation of out to` `//occur must go into the always @(...) sensitivity list.` `always @(s1 or s0 or i0 or i1 or i2 or i3)` `begin` `  case ({s1, s0})` `  2'b00: out = i0;` `  2'b01: out = i1;` `  2'b10: out = i2;` `  2'b11: out = i3;` `  default: out = 1'bx;` `  endcase` `end` `endmodule`	6

4(a)	Four types of looping statements in Verilog: while, for, repeat, and forever.	6
(b)	**Special Features of Blocks** Nested blocks, `//Nested blocks` `initial` `begin` `x = 1'b0;` `fork` `#5 y = 1'b1;` `#10 z = {x, y};` `join` `#20 w = {y, x};` `end` named blocks, `//Named blocks` `module top;` `initial` `begin: block1 //sequential block named block1` `integer i; //integer i is static and local to block1` `// can be accessed by hierarchical name, top.block1.i` `…` `…` `end` `initial` `fork: block2 //parallel block named block2` `reg i; // register i is static and local to block2` `// can be accessed by hierarchical name, top.block2.i` `…` `…` `join` Disabling of named blocks: `initial` `begin` `flag = 16'b 0010_0000_0000_0000;` `i = 0;` `begin: block1 //The main block inside while is named block1` `while(i < 16)` `begin` `if (flag[i])` `begin` `$display("Encountered a TRUE bit at element number %d", i);` `disable block1; //disable block1 because you found true bit.` `end` `i = i + 1;` `end` `end` `end`	6

Course in charge

Module Coordinator

HOD/ECE

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGG

Course:Verilog HDL/18EC56 | **sem:5** | **sec:A&B**

Sl. No.	USN No.	Name	IA1	IA2	IA3	A1	A2	A3	Average Assignment	Average of three IA's	Final IA(Assignment+IA)
1	1KS18EC001	A N BHOOMIKA CHOWDARY	29	26	28	10	10	10	10	28	38
2	1KS18EC002	ABHISHEK.V	29	25	23	10	10	10	10	26	36
3	1KS18EC003	ADITHI.S	29	26	26	10	10	10	10	27	37
4	1KS18EC004	AISHWARYA BANDIGANI	30	27	16	10	10	10	10	25	35
5	1KS18EC005	AISHWARYA R	30	23	23	10	10	10	10	25	35
6	1KS18EC006	AKASH R	25	24	7	10	10	10	10	19	29
7	1KS18EC007	AKHILA V	28	28	17	10	10	10	10	24	34
8	1KS18EC008	ANAGHA S	25	21	13	10	10	10	10	20	30
9	1KS18EC009	ANANYA ANANTH	26	22	20	10	10	10	10	23	33
10	1KS18EC010	ASHRITHA S C	28	27	21	10	10	10	10	25	35
11	1KS18EC011	AYEESHA RUMAN	29	27	14	10	9	9	9	23	33
12	1KS18EC012	C A SUSHMA	24	25	17	10	10	10	10	22	32
13	1KS18EC013	C M CHAITHANYA VARDHAN	25	22	1	10	0	0	3	16	19
14	1KS18EC014	CHANDAN Y C	25	24	23	10	10	10	10	24	34
15	1KS18EC015	CHARAN G	22	25	20	10	9	9	9	22	32
16	1KS18EC016	CHINNAPU CHARAN TEJA REDDY	25	25	24	10	10	10	10	25	35
17	1KS18EC017	CHITHRITHA G R	28	26	29	10	10	10	10	28	38
18	1KS18EC018	DARSHAN V	24	26	21	10	10	10	10	24	34
19	1KS18EC019	DARSHAN S	27	27	28	10	10	10	10	27	37
20	1KS18EC020	DEEKSHA S N	27	25	6	10	10	10	10	19	29
21	1KS18EC021	DEEPTHI ANDANI	28	28	17	10	10	10	10	24	34
22	1KS18EC022	DHANUSHREE C	27	26	2	10	10	10	10	18	28
23	1KS18EC023	DHEERAJ M S	28	26	25	10	10	10	10	26	36
24	1KS18EC024	DHRITHIRHUTH RAJANNA	28	23	20	10	10	10	10	24	34
25	1KS18EC025	DINESH KUMAR NAYAK	28	27	25	10	10	10	10	27	37
26	1KS18EC026	DIVAKARBABU Y	28	26	25	10	10	10	10	26	36
27	1KS18EC027	G.J.NITHIN	29	26	16	10	10	10	10	24	34
28	1KS18EC028	GANESH P	27	25	22	10	10	10	10	25	35
29	1KS18EC029	GOKUL G	25	26	22	10	10	10	10	24	34
30	1KS18EC030	HARSH SHARMA	28	26	10	10	8	8	9	21	30
31	1KS18EC031	HARSHITHA S	23	29	26	10	10	10	10	26	36
32	1KS18EC032	JAHNAVI A P	26	23	7	10	10	10	10	19	29
33	1KS18EC033	JANHAVI K P	28	24	24	10	10	10	10	25	35
34	1KS18EC034	JHANAVI V	28	25	12	10	10	10	10	22	32
35	1KS18EC035	JISHNU S	24	20	26	10	10	10	10	23	33
36	1KS18EC036	JYOTSNA B UPADHYE	28	25	10	10	10	10	10	21	31
37	1KS18EC037	K RISHIKA RAVI	27	24	21	10	10	10	10	24	34
38	1KS18EC038	KARISHMA M	29	26	10	10	10	10	10	22	32
39	1KS18EC039	KOMALA K V	28	28	9	10	10	10	10	22	32
40	1KS18EC040	LAVANYA M	27	26	12	10	10	10	10	22	32
41	1KS18EC041	M.NIHITHA YADAV	24	21	22	10	10	10	10	22	32

42	1KS18EC042	MAHANTH SAI M	28	22	17	10	10	10	10	22	32
43	1KS18EC043	MANOJ G S	25	24	18	10	10	10	10	22	32
44	1KS18EC044	MEGHA R	28	23	23	10	10	10	10	25	35
45	1KS18EC045	MEGHANA B S	29	26	21	10	10	10	10	25	35
46	1KS18EC046	MEGHANA GOWDA V	27	24	10	10	10	10	10	20	30
47	1KS18EC047	MOHAMMED FAIZAN SHAFI	26	24	6	10	10	10	10	19	29
48	1KS18EC048	MONISHA B R	28	25	18	10	10	10	10	24	34
49	1KS18EC049	N S V JASHWANTH	26	22	8	10	10	10	10	19	29
50	1KS18EC050	NAGA OMKAR N	27	24	22	10	8	8	9	24	33
51	1KS18EC051	NAGASHREE A	30	23	15	10	10	10	10	23	33
52	1KS18EC052	NAMITH R	28	24	14	10	10	10	10	22	32
53	1KS18EC053	NAVYA M S	26	26	24	10	10	10	10	25	35
54	1KS18EC054	NIHARIKA S A	24	27	24	10	10	10	10	25	35
55	1KS18EC055	NIROSHA G J	26	24	23	10	10	10	10	24	34
56	1KS18EC056	NISHANTH J RAO	28	22	15	10	9	9	9	22	31
57	1KS18EC057	P SAI GOVARDHAN	26	23	24	10	10	10	10	24	34
58	1KS18EC058	PARIKSHITH S	28	24	17	10	10	10	10	23	33
59	1KS18EC059	PAVAN KUMAR P	28	21	7	8	8	8	8	19	27
60	1KS18EC060	POOJA S	27	25	17	10	10	10	10	23	33
61	1KS18EC061	PRAKRUTHI S H	26	28	18	10	10	10	10	24	34
62	1KS18EC063	PUNEETH M	27	24	10	10	10	10	10	20	30
63	1KS18EC064	PURUSHOTHAM V R	28	26	18	10	10	10	10	24	34
64	1KS18EC066	RAGHAVENDRA.K.P	28	27	27	10	8	8	9	27	36
65	1KS18EC067	RAGHU B T	27	26	25	10	9	9	9	26	35
66	1KS18EC068	RAJ KRISHNA	28	25	4	10	10	10	10	19	29
67	1KS18EC069	RAJATH S BHUSHAN	19	20	7	10	10	10	10	15	25
68	1KS18EC070	RAM BAHADUR MAHARA	28	23	13	10	9	9	9	21	31
69	1KS18EC071	RASETTY SANDEEP	28	22	1	6	0	0	2	17	19
70	1KS18EC073	RITHVIK P	28	23	25	10	10	10	10	25	35
71	1KS18EC074	S MANOJ	28	23	25	10	10	10	10	25	35
72	1KS18EC075	S RAHUL	28	26	13	10	10	10	10	22	32
73	1KS18EC076	S TUSHAR HARINATH	29	27	23	10	9	9	9	26	36
74	1KS18EC077	SAGAR T C	28	22	14	8	10	10	9	21	31
75	1KS18EC078	SANJANA B	29	28	16	10	10	10	10	24	34
76	1KS18EC079	SANKET B PASCHAPURI	26	24	3	10	9	9	9	18	27
77	1KS18EC080	SHASHANK H K	28	25	6	10	10	10	10	20	30
78	1KS18EC081	SHEETAL N GOWDA	27	25	20	10	10	10	10	24	34
79	1KS18EC082	SHIVA SHANKAR.B	28	24	17	10	10	10	10	23	33
80	1KS18EC083	SHREYA V DEV	28	23	20	10	10	10	10	24	34
81	1KS18EC084	SHREYAS C	29	23	16	10	10	10	10	23	33
82	1KS18EC085	SHREYAS D R	28	23	26	10	10	10	10	26	36
83	1KS18EC086	SHRIKANTH C K	17	24	1	6	6	6	6	14	20
84	1KS18EC087	SIRI RAVINATH	28	23	7	10	10	10	10	19	29
85	1KS18EC088	SIRISHA.M	30	25	14	10	10	10	10	23	33
86	1KS18EC090	SOMASHEKAR M	28	27	25	10	10	10	10	27	37
87	1KS18EC091	SUDHEER B	29	22	16	10	10	10	10	22	32
88	1KS18EC092	SUJAY R	29	27	23	10	10	10	10	26	36
89	1KS18EC093	SUPRIYA S	29	21	27	10	10	10	10	26	36
90	1KS18EC094	SURAJ V GHORPADE	29	26	23	10	10	10	10	26	36
91	1KS18EC095	SUSHMA.A.V	29	28	8	10	10	10	10	22	32
92	1KS18EC096	SUSHMITHA R	28	21	26	10	10	10	10	25	35

93	1KS18EC097	THANUSH R S	28	25	13	10	10	10	10	22	32
94	1KS18EC098	THANUSHREE D	29	24	23	10	10	10	10	25	35
95	1KS18EC099	VAISHNAVI G	28	25	14	10	10	10	10	22	32
96	1KS18EC100	VAKKALA GADDA ANIL	22	23	0	5	5	5	5	15	20
97	1KS18EC101	VANDANA K	28	25	25	10	10	10	10	26	36
98	1KS18EC102	VARSHINI.B.M	29	27	7	10	10	10	10	21	31
99	1KS18EC103	VASANTH PAI.M	29	24	6	10	10	10	10	20	30
100	1KS18EC104	VIJAY BABU K	28	26	16	10	10	10	10	23	33
101	1KS18EC105	VINAY K	29	25	15	10	10	10	10	23	33
102	1KS18EC106	VINAY S	28	26	7	10	10	10	10	20	30
103	1KS18EC108	VISHAL MADHUSUDAN	28	28	6	10	10	10	10	21	31
104	1KS18EC109	VISHWAS P	29	25	9	10	10	10	10	21	31
105	1KS18EC110	VIVEKGOWDA J	26	26	14	10	10	10	10	22	32
106	1KS18EC111	VRINDHA SHAM BHATT	29	24	24	10	10	10	10	26	36
107	1KS19EC400	HEMANTHA V	23	15	9	10	10	10	10	16	26
108	1KS19EC401	KARTHIK B P	29	15	19	10	10	10	10	21	31
109	1KS19EC402	KRISHNAPRASAD B	24	15	19	10	10	10	10	19	29
110	1KS19EC403	NAVEEN G	29	13	19	10	8	8	9	20	29
111	1KS19EC405	PRUTHVI DINESH	26	23	3	10	8	8	9	17	26
112	1KS19EC406	RAGHOTHAM C G	25	21	11	10	10	10	10	19	29
113	1KS19EC407	SADANA M	29	22	8	10	10	10	10	20	30
114	1KS19EC408	SINDHU G	29	23	16	10	10	10	10	23	33
115	1KS19EC409	VARSHA M S	28	21	7	10	10	10	10	19	29

K S INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

2020 – 21 (Odd)

List of students who are identified as slow learners and their marks in every internals

Subject with Code: **Verilog HDL (18EC56)**
Semester and Section: V A & B

Sl. No.	USN Number	Name of the student	Remedial class attendance Date: 13/10/2020 & 26/11/2020	First Test Marks (30)	Second Test Marks (30)	Third Test Marks (30)	Average Test Marks (40)
1	1KS18EC035	JISHNU S	P, P	24	20	26	33
2	1KS18EC069	RAJATH S BHUSHAN	P, P	19	20	7	25
3	1KS18EC086	SHRIKANTH C K	P, P	17	24	1	20
4	1KS19EC400	HEMANTHA V	P, P	23	15	9	26
5	1KS19EC401	KARTHIK B P	P, P	29	15	19	31
6	1KS19EC402	KRISHNAPRASAD B	P, P	24	15	19	29
7	1KS19EC403	NAVEEN G	P, P	29	13	19	29

Sunil Kumar. GR

Name and Signature of the Faculty

Signature of the HOD

K. S. INSTITUTE OF TECHNOLOGY, BANGALORE – 560109

Department of Electronics and Communication

CHALLENGING QUESTIONS

Course Title: VERILOG HDL Course Code: 18EC56

---

1. List and explain short comings of VHDL.
2. Explain the declaration of constant, variable and signal in VHDL with examples.
3. Explain the design tool flow followed in VLSI design with neat flow diagram.
4. Construct a VHDL program for two 4-bit comparator using dataflow descriptions.
5. Construct a VHDL program for half adder in behavioral descriptions.
6. Write VHDL data flow descriptions of 1-bit full adder.
7. Construct a VHDL program for binary to gray convertor using dataflow descriptions.
8. Write VHDL data flow descriptions for 2 to 4 decoders.

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

DEARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

TEACHING AND LEARNING

PEDAGOGY REPORT

Academic Year	2020-21 (Odd)
Name of the Faculty	Mr. Sunil Kumar G R
Course Name /Code	Verilog HDL/18EC56
Semester/Section	V/A & B
Activity Name	Online Quiz
Topic Covered	Basics Concepts of Verilog HDL
Date	31/12/2020
No. of Participants	133
Objectives/Goals	To get familiar with Verilog language.
ICT Used	Google Forms
Appropriate Method/Instructional materials/Exam Questions Faculty covered the concepts related to Verilog programming, and students were asked to attend the quiz and give their responses through Google Forms.	
Relevant PO's	1,5,12
Significance of Results/Outcomes	The activity improved student' concepts of Verilog HDL language.
Reflective Critique	Students fared well with a score of 8.8 out of 10.

Proofs (Photographs/Videos/Reports/Charts/Models):

The below link gives the detail of the responses received for the quiz and how they have fared well in the quiz.

https://docs.google.com/forms/d/1w4msWOgySPkMhkBgjkzoOYKI6T7dtZdiQ3MjiAMQi9I/edit#responses

Signature of Course In charge

Signature of HOD/ECE

K. S. INSTITUTE OF TECHNOLOGY, BANGALORE – 560109

Department of Electronics and Communication

QUESTION BANK

Course Title : VERILOG HDL Course Code : 18EC56

Module – 1

1. Explain a typical design flow for designing VLSI IC circuits using the block diagram.
2. Explain the different levels of Abstraction used for programming in Verilog.
3. Explain a top-down design methodology and a bottom-up design methodology.
4. Explain the factors that have made Verilog HDL popular

Module – 2

5. Write a note on i) Comments ii) Number Specification iii) X and Z values and iv) Identifiers and Keywords with suitable examples.
6. Explain $display and $monitor tasks with examples.
7. Explain the trends in HDLs.
8. Explain the following data types with an example in Verilog module: (i) Nets, (ii) Register, (iii) Integers, (iv) Real, (v) Time Register
9. Explain Compiler Directives ‘define and ‘include with examples.
10. Explain the following data types with an example in Verilog module: (i) Arrays, (ii) Memories, (iii) Parameters, (iv) Strings.
11. Explain the port connection rules.
12. Explain the two methods of connecting ports to external signals with an example.

Module – 3

13. Discuss different types of Gates with respect to logic symbols, gate instantiation and truth tables.
14. What are rise, fall and turnoff delays? How they are specified in Verilog.
15. Write a Verilog code and stimulus in gate level description for 4-bit full Adder.
16. Explain regular assignment delay, implicit assignment delay, net declaration delay for continuous assignment statements.
17. Write the Verilog code for 4-to-1Multiplexer using (i) Logic Equations and (ii) Conditional Operator.
18. Write a Verilog dataflow description for 4-bit Full Adder with carry look ahead.
19. What would be the output of the following a= 4’b0111, and b= 4’b1001, (i) &b, (ii) a<<2, (iii) {a,b}, (iv) {2{b},a}, (v) a^b, (vi) a|b
20. Write a Verilog dataflow description for 4-bit Ripple Carry Counter.

Module – 4

21. Explain sequential blocks and parallel blocks with examples.
22. Explain three types of delay control for procedural assignments.
23. Construct a 4-to-1 multiplexer with the behavioral case statement; test its functionality by writing a test bench.
24. Chart out the differences between tasks and functions.
25. Discuss special features of block statements.
26. Illustrate with example four types of looping statements in Verilog.
27. Illustrate Tasks and Functions with respect to: (i) Usage condition, (ii) Declaration and Invocation, and (iii) Examples
28. Construct an 8-to-1 multiplexer with the behavioral case statement; test its functionality by writing a test bench.
29. Explain conditional compilation and conditional execution with examples.
30. Explain any three useful system tasks.
31. Discuss the two ways to override parameter values with examples.
32. What is logic synthesis? Illustrate the designer's mind as the logic synthesis tool.
33. Discuss at least six limitations of design process and how automation of logic synthesis tool has addressed these limitations.
34. Interpret the following statements into Verilog constructs: (i) assign out = (a & b) | c; (ii) assign {c_out, sum} = a+b+c_in; (iii) assign out= (s)? i1 : i0;

Module – 5

35. Discuss the logic synthesis flow from RTL to Gates with a neat diagram.
36. Discuss the following Procedural Continuous Assignments: (i)assign and deassign, (ii) force and release.
37. Illustrate the automated logic synthesis tool with a neat flow diagram.
38. Discuss the limitations of manual synthesis.
39. Explain event-based timing control with examples.
40. Explain two structured procedure statements with examples.

CBCS SCHEME

USN | | | | | | | | | | |

18EC56

Fifth Semester B.E. Degree Examination, July/August 2021
Verilog HDL

Time: 3 hrs. Max. Marks: 100

Note: *Answer any FIVE full questions.*

1 a. Explain the various stages used in VLSI design with a neat flow diagram. (08 Marks)

b. Design a 4-bit ripple carry counter using a top-down design methodology. (08 Marks)

c. Compare the HDL programming to traditional software programming. (04 Marks)

2 a. Give the importance of stimulus block. Explain the different styles of stimulus block used for testing the design. (08 Marks)

b. Explain the different levels of abstraction. (06 Marks)

c. Write a pseudo verilog code for 4-bit ripple carry adder with following description.
   i) Define a module FA with input A, B C in, sum and carry with no internals.
   ii) Instantiate 4 full adders of the type FA in the module Ripple-Add and name them as FA0, FA1, FA2 and FA3. (06 Marks)

3 a. Illustrate with examples the data types used to define nets, registers, vectors and arrays. (08 Marks)

b. Differentiate i) $display and$ monitor ii) $stop and $finish with examples. (06 Marks)

c. Declare a top-level module as TOP for stimulus. Define a constant N of size 8. IN_REG (8 bit) LOAD_EN(1-bit), LOAD_VAL (8-bit) and CLK(1bit) as register variables, and OUI_ REG (8-bit) as wire. Instantiate the module shift_reg and call it as SRI. Connect the port by named list. (06 Marks)

4 a. Illustrate with example the post connection rule of verilog HDL programming. (08 Marks)

b. Draw the logic diagram of SR latch. Develop the verilog code for SR latch. Identify the components and hence write the test bench to verify the functionality. (08 Marks)

c. Declare the following variables in verilog.
   i) Net 'A is fixed to logic value '0' at declaration
   ii) Vector register, Address_bus of 41 bit wide
   iii) A memory MEM containing 256 words of 64 bit each
   iv) An integer called count. (04 Marks)

5 a. Design a 4-bit ripple carry full adder using 1-bit full adder. Develop the verilog code for a 4-bit ripple carry full adder using gate level modeling. Verify the functionality with appropriate test bench. (08 Marks)

b. Given A = 5'b10101 ; B = 5'b11101 ; C = 5'b11001 ; D = 5'b10011. Evaluate.
   i) Y = A & B ii) Y = ~ (& C) iii) Y = C ^ D
   iv) Y = C% A v) Y = A + (D>>>1) vi) Y = {B[3], C[2], A} (06 Marks)

c. Discuss the gate delays along with its types of delay specification. (06 Marks)

6 a. Design a 4-bit ripple carry counter using TFF. Write the verilog code using data flow modeling. Verify the code with appropriate test bench. (08 Marks)

b. Design a 2×1 MUX using bufif0 and bufif1 gates. Write the verilog code using gate level modeling for the given delay specification.

	Min	Max	Typ
Rise	1	3	2
Fall	3	5	4
Turnoff	5	7	6

(06 Marks)

c. Discuss the types of delays used in the continuous assignment statement. (06 Marks)

7 a. i) Differentiate blocking and non-blocking statement with appropriate examples.
ii) Design a clock with period 40 and a duty cycle of 25% by using the always and initial statement. The value of clock at time = 0 is initialized to 0. Display the value. (08 Marks)

b. Design a 4×1 MUX and develop a verilog code using case statement. (06 Marks)

c. Bring out the differences and similarities between task and function. (06 Marks)

8 a. Compare sequential and parallel block with appropriate example. (06 Marks)

b. Define a task to compute the parity of a 16-bit data. Write a verilog code to call task calc-parity to compute the parity. Display the message as even or odd parity. (08 Marks)

c. Discuss the for loop and forever statement with example. (06 Marks)

9 a. Illustrate with examples the system tasks related to files. (06 Marks)

b. Write a verilog program for a positive edge triggered DFF with asynchronous clear (q = 0) and preset (q = 1) using assign and deassign statements. (06 Marks)

c. Give the importance of parameter overriding. Explain the two techniques of parameter overriding with examples. (08 Marks)

10 a. List the limitation of manually obtained gate level synthesis of design. How these are analyzed and addressed using automated logic synthesis tools. (08 Marks)

b. Discuss in detail the steps involved in the logic synthesis flow from RTL to gates with a neat flow diagram. (08 Marks)

c. Interpret the gatelevel netlist diagram for the following when run on a synthesis tool.

i) assign out = (Sel)? I1 : I0 ;

ii) always @ (posedge clk)
q <= d ;

(04 Marks)

* * * * *

CBCS SCHEME

USN | | | | | | | | | | |

18EC56

Fifth Semester B.E. Degree Examination, Jan./Feb. 2021
Verilog HDL

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Module-1

1 a. Explain the typical design flow for VLSI IC circuit using block diagram. **(08 Marks)**

b. Explain the trends in HDLs (Hardware Description Languages). **(04 Marks)**

c. Apply the bottom-up methodology to demonstrate the design of 4-bit ripple carry counter. **(08 Marks)**

OR

2 a. Define Module and Instance. Describe 4 different levels of abstractions used in Verilog HDL to describe target design. **(10 Marks)**

b. Explain top down design methodology and bottom up design methodology. **(10 Marks)**

Module-2

3 a. What are system tasks and compiler directives? Explain with example. **(08 Marks)**

b. Check the correctness of the following legal strings. If not, write the correct strings.

i) "This is a string displaying the % sign"
ii) "Out = in1 + in2"
iii) "Please ring a bell \ 007"
iv) "This is a backslash \ character \ n"

(04 Marks)

c. Declare the following variables in verilog:

i) An 8-bit vector net called a_in
ii) A 32 bit storage register called address. Bit 31 must be in MSB. Set the value of the reg. to a 32 bit decimal number equal to 3.
iii) An integer called count
iv) A time variable called snap_shot
v) An array called delays, Array contains 20 elements of the type integer
vi) A memory MEM containing 256 words of 64 bits each
vii) A parameter cache-size equal to 512.

(08 Marks)

OR

4 a. With a neat block diagram, explain the components of a verilog module by highlighting mandatory blocks. **(08 Marks)**

b. Explain the port connection rules of verilog HDL. **(08 Marks)**

c. A 4-bit parallel shift register has I/O pins as shown in Fig.Q.4(c) below. Write the module definition for this module shift_reg. Include the list of ports and port declaration. **(04 Marks)**

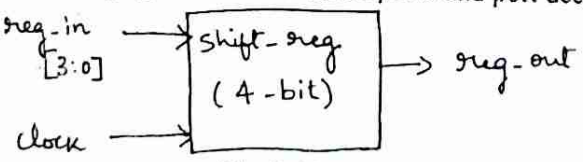


Fig.Q.4(c)

Module-3

5 a. Use gate level description of verilog HDL to design 4:1 MUX truth table, gate level block, logic expression and logic diagram. Write the stimulus block. (10 Marks)

b. Write gate level description to implement y = ab + c with 5 and 4 time units of gate delay for AND and OR gate respectively. Also write the stimulus block and simulation waveforms. (10 Marks)

OR

6 a. Write the dataflow modeling verilog code for 4-to-1 multiplexer using
i) Logic equation ii) Conditional operator. (10 Marks)

b. Explain assignment delay, implicit assignment delay and net declaration delay for continuous assignment statements with examples. (04 Marks)

c. Write a dataflow level verilog code using + and { } operators for 4-bit full adders. (06 Marks)

Module-4

7 a. Explain the blocking assignments and non-blocking assignment statements with relevant examples. (08 Marks)

b. Explain briefly the different types of event based timing control in verilog. (08 Marks)

c. Write a note on the following loop statements:
i) While loop ii) Forever loop. (04 Marks)

OR

8 a. Write a verilog behavioral code for 4 to 1 MUX using CASE statement. (08 Marks)

b. Explain the sequential and parallel blocks with examples. (08 Marks)

c. Define a function to multiple two 4-bit numbers 'a' and 'b'. The output is an 8 bit value. Invoke function by using stimulus and check results. (04 Marks)

Module-5

9 a. Write a note on:
i) Assign and deassign
ii) Overriding parameters. (10 Marks)

b. Create a design that uses the full adder. Use a conditional compilation ('if def.). Compile the fulladder 4 with def param statement if the text macro DPARAM is defined by the 'define statement; otherwise, compile the Fulladder4 with module instance parameter values. (06 Marks)

c. What will be the output of the $display statement shown below

```
Module TOP;
A a1( );
end module
Module A;
B b1( );
end module
Module B;
initial
$display {"I am inside instance %m"); end module.
```

(04 Marks)

OR

10 a. With a neat flow chart explain computer-aided logic synthesis process. (10 Marks)

b. Write RTL description for magnitude comparator. (06 Marks)

c. What is logic synthesis? (04 Marks)

* * * * *

CBCS SCHEME

USN | | | | | | | | | | |

17EC53

Fifth Semester B.E. Degree Examination, July/August 2021
Verilog HDL

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions.

1	a.	Explain the typical design flow for designing VLSI IC circuits.	(10 Marks)
	b.	Discuss the evaluation of computer aided design.	(05 Marks)
	c.	Explain top-down design methodology.	(05 Marks)
2	a.	Discuss modules, instances with the help of 4-bit ripple carry counter example.	(10 Marks)
	b.	Describe instance and instantiation with example.	(05 Marks)
	c.	Explain stimulus and design block with an example.	(05 Marks)
3	a.	Discuss the data types used in verilog with an example.	(10 Marks)
	b.	Explain system task and compiler directives in verilog.	(10 Marks)
4	a.	Explain components of verilog module with an example.	(10 Marks)
	b.	Explain port declaration, port connection rules and connecting ports to external signals.	(10 Marks)
5	a.	Write a verilog gate level description for 4:1 multiplexes also write stimulus block.	(10 Marks)
	b.	Explain rise delay, fall delay, turn off delay, min value, typical value and max value.	(10 Marks)
6	a.	Describe continuous assignment statement and implicit continuous assignment statement.	(10 Marks)
	b.	Explain arithmetic and logical operators with example.	(10 Marks)
7	a.	Explain blocking and non blocking procedural assignment in behavioral modeling.	(10 Marks)
	b.	Describe event-based-timing control mechanism in behavioral modeling.	(10 Marks)
8	a.	Explain conditional statements. Using if and else write a verilog HDL program for D_FF.	(10 Marks)
	b.	Describe multiway branching. Use case statement and write verilog program for 3-bit binary counter.	(10 Marks)
9	a.	Why we use VHDL? What are the short comings of VHDL?	(10 Marks)
	b.	Describe the design in VHDL.	(10 Marks)
10	a.	Discuss the basic building block of VHDL design with an example of dataflow behavioral description.	(10 Marks)
	b.	Write a VHDL description for 4 bit ripple carry adder, also write the circuit diagram for same.	(10 Marks)

* * * * *

CBCS SCHEME

USN | | | | | | | | | | |

15EC53

Fifth Semester B.E. Degree Examination, July/August 2021
Verilog HDL

Time: 3 hrs. Max. Marks:80

Note: ***Answer any FIVE full questions.***

1 a. Explain a typical design flow for designing VLSI IC circuit using block diagram. **(06 Marks)**

b. Develop virolog code for 4-bit ripple carry counter and with neat block diagram explain design hierarchy for the same. **(10 Marks)**

2 a. Explain to-down design methodology and bottom up design methodology. **(10 Marks)**

b. Explain the importance of HDL and also mention useful features of verilog HDL. **(06 Marks)**

3 a. Explain the following data-types with an example in verilog :
i) Nets ii) Registers iii) Vectors iv) Arrays. **(08 Marks)**

b. Explain the below mentioned system tasks NAND complier directives with examples :
i) $display ii) $monitor iii) 'Define iv) 'Include. **(08 Marks)**

4 a. With a neat block diagram explain components of verilog module. **(06 Marks)**

b. Explain part connection rules. **(06 Marks)**

c. Write a verilog code for SR latch using and gates as elements. **(04 Marks)**

5 a. What are rise, fall and turnoff delays? How they are specified in verilog. **(06 Marks)**

b. Design and develop verilog code for an 4-bit ripple carry adder using 1-bit fulladder as a component. Also write stimulus for 4-bit ripple carry fulladder. **(10 Marks)**

6 a. For the schematic network shown below. Write a verilog code for gate level implementation with delay s mentioned :

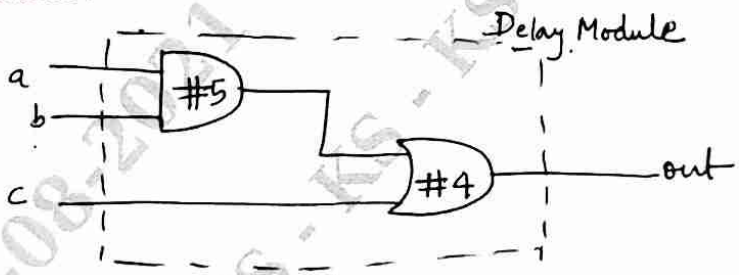


Fig.Q6(a)

Also write stimulus for the above example. **(06 Marks)**

b. Write a verilog code for :
i) 2 : 1 mux with conditional operator
ii) 4 : 1 mux with conditional operators
iii) 4 : 1 mux using logic equation
iv) 2 : 1 mux using logic equation. **(10 Marks)**

7 a. Explain with examples always and initial statements. **(08 Marks)**

b. Explain blocking assignment statements and non-blocking assignment statements with relevant examples. **(08 Marks)**

8 a. Explain sequential and parallel blocks with examples. **(08 Marks)**

b. Write a verilog code for :

i) 4 : 1 multiplexer using case statement

ii) 4 – bit counter with behavioral description. **(08 Marks)**

9 a. Explain design tool flow diagram with block diagram. **(08 Marks)**

b. i) Write VHDL dataflow description for – 4 – bit equality comparator using logic equations and block diagrams. **(04 Marks)**

ii) Write VHDL structural description for 4-bit comparator with necessary block diagrams. **(04 Marks)**

10 a. Explain the declaration of constant, variable and signal in VHDL, with example. **(08 Marks)**

b. Explain attributes in VHDL. **(04 Marks)**

c. Write a VHDL code for half adder using behavioral description. **(04 Marks)**

* * * * *

K.S.Institute of Technology,Bangalore -109
Department of Electronics and Communication Engg
5th sem Course End Survey 2020-21

Subject : Verilog HDL

Subject Code : 18EC56

Q1. How well are you able to write Verilog programs in gate, dataflow and behavioral modeling levels of abstraction?

Q2.Are you able to design and verify the functionality of digital circuits using test benches?

Q3.Can you identify suitable abstraction level (gate, dataflow and behavioral) for a particular digital design?

Q4.Are you able to write the programs more effectively using Verilog tasks, functions and directives?

Q5.How efficiently you can perform timing and delay simulation?

Sl No	Date	Name of the Student	USN	semester and section	Faculty Name	Q1	Q2	Q3	Q4	Q5
1	22-01-21	Baskar Sabarish	1ks15ec015	V B	Sunil Kumar G R	3	3	3	3	3
2	23-01-21	Akash Chandrappa Guruvannavar	1KS16EC005	5 - B	Sunil Kumar G R	2	2	2	2	2
3	23-01-21	Gowda Anupkumar Prakash	1KS16EC027	V B	Sunil Kumar G R	3	3	3	3	3
4	23-01-21	VENKATESH M N	1KS16EC107	5 B	Sunil Kumar G R	3	3	3	3	3
5	23-01-21	VIDYASAGAR R	1KS16EC110	5 B	Sunil Kumar G R	3	3	3	3	3
6	22-01-21	GAGAN KUMAR. G	1KS17EC037	5th sem, "B"sec	Sunil Kumar G R	3	3	3	3	3
7	22-01-21	Sahana GP	1ks17eco83	Vsem bsec	Sunil Kumar G R	3	3	3	3	3
8	23-01-21	Bhoomika	1KS18EC001	5th A sec	Sunil Kumar G R	3	3	3	3	3
9	22-01-21	Abhishek V	1KS18EC002	5th sem and A sec	Sunil Kumar G R	3	3	3	3	3
10	22-01-21	Abhishek V	1KS18EC002	5th sem and A sec	Sunil Kumar G R	3	3	3	3	3
11	22-01-21	Adithi	1KS18EC003	5 A	Sunil Kumar G R	3	3	3	3	3
12	22-01-21	Aishwarya Bandigani	1KS18EC004	5 th sem A section	Sunil Kumar G R	3	3	3	3	3
13	22-01-21	Aishwarya R	1KS18EC005	5 th sem,A sec	Sunil Kumar G R	2	2	2	2	2
14	25-01-21	Pavan Kumar P	1KS18EC0059	5th sem 'A' SEC	Sunil Kumar G R	3	3	3	3	3
15	22-01-21	Akash R	1ks18ec006	5thsem/Asec	Sunil Kumar G R	1	1	1	1	1
16	22-01-21	Akhila v	1ks18ec007	5th sem A sec	Sunil Kumar G R	3	2	2	2	2
17	22-01-21	Anagha.s	1KS18EC008	5 A	Sunil Kumar G R	2	2	2	2	2
18	22-01-21	Ananya Ananth	1KS18EC009	5th 'A'	Sunil Kumar G R	1	1	1	1	1
19	22-01-21	Ashritha S C	1KS18EC010	5 sem, A section	Sunil Kumar G R	2	2	2	2	2
20	23-01-21	Ayeesha Ruman	1KS18EC011	5th sem, A section	Sunil Kumar G R	3	3	3	3	3

Sl No	Date	Name of the Student	USN	semester and section	Faculty Name	Q1	Q2	Q3	Q4	Q5
21	22-01-21	C A Sushma	1KS18EC012	5 - A	Sunil Kumar G R	3	2	2	2	3
22	22-01-21	chaithanya vardhan	1KS18EC013	5th sem A sec	Sunil Kumar G R	3	3	3	3	3
23	22-01-21	chaithanya vardhan	1KS18EC013	5th sem A sec	Sunil Kumar G R	3	3	3	3	3
24	22-01-21	CHANDAN YC	1KS18EC014	5th sem A sec	Sunil Kumar G R	2	2	2	3	3
25	22-01-21	CHANDAN YC	1KS18EC014	5th sem A sec	Sunil Kumar G R	2	2	2	3	3
26	22-01-21	Charan G	1KS18EC015	5th sem A section	Sunil Kumar G R	3	2	2	2	2
27	22-01-21	Chinnapu charan Teja Reddy	1KS18EC016	5 and A	Sunil Kumar G R	2	2	2	2	1
28	22-01-21	Chithritha G R	1KS18EC017	5 th sem 'A' sec	Sunil Kumar G R	2	2	2	2	2
29	22-01-21	Darshan v	1KS18EC018	5th&A	Sunil Kumar G R	3	3	2	2	2
30	22-01-21	Darshan s	1ks18ec019	5th A	Sunil Kumar G R	2	2	2	2	2
31	22-01-21	Deeksha S N	1KS18EC020	5th sem A sec	Sunil Kumar G R	2	2	2	2	2
32	22-01-21	Deepthi Andani	1KS18EC021	5th A	Sunil Kumar G R	2	2	2	2	2
33	22-01-21	Dhanushree.C	1KS18EC022	5th sem and A sec	Sunil Kumar G R	2	2	2	2	2
34	23-01-21	Dheeraj M S	1KS18EC023	5th sem and A sec	Sunil Kumar G R	3	2	2	2	2
35	22-01-21	DHRITHIRHUTH RAJANNA	1KS18EC024	5th/A	Sunil Kumar G R	1	3	2	2	2
36	22-01-21	Dinesh Kumar nayak	1KS18EC025	5th/A	Sunil Kumar G R	2	1	2	2	2
37	23-01-21	Divakar babu y	1KS18EC026	5th/A	Sunil Kumar G R	2	2	2	2	2
38	22-01-21	Gj Nithin	1KS18EC027	5th sem A section	Sunil Kumar G R	2	2	2	2	2
39	22-01-21	Ganesh P	1KS18EC028	5 A	Sunil Kumar G R	2	2	1	2	2
40	22-01-21	Gokul.G	1KS18EC029	5 A	Sunil Kumar G R	2	2	3	2	2
41	22-01-21	Harsh Sharma	1KS18EC030	5th Sem 'A'	Sunil Kumar G R	3	3	3	3	3
42	22-01-21	Harshitha S	1KS18EC031	5th SEM A section	Sunil Kumar G R	2	2	2	2	2
43	23-01-21	Jahnavi AP	1KS18EC032	5th sem, A section	Sunil Kumar G R	3	3	3	3	3
44	22-01-21	Janhavi K P	1KS18EC033	5th sem / A sec	Sunil Kumar G R	2	2	2	2	2
45	23-01-21	Jhanavi V	1KS18EC034	5th sem , A section	Sunil Kumar G R	3	3	3	3	3
46	22-01-21	Jishnu S	1KS18EC035	V A	Sunil Kumar G R	2	2	2	2	2
47	22-01-21	Jyotsna B Upadhye	1KS18EC036	5 A	Sunil Kumar G R	3	3	3	3	3
48	22-01-21	K RISHIKA RAVI	1KS18EC037	5th SEM A Section	Sunil Kumar G R	3	3	3	3	3

No	Date	Name of the Student	USN	semester and section	Faculty Name	Q1	Q2	Q3	Q4	Q5
49	22-01-21	Karishma M	1KS18EC038	5th Semester, 'A' Section	Sunil Kumar G R	2	2	2	2	2
50	22-01-21	Lavanya m	1KS18EC040	5th A sec	Sunil Kumar G R	2	2	2	2	2
51	23-01-21	Nihitha	1KS18EC041	5th-A	Sunil Kumar G R	3	3	3	3	3
52	23-01-21	Mahanth	1KS18EC042	5th A	Sunil Kumar G R	2	2	2	2	2
53	22-01-21	Manoj.G.S	1KS18EC043	5th sem and A sec	Sunil Kumar G R	3	3	3	3	1
54	23-01-21	Manoj.G.S	1KS18EC043	5th sem and A sec	Sunil Kumar G R	3	3	3	3	1
55	22-01-21	Megha R	1KS18EC044	5 A	Sunil Kumar G R	2	2	2	2	2
56	22-01-21	Meghana B S	1KS18EC045	5th and "A" sec	Sunil Kumar G R	3	3	3	3	3
57	23-01-21	Mohammed Faizan Shafi	1KS18EC047	5th A	Sunil Kumar G R	3	3	3	3	3
58	22-01-21	Monisha B R	1KS18EC048	5A	Sunil Kumar G R	3	3	3	3	2
59	22-01-21	NSV JASHWANTH	1KS18EC049	5 A	Sunil Kumar G R	3	3	3	3	3
60	22-01-21	N Naga Omkar	1ks18ec050	5A	Sunil Kumar G R	3	3	3	3	3
61	22-01-21	Nagashree.a	1KS18EC051	5A	Sunil Kumar G R	3	3	3	3	3
62	22-01-21	Navya MS	1KS18EC053	5th 'A'	Sunil Kumar G R	3	3	2	3	2
63	22-01-21	Niharika SA	1KS18EC054	5th sem Asec	Sunil Kumar G R	2	2	2	2	2
64	22-01-21	NIROSHA GJ	1KS18EC055	5th/A	Sunil Kumar G R	2	2	2	2	2
65	25-01-21	Nishanth J Rao	1KS18EC056	5th sem A sec	Sunil Kumar G R	3	3	3	3	3
66	26-01-21	Nishanth J Rao	1KS18EC056	5th sem A sec	Sunil Kumar G R	3	3	3	3	3
67	26-01-21	Nishanth J Rao	1KS18EC056	5th sem A sec	Sunil Kumar G R	3	3	3	3	3
68	22-01-21	P SAI GOVARDHAN	1KS18EC057	5th A	Sunil Kumar G R	2	2	2	2	2
69	22-01-21	Parikshith S	1KS18EC058	5 A	Sunil Kumar G R	3	2	2	2	2
70	22-01-21	Prakruthi S H	1KS18EC061	5 B	Sunil Kumar G R	3	3	3	3	3
71	22-01-21	Prakruthi S H	1KS18EC061	5 B	Sunil Kumar G R	3	3	3	3	3
72	22-01-21	Puneeth M	1KS18EC063	5th sem B section	Sunil Kumar G R	2	2	2	2	2
73	22-01-21	Purushotham V R	1ks18ec064	5th sem b sec	Sunil Kumar G R	1	1	1	1	1
74	23-01-21	Raghavendra K P	1KS18EC066	5th sem bsec	Sunil Kumar G R	1	2	2	2	2
75	22-01-21	Raghu B T	1ks18ec067	5th sem B sec	Sunil Kumar G R	3	3	3	2	2
76	22-01-21	Raj krishna	1KS18EC068	5 sem,b sec	Sunil Kumar G R	2	2	2	2	2

Sl No	Date	Name of the Student	USN	semester and section	Faculty Name	Q1	Q2	Q3	Q4	Q5
77	22-01-21	Rajath S Bhushan	1KS18EC069	5-B	Sunil Kumar G R	3	3	3	3	3
78	23-01-21	Ram Bahadur Mahara	1KS18EC070	5 bsecion	Sunil Kumar G R	3	3	3	2	3
79	22-01-21	RASETTY SANDEEP	1KS18EC071	5th sem	Sunil Kumar G R	3	3	3	3	3
80	23-01-21	Rithvik P	1KS18EC073	5th B	Sunil Kumar G R	3	3	3	3	3
81	23-01-21	Manoj.s	1KS18EC074	5th sem & b sec	Sunil Kumar G R	3	3	3	3	3
82	22-01-21	S RAHUL	1KS18EC075	5th B	Sunil Kumar G R	3	3	3	3	2
83	23-01-21	S Tushar Harinath	1KS18EC076	5th b sec	Sunil Kumar G R	3	3	3	3	3
84	22-01-21	Sagar T C	1KS18EC077	5th b sec	Sunil Kumar G R	2	2	2	2	1
85	22-01-21	Sanjana.B	1Ks18ec078	5th bsec	Sunil Kumar G R	3	3	3	3	3
86	22-01-21	Sanket b paschapuri	1KS18EC079	5 th B sec	Sunil Kumar G R	2	2	1	2	1
87	23-01-21	Shashank H k	1ks18ec080	5 th B	Sunil Kumar G R	3	2	2	3	2
88	22-01-21	Sheetal N gowda	1KS18EC081	5 sem and B sec	Sunil Kumar G R	2	2	2	2	2
89	22-01-21	Shiva shankar	1KS18EC082	3rd sem b sec	Sunil Kumar G R	3	3	3	3	3
90	22-01-21	Shreya V. Dev.	1KS18EC083	V B	Sunil Kumar G R	1	1	1	1	1
91	22-01-21	Shreyas c	1KS18EC084	5th sem,B sec	Sunil Kumar G R	3	3	3	3	3
92	22-01-21	SHREYAS D.R	1KS18EC085	5th sem b sec	Sunil Kumar G R	1	1	1	1	1
93	22-01-21	Shrikanth c k	1KS18EC086	5 th B	Sunil Kumar G R	3	3	3	3	2
94	22-01-21	Siri ravinath	1KS18EC087	5th sem & B sec	Sunil Kumar G R	2	2	2	2	2
95	22-01-21	Sirisha M	1KS18EC088	5th Sem B Section	Sunil Kumar G R	3	3	3	3	3
96	22-01-21	Somashekar M	1KS18EC090	5th Sem B Sec	Sunil Kumar G R	3	3	3	2	2
97	23-01-21	SUDHEER B	1KS18EC091	5th sem B sec	Sunil Kumar G R	3	3	2	2	2
98	22-01-21	Sujay R	1KS18EC092	5 B	Sunil Kumar G R	3	3	3	3	3
99	22-01-21	Supriya S	1KS18EC093	5th and B section	Sunil Kumar G R	2	2	2	2	2
100	22-01-21	Suraj V Ghorpade	1KS18EC094	5th sem B	Sunil Kumar G R	3	3	3	3	3
101	22-01-21	Sushma A V	1KS18EC095	5thB	Sunil Kumar G R	3	2	2	2	2
102	22-01-21	Sushmitha.R	1KS18EC096	5 B	Sunil Kumar G R	3	3	3	3	3
103	22-01-21	THANUSH RS	1KS18EC097	Vth sem b section	Sunil Kumar G R	3	3	3	3	3
104	22-01-21	Thanushree.D	1KS18EC098	5th B sec	Sunil Kumar G R	3	3	3	3	3

Sl No	Date	Name of the Student	USN	semester and section	Faculty Name	Q1	Q2	Q3	Q4	Q5
105	1/22/2021	Vaishnavi.G	1KS18EC099	5th sem b sec	Sunil Kumar G R	2	2	2	2	2
106	1/22/2021	Anil.v	1KS18EC100	Vth semester and B section	Sunil Kumar G R	3	3	3	3	2
107	1/22/2021	Varshini B.M.	1KS18EC102	5 B	Sunil Kumar G R	2	2	2	2	2
108	1/22/2021	Varshini B.M.	1KS18EC102	5 B	Sunil Kumar G R	2	2	2	2	2
109	1/22/2021	Vasanth.pai.m	1KS18EC103	5th Sem B section	Sunil Kumar G R	3	3	3	3	3
110	1/22/2021	VIJAYBABU.K	1KS18EC104	5th sem B sec	Sunil Kumar G R	2	2	2	2	2
111	1/22/2021	VIJAYBABU.K	1KS18EC104	5th sem B sec	Sunil Kumar G R	2	2	2	2	2
112	1/22/2021	VIJAYBABU.K	1KS18EC104	5th sem B sec	Sunil Kumar G R	2	2	2	2	2
113	1/22/2021	Vinay K	1KS18EC105	5th Semester and B Section	Sunil Kumar G R	2	2	2	2	2
114	1/22/2021	Vinay K	1KS18EC105	5th Semester and B Section	Sunil Kumar G R	2	2	2	2	2
115	1/22/2021	Vinay s	1KS18EC106	5th sem b sec	Sunil Kumar G R	1	1	1	1	1
116	1/22/2021	Vishal.M	1ks18ec108	5th B	Sunil Kumar G R	2	2	2	2	2
117	1/22/2021	Sushma	1ks18ec108	5th b	Sunil Kumar G R	2	3	3	2	2
118	1/22/2021	Vishwas	1KS18EC109	5th sem B sec	Sunil Kumar G R	1	1	1	1	1
119	1/22/2021	Vivek gowda j	1KS18EC110	5th sem,B sec	Sunil Kumar G R	1	1	1	1	1
120	1/22/2021	Vrindha Sham Bhatt	1KS18EC111	5 B	Sunil Kumar G R	3	3	3	3	3
121	1/22/2021	Vrindha Sham Bhatt	1KS18EC111	5 B	Sunil Kumar G R	3	3	3	3	3
122	1/22/2021	KISHOR KUMAR N	1KS18EC404	5th sem "B" sec	Sunil Kumar G R	3	3	3	3	3
123	1/23/2021	Kishore.R	1ks18ec405	5th sem B sec	Sunil Kumar G R	3	3	3	3	3
124	1/22/2021	Hemantha	1ks19ec400	5th nd B	Sunil Kumar G R	3	3	3	3	3
125	1/22/2021	Karthik BP	1KS19EC401	5th sem and "B"	Sunil Kumar G R	3	2	2	2	2
126	1/22/2021	Krishnaprasad B	1KS19EC402	5th sem and B sec	Sunil Kumar G R	2	2	2	2	2
127	1/22/2021	NAVEEN.G	1KS19EC403	5TH SEM B SECTION	Sunil Kumar G R	3	3	2	3	2
128	1/22/2021	RAGHOTHAM C G	1KS19EC406	5th sem B section	Sunil Kumar G R	3	3	3	3	3
129	1/23/2021	Sadhana M	1KS19EC407	5th B	Sunil Kumar G R	3	3	3	3	3
130	1/23/2021	Sindhu.g	1KS19EC408	5th semester 'B'section	Sunil Kumar G R	3	2	2	3	3
131	1/22/2021	Varsha ms	1KS19EC409	5th sem B sec	Sunil Kumar G R	2	2	2	2	2
132	1/27/2021	VENKATESH MN	1KS16EC107	5B	Sunil Kumar G R	3	3	3	3	3
					NO. OF 1S	**10**	**9**	**10**	**9**	**13**
					Total count	122	121	121	121	121
					Percentage	**91.80**	**92.56**	**91.74**	**92.56**	**89.26**
					Average	**91.58**				

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGG

YEAR / SEMESTER	**III/V**
COURSE TITLE	**Verilog HDL**
COURSE CODE	**18EC56**
ACADEMIC YEAR	**2020-21**
BATCH	**2018-2022**

CO	Significance
Level 3	60% and above students should have scored >= 60% of Total marks
Level 2	55% to 59% of students should have scored >= 60% of Total marks
Level 1	50% to 54% of students should have scored >= 60% of Total marks

For Direct attainment , 50% of CIE and 50% of SEE marks are considered.

For indirect attainment, Course end survey is considered.

CO attainment is 90%of direct attainment + 10% of Indirect atttainment.

PO attainment = CO-PO mapping strength/3 * CO attainment .

SI. NO.	USN	NAME	IA1				Assignment 1				IA2				Assignment 2				IA3				Assignment 3							EXTERNAL		
			IA1	CO1	Scores	Target 60%	A1	CO1	Scores	Target 60%	IA2	CO2	Scores	Target 60%	A2	CO2	Scores	Target 60%	IA3	CO4	Scores	Target 60%	A3	CO4	Scores	Target 60%	CO5	Scores	Target 60%	SEE	Scores	Target 60%
	Maximum Marks		**30**	**18**			**10**	**6**			**30**	**6**			**10**	**2**			**30**	**12**			**10**	**6**			**4**			**60**		
1	1KS18EC001	A N BHOOMIKA CHOWDARY	29	17	3	Y	10	6	3	Y	26	6	3	Y	10	2	3	Y	28	12	3	Y	10	6	3	Y	4	3	Y	**36**	3	Y
2	1KS18EC002	ABHISHEK.V	29	17	3	Y	10	6	3	Y	25	4	3	Y	10	2	3	Y	23	11	3	Y	10	6	3	Y	4	3	Y	**34**	2	N
3	1KS18EC003	ADITHI.S	29	18	3	Y	10	6	3	Y	26	6	3	Y	10	2	3	Y	26	10	3	Y	10	6	3	Y	4	3	Y	**21**	0	N
4	1KS18EC004	AISHWARYA BANDIGANI	30	18	3	Y	10	6	3	Y	27	5	3	Y	10	2	3	Y	16	8	3	Y	10	6	3	Y	4	3	Y	**21**	0	N
5	1KS18EC005	AISHWARYA R	30	18	3	Y	10	6	3	Y	23	5	3	Y	10	2	3	Y	23	8	3	Y	10	6	3	Y	4	3	Y	**33**	2	N
6	1KS18EC006	AKASH R	25	14	3	Y	10	6	3	Y	24	5	3	Y	10	2	3	Y	7	5	0	N	10	6	3	Y	4	3	Y	**A**	3	Y
7	1KS18EC007	AKHILA V	28	16	3	Y	10	6	3	Y	28	6	3	Y	10	2	3	Y	17	11	3	Y	10	6	3	Y	4	3	Y	**33**	2	N
8	1KS18EC008	ANAGHA S	25	13	3	Y	10	6	3	Y	21	3	1	N	10	2	3	Y	13	5	0	N	10	6	3	Y	4	3	Y	**33**	2	N
9	1KS18EC009	ANANYA ANANTH	26	15	3	Y	10	6	3	Y	22	3	1	N	10	2	3	Y	20	8	3	Y	10	6	3	Y	4	3	Y	**21**	0	N
10	1KS18EC010	ASHRITHA S C	28	17	3	Y	10	6	3	Y	27	6	3	Y	10	2	3	Y	21	9	3	Y	10	6	3	Y	4	3	Y	**23**	0	N
11	1KS18EC011	AYEESHA RUMAN	29	17	3	Y	10	6	3	Y	27	6	3	Y	9	1	1	N	14	8	3	Y	9	6	3	Y	3	3	Y	**30**	1	N
12	1KS18EC012	C A SUSHMA	24	12	3	Y	10	6	3	Y	25	4	3	Y	10	2	3	Y	17	6	1	N	10	6	3	Y	4	3	Y	**23**	0	N
13	1KS18EC013	C M CHAITHANYA VARDHAN	25	15	3	Y	10	6	3	Y	22	5	3	Y	0	0	0	N	1	1	0	N	0	0	0	N	0	0	N	**8**	0	N
14	1KS18EC014	CHANDAN Y C	25	14	3	Y	10	6	3	Y	24	4	3	Y	10	2	3	Y	23	7	2	N	10	6	3	Y	4	3	Y	**32**	1	N
15	1KS18EC015	CHARAN G	22	13	3	Y	10	6	3	Y	25	5	3	Y	9	1	1	N	20	11	3	Y	9	6	3	Y	3	3	Y	**24**	0	N
16	1KS18EC016	CHINNAPU CHARAN TEJA REDDY	25	15	3	Y	10	6	3	Y	25	5	3	Y	10	2	3	Y	24	9	3	Y	10	6	3	Y	4	3	Y	**41**	3	Y
17	1KS18EC017	CHITHRITHA G R	28	18	3	Y	10	6	3	Y	26	6	3	Y	10	2	3	Y	29	12	3	Y	10	6	3	Y	4	3	Y	**39**	3	Y
18	1KS18EC018	DARSHAN V	24	14	3	Y	10	6	3	Y	26	5	3	Y	10	2	3	Y	21	9	3	Y	10	6	3	Y	4	3	Y	**28**	0	N
19	1KS18EC019	DARSHAN S	27	15	3	Y	10	6	3	Y	27	5	3	Y	10	2	3	Y	28	12	3	Y	10	6	3	Y	4	3	Y	**43**	3	Y
20	1KS18EC020	DEEKSHA S N	27	16	3	Y	10	6	3	Y	25	5	3	Y	10	2	3	Y	6	6	1	N	10	6	3	Y	4	3	Y	**21**	0	N
21	1KS18EC021	DEEPTHI ANDANI	28	17	3	Y	10	6	3	Y	28	5	3	Y	10	2	3	Y	17	10	3	Y	10	6	3	Y	4	3	Y	**37**	3	Y
22	1KS18EC022	DHANUSHREE C	27	15	3	Y	10	6	3	Y	26	5	3	Y	10	2	3	Y	2	2	0	N	10	6	3	Y	4	3	Y	**18**	0	N
23	1KS18EC023	DHEERAJ M S	28	17	3	Y	10	6	3	Y	26	5	3	Y	10	2	3	Y	25	8	3	Y	10	6	3	Y	4	3	Y	**47**	3	Y
24	1KS18EC024	DHRITHIRHUTH RAJANNA	28	16	3	Y	10	6	3	Y	23	5	3	Y	10	2	3	Y	23	8	3	Y	10	6	3	Y	4	3	Y	**23**	0	N
25	1KS18EC025	DINESH KUMAR NAYAK	28	17	3	Y	10	6	3	Y	27	5	3	Y	10	2	3	Y	25	9	3	Y	10	6	3	Y	4	3	Y	**38**	3	Y
26	1KS18EC026	DIVAKARBABU Y	28	17	3	Y	10	6	3	Y	26	5	3	Y	10	2	3	Y	25	9	3	Y	10	6	3	Y	4	3	Y	**36**	3	Y
27	1KS18EC027	G.J.NITHIN	29	17	3	Y	10	6	3	Y	26	5	3	Y	10	2	3	Y	16	11	3	Y	10	6	3	Y	4	3	Y	**22**	0	N
28	1KS18EC028	GANESH P	27	16	3	Y	10	6	3	Y	25	5	3	Y	10	2	3	Y	22	11	3	Y	10	6	3	Y	4	3	Y	**44**	3	Y
29	1KS18EC029	GOKUL G	25	14	3	Y	10	6	3	Y	26	5	3	Y	10	2	3	Y	22	11	3	Y	10	6	3	Y	4	3	Y	**32**	1	N
30	1KS18EC030	HARSH SHARMA	28	16	3	Y	10	6	3	Y	26	6	3	Y	8	1	1	N	10	8	3	Y	8	5	3	Y	3	3	Y	**51**	3	Y
31	1KS18EC031	HARSHITHA S	23	11	3	Y	10	6	3	Y	29	6	3	Y	10	2	3	Y	26	11	3	Y	10	6	3	Y	4	3	Y	**51**	3	Y
32	1KS18EC032	JAHNAVI A P	26	16	3	Y	10	6	3	Y	23	5	3	Y	10	2	3	Y	7	5	0	N	10	6	3	Y	4	3	Y	**30**	1	N
33	1KS18EC033	JANHAVI K P	28	16	3	Y	10	6	3	Y	24	4	3	Y	10	2	3	Y	24	11	3	Y	10	6	3	Y	4	3	Y	**31**	1	N
34	1KS18EC034	JHANAVI V	28	16	3	Y	10	6	3	Y	25	5	3	Y	10	2	3	Y	12	8	3	Y	10	6	3	Y	4	3	Y	**30**	1	N
35	1KS18EC035	JISHNU S	24	14	3	Y	10	6	3	Y	20	5	3	Y	10	2	3	Y	26	10	3	Y	10	6	3	Y	4	3	Y	**42**	3	Y
36	1KS18EC036	JYOTSNA B UPADHYE	28	18	3	Y	10	6	3	Y	25	5	3	Y	10	2	3	Y	10	10	3	Y	10	6	3	Y	4	3	Y	**32**	1	N
37	1KS18EC037	K RISHIKA RAVI	27	16	3	Y	10	6	3	Y	24	5	3	Y	10	2	3	Y	21	6	1	N	10	6	3	Y	4	3	Y	**21**	0	N
38	1KS18EC038	KARISHMA M	29	17	3	Y	10	6	3	Y	26	5	3	Y	10	2	3	Y	10	8	3	Y	10	6	3	Y	4	3	Y	**21**	0	N
39	1KS18EC039	KOMALA K V	28	17	3	Y	10	6	3	Y	28	5	3	Y	10	2	3	Y	9	2	0	N	10	6	3	Y	4	3	Y	**24**	0	N
40	1KS18EC040	LAVANYA M	27	17	3	Y	10	6	3	Y	26	5	3	Y	10	2	3	Y	12	2	0	N	10	6	3	Y	4	3	Y	**36**	3	Y
41	1KS18EC041	M.NIHITHA YADAV	24	15	3	Y	10	6	3	Y	21	5	3	Y	10	2	3	Y	22	9	3	Y	10	6	3	Y	4	3	Y	**41**	3	Y
42	1KS18EC042	MAHANTH SAI M	28	17	3	Y	10	6	3	Y	22	5	3	Y	10	2	3	Y	17	5	0	N	10	6	3	Y	4	3	Y	**21**	0	N
43	1KS18EC043	MANOJ G S	25	15	3	Y	10	6	3	Y	24	5	3	Y	10	2	3	Y	18	7	2	N	10	6	3	Y	4	3	Y	**37**	3	Y
44	1KS18EC044	MEGHA R	28	16	3	Y	10	6	3	Y	23	5	3	Y	10	2	3	Y	23	10	3	Y	10	6	3	Y	4	3	Y	**32**	1	N
45	1KS18EC045	MEGHANA B S	29	17	3	Y	10	6	3	Y	26	5	3	Y	10	2	3	Y	21	7	2	N	10	6	3	Y	4	3	Y	**29**	0	N

46	1KS18EC046	MEGHANA GOWDA V	27	16	3	Y	11	3	Y	10	6	3	Y	4	3	Y	24	3	1	N	16	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	10	7	2	N	3	0	N	10	6	3	Y	4	3	Y	**24**	0	N
47	1KS18EC047	MOHAMMED FAIZAN SHAFI	26	15	3	Y	11	3	Y	10	6	3	Y	4	3	Y	24	5	3	Y	14	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	6	4	0	N	2	0	N	10	6	3	Y	4	3	Y	**24**	0	N
48	1KS18EC048	MONISHA B R	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	25	5	3	Y	15	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	18	8	3	Y	10	2	N	10	6	3	Y	4	3	Y	**21**	0	N
49	1KS18EC049	N S V JASHWANTH	26	16	3	Y	10	3	Y	10	6	3	Y	4	3	Y	22	4	3	Y	13	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	8	5	0	N	3	0	N	10	6	3	Y	4	3	Y	**38**	3	Y
50	1KS18EC050	NAGA OMKAR N	27	17	3	Y	10	3	Y	10	6	3	Y	4	3	Y	24	5	3	Y	14	3	Y	5	3	Y	8	1	1	N	6	3	Y	1	1	N	22	10	3	Y	12	3	Y	8	5	3	Y	3	3	Y	**42**	3	Y
51	1KS18EC051	NAGASHREE A	30	18	3	Y	12	3	Y	10	6	3	Y	4	3	Y	23	3	1	N	14	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	15	9	3	Y	6	0	N	10	6	3	Y	4	3	Y	**40**	3	Y
52	1KS18EC052	NAMITH R	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	24	5	3	Y	14	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	14	7	2	N	7	0	N	10	6	3	Y	4	3	Y	**21**	0	N
53	1KS18EC053	NAVYA M S	26	15	3	Y	11	3	Y	10	6	3	Y	4	3	Y	26	4	3	Y	16	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	24	8	3	Y	16	3	Y	10	6	3	Y	4	3	Y	**22**	0	N
54	1KS18EC054	NIHARIKA S A	24	15	3	Y	9	3	Y	10	6	3	Y	4	3	Y	27	6	3	Y	15	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	24	8	3	Y	16	3	Y	10	6	3	Y	4	3	Y	**53**	3	Y
55	1KS18EC055	NIROSHA G J	26	16	3	Y	10	3	Y	10	6	3	Y	4	3	Y	24	4	3	Y	14	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	23	8	3	Y	15	3	Y	10	6	3	Y	4	3	Y	**33**	2	N
56	1KS18EC056	NISHANTH J RAO	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	22	5	3	Y	12	3	Y	5	3	Y	9	1	1	N	6	3	Y	2	3	Y	15	7	2	N	8	0	N	9	6	3	Y	3	3	Y	**21**	0	N
57	1KS18EC057	P SAI GOVARDHAN	26	16	3	Y	10	3	Y	10	6	3	Y	4	3	Y	23	4	3	Y	14	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	24	8	3	Y	16	3	Y	10	6	3	Y	4	3	Y	**41**	3	Y
58	1KS18EC058	PARIKSHITH S	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	24	5	3	Y	14	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	17	9	3	Y	8	0	N	10	6	3	Y	4	3	Y	**53**	3	Y
59	1KS18EC059	PAVAN KUMAR P	28	17	3	Y	11	3	Y	8	5	3	Y	3	3	Y	21	4	3	Y	12	3	Y	5	3	Y	8	1	1	N	6	3	Y	1	1	N	7	4	0	N	3	0	N	8	5	3	Y	3	3	Y	**21**	0	N
60	1KS18EC060	POOJA S	27	17	3	Y	10	3	Y	10	6	3	Y	4	3	Y	25	5	3	Y	14	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	17	7	2	N	10	2	N	10	6	3	Y	4	3	Y	**35**	3	Y
61	1KS18EC061	PRAKRUTHI S H	26	14	3	Y	12	3	Y	10	6	3	Y	4	3	Y	28	5	3	Y	17	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	18	8	3	Y	10	2	N	10	6	3	Y	4	3	Y	**45**	1	N
62	1KS18EC063	PUNEETH M	27	16	3	Y	11	3	Y	10	6	3	Y	4	3	Y	24	5	3	Y	13	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	10	6	1	N	4	0	N	10	6	3	Y	4	3	Y	**32**	2	N
63	1KS18EC064	PURUSHOTHAM V R	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	26	5	3	Y	15	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	18	5	0	N	13	3	Y	10	6	3	Y	4	3	Y	**35**	3	Y
64	1KS18EC066	RAGHAVENDRA.K.P	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	27	5	3	Y	16	3	Y	6	3	Y	8	1	1	N	6	3	Y	1	1	N	27	11	3	Y	16	3	Y	8	5	3	Y	3	3	Y	**44**	3	Y
65	1KS18EC067	RAGHU B T	27	17	3	Y	10	3	Y	10	6	3	Y	4	3	Y	26	5	3	Y	15	3	Y	6	3	Y	9	1	1	N	6	3	Y	2	3	Y	25	10	3	Y	15	3	Y	9	6	3	Y	3	3	Y	**36**	3	Y
66	1KS18EC068	RAJ KRISHNA	28	16	3	Y	12	3	Y	10	6	3	Y	4	3	Y	25	5	3	Y	14	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	4	2	0	N	2	0	N	10	6	3	Y	4	3	Y	**44**	0	N
67	1KS18EC069	RAJATH S BHUSHAN	19	8	0	N	11	3	Y	10	6	3	Y	4	3	Y	20	3	1	N	14	3	Y	3	1	N	10	2	3	Y	6	3	Y	2	3	Y	7	3	0	N	4	0	N	10	6	3	Y	4	3	Y	**13**	0	N
68	1KS18EC070	RAM BAHADUR MAHARA	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	23	4	3	Y	14	3	Y	5	3	Y	9	1	1	N	6	3	Y	2	3	Y	13	4	0	N	9	0	N	9	6	3	Y	3	3	Y	**27**	0	N
69	1KS18EC071	RASETTY SANDEEP	28	17	3	Y	11	3	Y	6	4	3	Y	2	1	N	22	3	1	N	13	3	Y	6	3	Y	0	0	0	N	0	0	N	0	0	N	1	1	0	N	0	0	N	0	0	0	N	0	0	N	**22**	2	N
70	1KS18EC073	RITHVIK P	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	23	4	3	Y	14	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	25	10	3	Y	15	3	Y	10	6	3	Y	4	3	Y	**34**	1	N
71	1KS18EC074	S MANOJ	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	23	5	3	Y	14	3	Y	4	3	Y	10	2	3	Y	6	3	Y	2	3	Y	25	10	3	Y	15	3	Y	10	6	3	Y	4	3	Y	**32**	0	N
72	1KS18EC075	S RAHUL	28	16	3	Y	12	3	Y	10	6	3	Y	4	3	Y	26	5	3	Y	15	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	13	5	0	N	8	0	N	10	6	3	Y	4	3	Y	**29**	0	N
73	1KS18EC076	S TUSHAR HARINATH	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	27	5	3	Y	16	3	Y	6	3	Y	9	1	1	N	6	3	Y	2	3	Y	23	10	3	Y	13	3	Y	9	6	3	Y	3	3	Y	**29**	0	N
74	1KS18EC077	SAGAR T C	28	17	3	Y	11	3	Y	8	5	3	Y	3	3	Y	22	5	3	Y	13	3	Y	4	3	Y	10	2	3	Y	6	3	Y	2	3	Y	14	8	3	Y	6	0	N	10	6	3	Y	4	3	Y	**25**	1	N
75	1KS18EC078	SANJANA B	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	28	5	3	Y	17	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	16	11	3	Y	5	0	N	10	6	3	Y	4	3	Y	**30**	0	N
76	1KS18EC079	SANKET B PASCHAPURI	26	15	3	Y	11	3	Y	10	6	3	Y	4	3	Y	24	4	3	Y	14	3	Y	6	3	Y	9	1	1	N	6	3	Y	2	3	Y	3	3	0	N	0	0	N	9	6	3	Y	3	3	Y	**26**	0	N
77	1KS18EC080	SHASHANK H K	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	25	5	3	Y	14	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	6	4	0	N	2	0	N	10	6	3	Y	4	3	Y	**27**	3	Y
78	1KS18EC081	SHEETAL N GOWDA	27	16	3	Y	11	3	Y	10	6	3	Y	4	3	Y	25	5	3	Y	14	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	20	9	3	Y	11	3	Y	10	6	3	Y	4	3	Y	**36**	3	Y
79	1KS18EC082	SHIVA SHANKAR.B	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	24	5	3	Y	14	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	17	6	1	N	11	3	Y	10	6	3	Y	4	3	Y	**38**	0	N
80	1KS18EC083	SHREYA V DEV	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	23	5	3	Y	14	3	Y	4	3	Y	10	2	3	Y	6	3	Y	2	3	Y	20	5	0	N	15	3	Y	10	6	3	Y	4	3	Y	**25**	2	N
81	1KS18EC084	SHREYAS C	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	23	5	3	Y	14	3	Y	4	3	Y	10	2	3	Y	6	3	Y	2	3	Y	16	6	1	N	10	2	N	10	6	3	Y	4	3	Y	**35**	3	Y
82	1KS18EC085	SHREYAS D R	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	23	4	3	Y	14	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	26	10	3	Y	16	3	Y	10	6	3	Y	4	3	Y	**36**	0	N
83	1KS18EC086	SHRIKANTH C K	17	12	3	Y	5	0	N	6	4	3	Y	2	1	N	24	5	3	Y	13	3	Y	6	3	Y	6	0	0	N	6	3	Y	0	0	N	1	1	0	N	0	0	N	6	4	3	Y	2	0	N	**28**	0	N
84	1KS18EC087	SIRI RAVINATH	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	23	5	3	Y	13	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	7	6	1	N	1	0	N	10	6	3	Y	4	3	Y	**21**	3	Y
85	1KS18EC088	SIRISHA.M	30	18	3	Y	12	3	Y	10	6	3	Y	4	3	Y	25	6	3	Y	17	3	Y	2	0	N	10	2	3	Y	6	3	Y	2	3	Y	14	12	3	Y	2	0	N	10	6	3	Y	4	3	Y	**43**	3	Y
86	1KS18EC090	SOMASHEKAR M	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	27	5	3	Y	16	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	25	9	3	Y	16	3	Y	10	6	3	Y	4	3	Y	**41**	3	Y
87	1KS18EC091	SUDHEER B	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	22	4	3	Y	15	3	Y	3	1	N	10	2	3	Y	6	3	Y	2	3	Y	16	6	1	N	10	2	N	10	6	3	Y	4	3	Y	**38**	0	N
88	1KS18EC092	SUJAY R	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	27	4	3	Y	17	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	23	9	3	Y	14	3	Y	10	6	3	Y	4	3	Y	**29**	2	N
89	1KS18EC093	SUPRIYA S	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	21	5	3	Y	11	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	27	10	3	Y	17	3	Y	10	6	3	Y	4	3	Y	**33**	3	Y
90	1KS18EC094	SURAJ V GHORPADE	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	26	5	3	Y	16	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	23	9	3	Y	14	3	Y	10	6	3	Y	4	3	Y	**48**	1	N
91	1KS18EC095	SUSHMA.A.V	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	28	5	3	Y	17	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	8	5	0	N	3	0	N	10	6	3	Y	4	3	Y	**32**	1	N
92	1KS18EC096	SUSHMITHA R	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	21	4	3	Y	14	3	Y	3	1	N	10	2	3	Y	6	3	Y	2	3	Y	26	10	3	Y	16	3	Y	10	6	3	Y	4	3	Y	**31**	0	N
93	1KS18EC097	THANUSH R S	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	25	5	3	Y	14	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	13	5	0	N	8	0	N	10	6	3	Y	4	3	Y	**29**	1	N
94	1KS18EC098	THANUSHREE D	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	24	5	3	Y	15	3	Y	4	3	Y	10	2	3	Y	6	3	Y	2	3	Y	23	12	3	Y	11	3	Y	10	6	3	Y	4	3	Y	**32**	1	N
95	1KS18EC099	VAISHNAVI G	28	16	3	Y	12	3	Y	10	6	3	Y	4	3	Y	25	5	3	Y	15	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	14	10	3	Y	4	0	N	10	6	3	Y	4	3	Y	**32**	0	N
96	1KS18EC100	VAKKALA GADDA ANIL	22	16	3	Y	6	1	N	5	3	1	N	2	1	N	23	5	3	Y	13	3	Y	5	3	Y	5	0	0	N	5	3	Y	0	0	N	AB	0	0	N	0	0	N	5	3	1	N	2	0	N	**24**	1	N
97	1KS18EC101	VANDANA K	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	25	5	3	Y	14	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	25	10	3	Y	15	3	Y	10	6	3	Y	4	3	Y	**30**	0	N
98	1KS18EC102	VARSHINI.B.M	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	27	5	3	Y	16	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	7	6	1	N	1	0	N	10	6	3	Y	4	3	Y	**27**	0	N
99	1KS18EC103	VASANTH PAI.M	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	24	5	3	Y	13	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	6	4	0	N	2	0	N	10	6	3	Y	4	3	Y	**21**	1	N
100	1KS18EC104	VIJAY BABU K	28	16	3	Y	12	3	Y	10	6	3	Y	4	3	Y	26	5	3	Y	15	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	16	11	3	Y	5	0	N	10	6	3	Y	4	3	Y	**30**	0	N
101	1KS18EC105	VINAY K	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	25	5	3	Y	14	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	15	4	0	N	11	3	Y	10	6	3	Y	4	3	Y	**21**	0	N
102	1KS18EC106	VINAY S	28	16	3	Y	12	3	Y	10	6	3	Y	4	3	Y	26	5	3	Y	15	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	7	6	1	N	1	0	N	10	6	3	Y	4	3	Y	**26**	1	N
103	1KS18EC108	VISHAL MADHUSUDAN	28	17	3	Y	11	3	Y	10	6	3	Y	4	3	Y	28	5	3	Y	17	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	6	5	0	N	1	0	N	10	6	3	Y	4	3	Y	**30**	0	N
104	1KS18EC109	VISHWAS P	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	25	5	3	Y	14	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	9	5	0	N	4	0	N	10	6	3	Y	4	3	Y	**25**	1	N
105	1KS18EC110	VIVEKGOWDA J	26	16	3	Y	10	3	Y	10	6	3	Y	4	3	Y	26	5	3	Y	15	3	Y	6	3	Y	10	2	3	Y	6	3	Y	2	3	Y	14	8	3	Y	6	0	N	10	6	3	Y	4	3	Y	**32**	3	Y
106	1KS18EC111	VRINDHA SHAM BHATT	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	24	5	3	Y	14	3	Y	5	3	Y	10	2	3	Y	6	3	Y	2	3	Y	24	10	3	Y	14	3	Y	10	6	3	Y	4	3	Y	**45**	0	N
107	1KS19EC400	HEMANTHA V	23	13	3	Y	10	3	Y	10	6	3	Y	4	3	Y	15	0	0	N	13	3	Y	2	0	N	10	2	3	Y	6	3	Y	2	3	Y	9	4	0	N	5	0	N	10	6	3	Y	4	3	Y	**24**	0	N
108	1KS19EC401	KARTHIK B P	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	15	0	0	N	14	3	Y	1	0	N	10	2	3	Y	6	3	Y	2	3	Y	19	6	1	N	13	3	Y	10	6	3	Y	4	3	Y	**21**	0	N
109	1KS19EC402	KRISHNAPRASAD B	24	13	3	Y	11	3	Y	10	6	3	Y	4	3	Y	15	0	0	N	13	3	Y	2	0	N	10	2	3	Y	6	3	Y	2	3	Y	19	2	0	N	17	3	Y	10	6	3	Y	4	3	Y	**29**	1	N
110	1KS19EC403	NAVEEN G	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	13	0	0	N	13	3	Y	0	0	N	8	1	1	N	6	3	Y	1	1	N	19	7	2	N	12	3	Y	8	5	3	Y	3	3	Y	**31**	0	N
111	1KS19EC405	PRUTHVI DINESH	26	15	3	Y	11	3	Y	10	6	3	Y	4	3	Y	23	5	3	Y	15	3	Y	3	1	N	8	1	1	N	6	3	Y	1	1	N	3	2	0	N	1	0	N	8	5	3	Y	3	3	Y	**23**	0	N

112	1KS19EC406	RAGHOTHAM C G	25	13	3	Y	12	3	Y	10	6	3	Y	4	3	Y	21	5	3	Y	13	3	Y	3	1	N	10	2	3	Y	6	3	Y	2	3	Y	11	4	0	N	7	0	N	10	6	3	Y	4	3	Y	**23**	0	N
113	1KS19EC407	SADANA M	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	22	4	3	Y	14	3	Y	4	3	Y	10	2	3	Y	6	3	Y	2	3	Y	8	5	0	N	3	0	N	10	6	3	Y	4	3	Y	**13**	0	N
114	1KS19EC408	SINDHU G	29	17	3	Y	12	3	Y	10	6	3	Y	4	3	Y	23	5	3	Y	16	3	Y	2	0	N	10	2	3	Y	6	3	Y	2	3	Y	16	8	3	Y	8	0	N	10	6	3	Y	4	3	Y	**14**	0	N
115	1KS19EC409	VARSHA M S	28	16	3	Y	12	3	Y	10	6	3	Y	4	3	Y	21	3	1	N	14	3	Y	4	3	Y	10	2	3	Y	6	3	Y	2	3	Y	7	5	0	N	2	0	N	10	6	3	Y	4	3	Y	**24**	0	N
	CO				**CO1**			**CO2**				**CO1**			**CO2**				**CO2**			**CO3**			**CO4**				**CO2**			**CO3**			**CO4**				**CO4**			**CO5**				**CO4**			**CO5**			**SEE**	
	Number of Not Attempted(NA)					**0**			**0**				**0**			**0**				**0**			**0**			**0**				**0**			**0**			**0**				**0**			**0**				**0**			**0**			**0**
	Score index & No of Y's				**3.0**	**114**		**3.0**	**113**			**3.0**	**114**		**2.9**	**112**			**2.77**	**72**		**3.0**	**115**			**101**			**2.7**	**98**		**2.95**	**113**		**2.8**	**105**			**1.9**	**62**		**1.5**	**54**			**2.93**	**112**		**2.9**	**111**		**1.22**	**34**
	No. of N's					**1**			**2**				**1**			**3**				**6**			**0**			**14**				**17**			**2**			**10**				**53**			**61**				**3**			**4**			**81**
	CO Attainment					**99**			**98**				**99**			**97**				**92**			**100**			**88**				**85**			**98.3**			**91**				**54**			**47**				**97.4**			**97**			**30**
	Level					**3**			**3**				**3**			**3**				**3**			**3**							**3**			**3**							**1**			**0**				**3**			**3**			**0**

CO	CIE	SEE	Direct Attainment	Level	Indirect Attainment	Final Att
CO1	99.13	29.6	64.35	3.0	3.00	3.0
CO2	93.29	29.6	61.43	3.0	3.00	3.0
CO3	99.13	29.6	64.35	3.0	3.00	3.0
CO4	82.61	29.6	56.09	2.0	3.00	2.1
CO5	71.74	29.6	50.65	1.0	3.00	1.2
AVERAGE						2.5

CO	Score index out of 3
CO1	2.10
CO2	2.32
CO3	2.11
CO4	1.53
CO5	1.37

Co-Po Mapping Table

CO'S	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	2	_	_	_	_	_	_	_	_	_	_	_	2	_
CO2	3	3	3	_	_	_	_	_	_	_	_	1	2	3
CO3	3	3	1	_	_	_	_	_	_	_	_	1	2	3
CO4	3	3	1	_	_	_	_	_	_	_	_	1	2	3
CO5	3	3	1	_	_	_	_	_	_	_	_	1	2	_
AVG	2.80	3.0	1.5	_	_	_	_	_	_	_	_	1.0	2.0	3.0

PO Attainment

CO'S	CO Attainment	CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3.00	Y	2.0	_	_	_	_	_	_	_	_	_	_	_	2.0	_
CO2	3.00	Y	3.0	3.0	3.0	_	_	_	_	_	_	_	_	1.0	2.0	3.0
CO3	3.00	Y	3.0	3.0	1.0	_	_	_	_	_	_	_	_	1.0	2.0	3.0
CO4	2.10	Y	2.1	2.1	0.7	_	_	_	_	_	_	_	_	0.7	1.4	2.1
CO5	1.20	N	1.2	1.2	0.4	_	_	_	_	_	_	_	_	0.4	0.8	_
Average			**2.3**	**2.3**	**1.3**	_	_	_	_	_	_	_	_	**0.8**	**1.6**	**2.7**

VERILOG HDL-18EC56

IA Marks:	40
Exam Marks:	60
Exam Hours:	03
Credits:	03

Text Book:

Samir Palnitkar, “Verilog HDL: A guide to digital design and synthesis”, Pearson Education, Second Edition.

Note: Most of the Images are taken from this text book

Course Learning Objectives:

- Learn different Verilog HDL constructs.
- Familiarize the different levels of abstraction in Verilog.
- Understand Verilog Tasks, Functions and Directives.
- Understand timing and delay Simulation.
- Understand the concept of logic synthesis and its impact in verification

Course Outcomes:

- Write Verilog programs in gate, dataflow (RTL), behavioral and switch modeling levels of Abstraction.
- Design and verify the functionality of digital circuit/system using test benches.
- Identify the suitable Abstraction level for a particular digital design.
- Write the programs more effectively using Verilog tasks, functions and directives.
- Perform timing and delay Simulation
- Interpret the various constructs in logic synthesis.

Syllabus

Module 1: Overview of Digital Design with Verilog HDL, Hierarchical Modeling Concepts.

Module 2: Basic Concepts, Modules and Ports

Module 3: a) Gate-Level Modeling, b) Dataflow Modeling

Module 4: c) Behavioral Modeling, Tasks & Functions

Module 5: Useful Modeling Techniques, Logic Synthesis with Verilog

M1: Overview of Digital Design with Verilog HDL

- Evolution of CAD Design
  - First IC's were called Small Scale Integration(SSI) chips. Gate count=10 Transistors
  - Medium Scale Integration(MSI) chips. Gate count= 100 Transistors
  - Large Scale Integration(LSI) chips. Gate count= 1000 Transistors
  - Very Large Scale Integration(VLSI) chips. Gate count= 1,00,000 Transistors

- Up to LSI the ckts were tested on breadboard
- Layout was done on paper or by hand on a graphic computer terminal
- Due to complexity in LSI, Electronic Design Automation (EDA) techniques began to evolve
- Complexities in VLSI, it became impossible to verify the circuits on a breadboard
- Computer Aided Techniques became critical for verification and design of VLSI digital circuits
- Computer programs to do automatic placement and routing of circuit layouts also became popular

Evolution of Logic Simulators

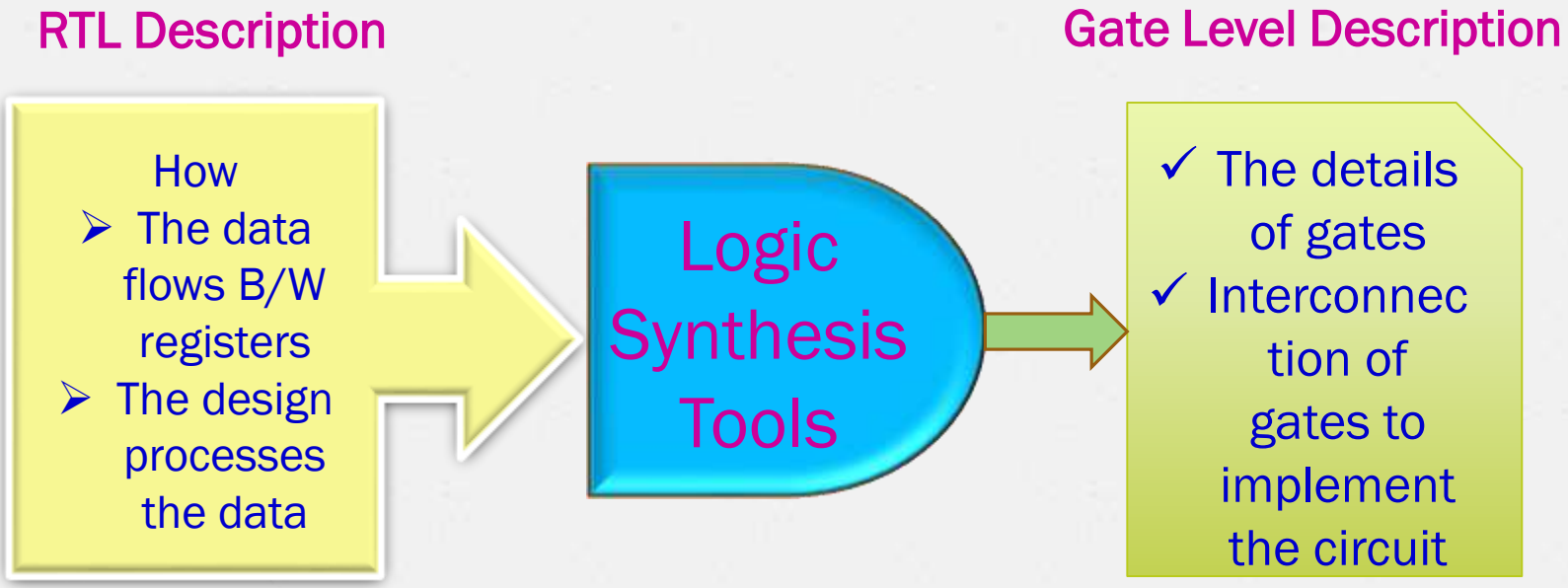
- Designers used to build small building blocks and then derive higher level blocks from them
- This process would continue until they had built the top-level block
- Logic simulators came into existence to verify the functionality of these circuits before they were fabricated on chip
- logic simulation assumed an important role in the design process.
- Designers could iron out functional bugs in the architecture before the chip was designed further.

Emergence of HDLs

- S/W programming languages: FORTRAN, Pascal, C and so on..
- H/W programming languages: Hardware Description Languages: Verilog HDL, VHDL
- Verilog HDL originated in 1983 at Gateway Design Automation
- VHDL (VHSIC-HDL, Very High Speed Integrated Circuit Hardware Description Language) was developed under contract from DARPA
- Both Verilog & VHDL simulators to simulate large digital circuits-- quickly gained acceptance from designers

Logic Synthesis Tools

- The arrival of logic synthesis in the late 1980s changed the design methodology radically
- Digital circuits could be described at a register transfer level (RTL) by use of an HDL



Usefulness of Logic Synthesis Tools

- Pushed the HDLs into the forefront of digital design
- Designers no longer had to manually place gates to build digital circuits
- Tool will implement the specified functionality in terms of gates and gate interconnections

HDLs are used…

- For system-level design
- Simulation of system boards, interconnect buses, FPGAs and PALs
- A common approach is to design each IC chip, using an HDL, and then verify system functionality via simulation.
- Today, Verilog HDL is an accepted IEEE standard

Typical Design Flow

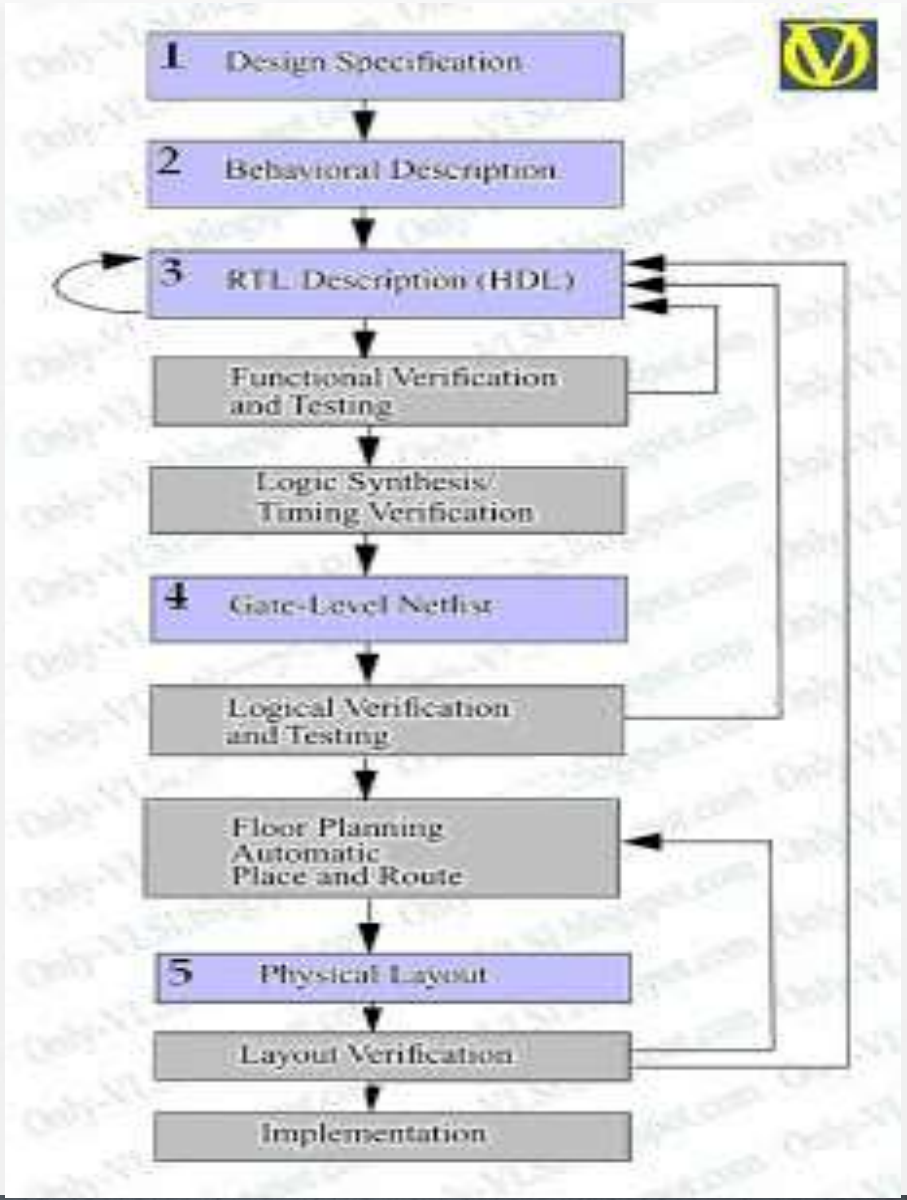


Image Source:
http://only-vlsi.blogspot.com/

Importance of HDLs

- Designs can be described at a very abstract level by use of HDLs
  - Designers can write their RTL description without choosing a specific fabrication technology
  - If a new technology emerges, designers do not need to redesign their circuit
  - The logic synthesis tool will optimize the circuit in area and timing for the new technology
- Functional verification of the design can be done early in the design cycle
- Designing with HDLs is analogous to computer programming

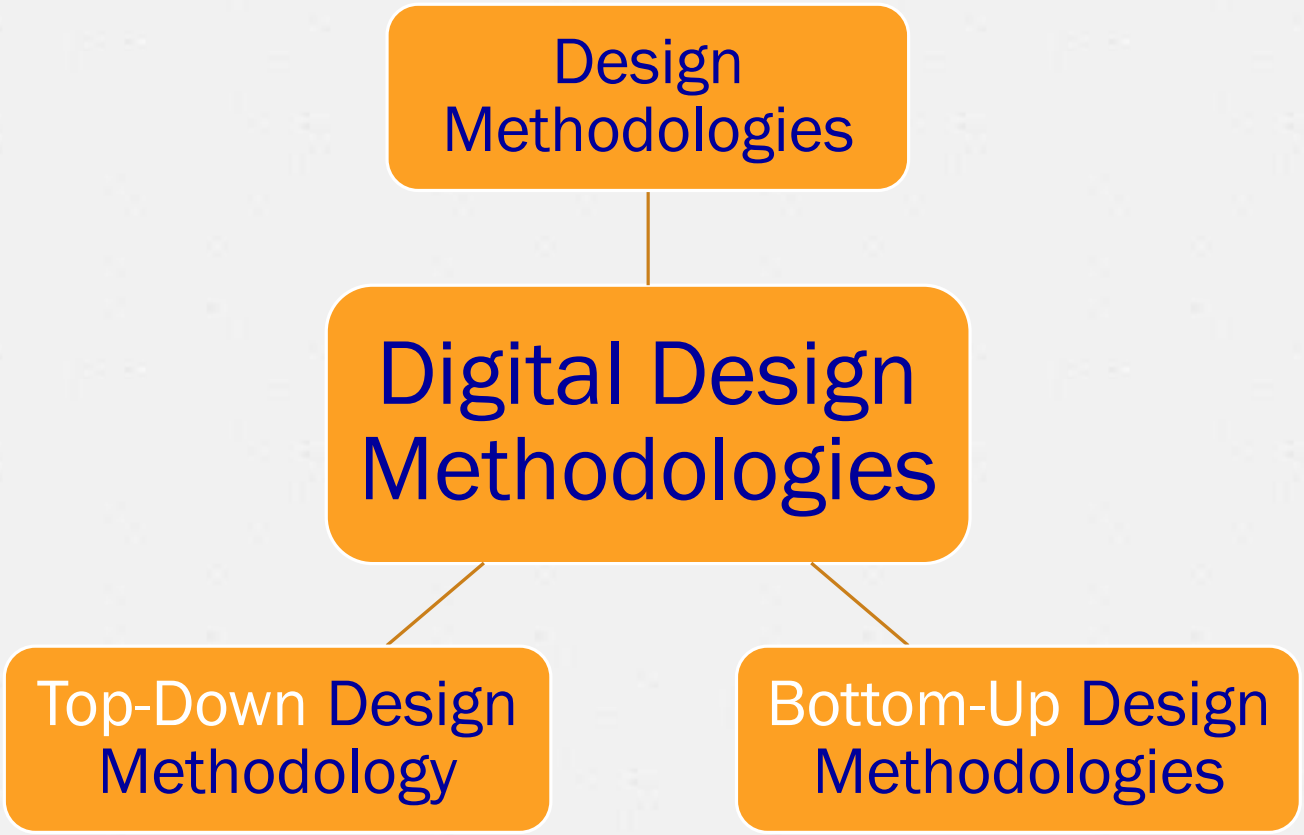
Popularity of Verilog HDL

- It is a general purpose language: easy to learn & easy to use– similar in syntax to the C
- It allows different levels of abstraction to be mixed in the same model: gate, RTL, or Behavioral design
- Most popular logic synthesis tools support Verilog HDL– making HDL popular among designers
- It allows the widest choice of vendors because all fabrication vendors provide Verilog HDL libraries for post-logic synthesis simulation
- Designers can customize a Verilog HDL simulator to their needs with the Programming Language Interface (PLI).

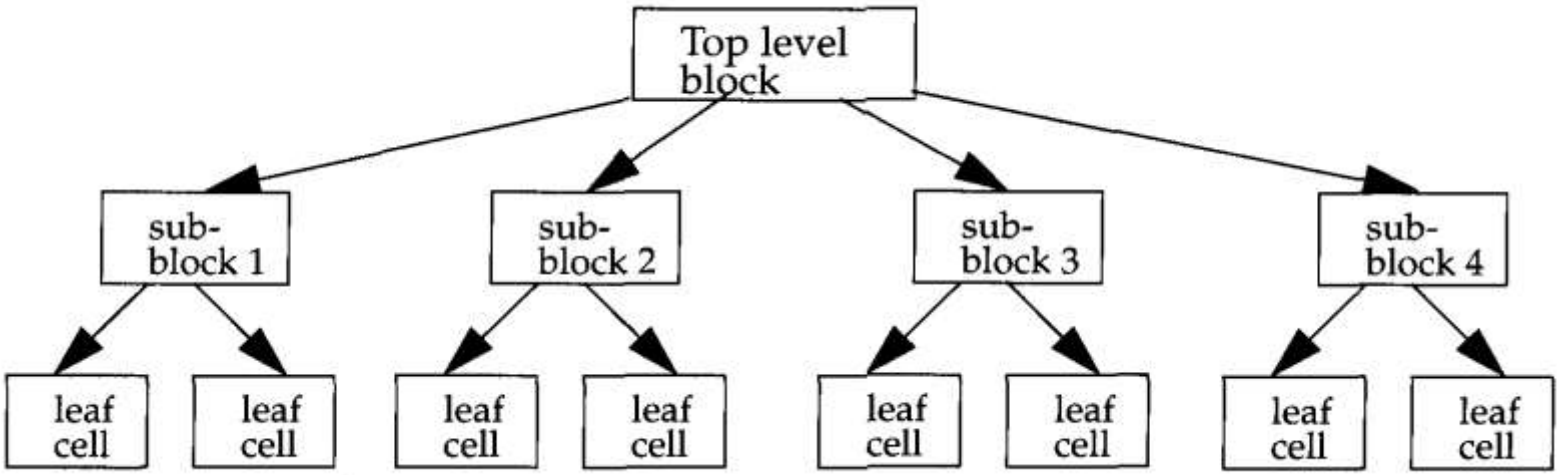
Trends in HDLs

- The speed and complexity of digital circuits has increased rapidly
- The most popular trend currently is to design in HDL at an RTL level, because logic synthesis tools can create gate-level net lists from RTL level design
- *Formal verification* techniques are also appearing on the horizon
- For very high speed and timing-critical circuits like microprocessors, designers often mix gate-level description directly into the RTL description to achieve optimum results
- A trend that is emerging for system-level design is a mixed bottom-up methodology

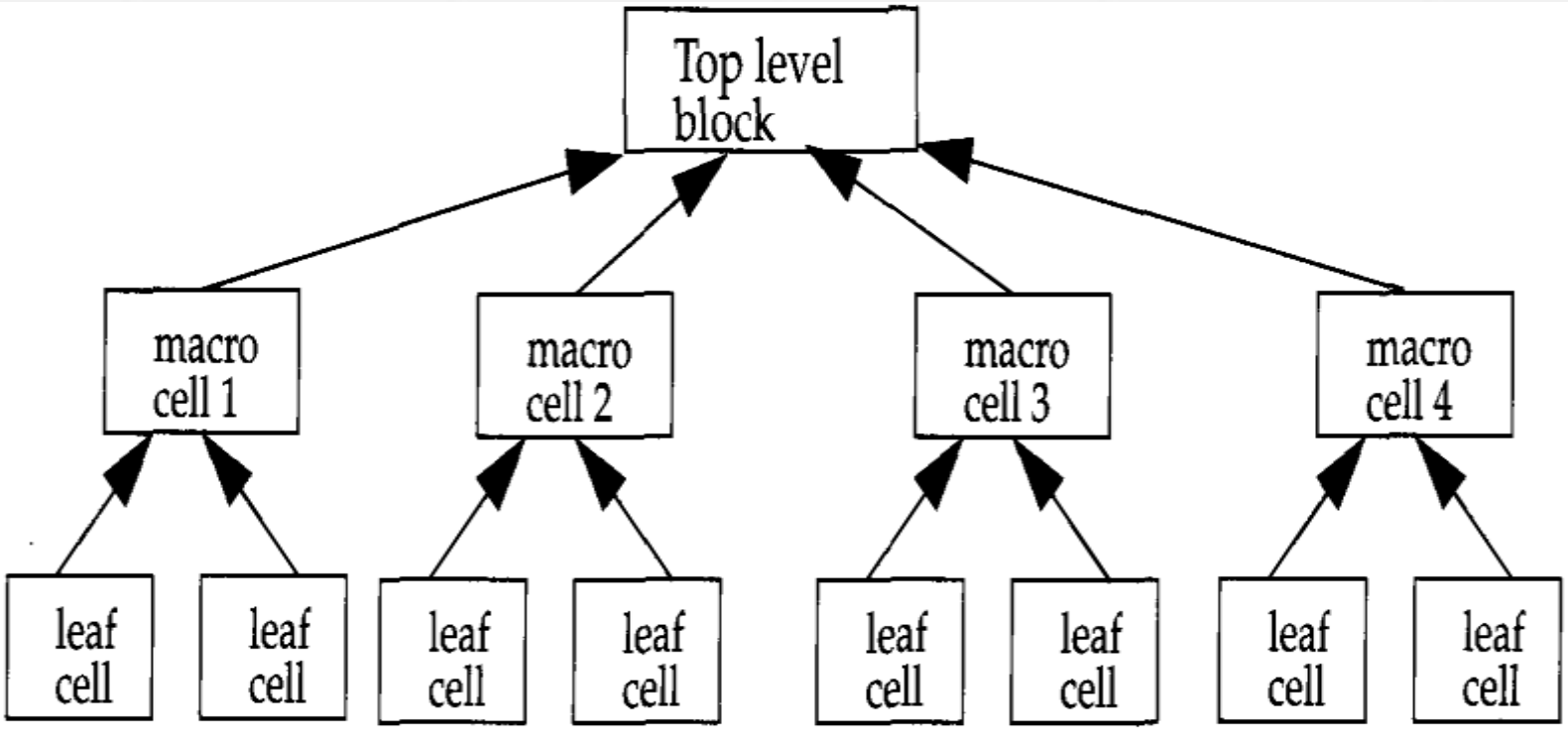
Hierarchical Modeling Concepts



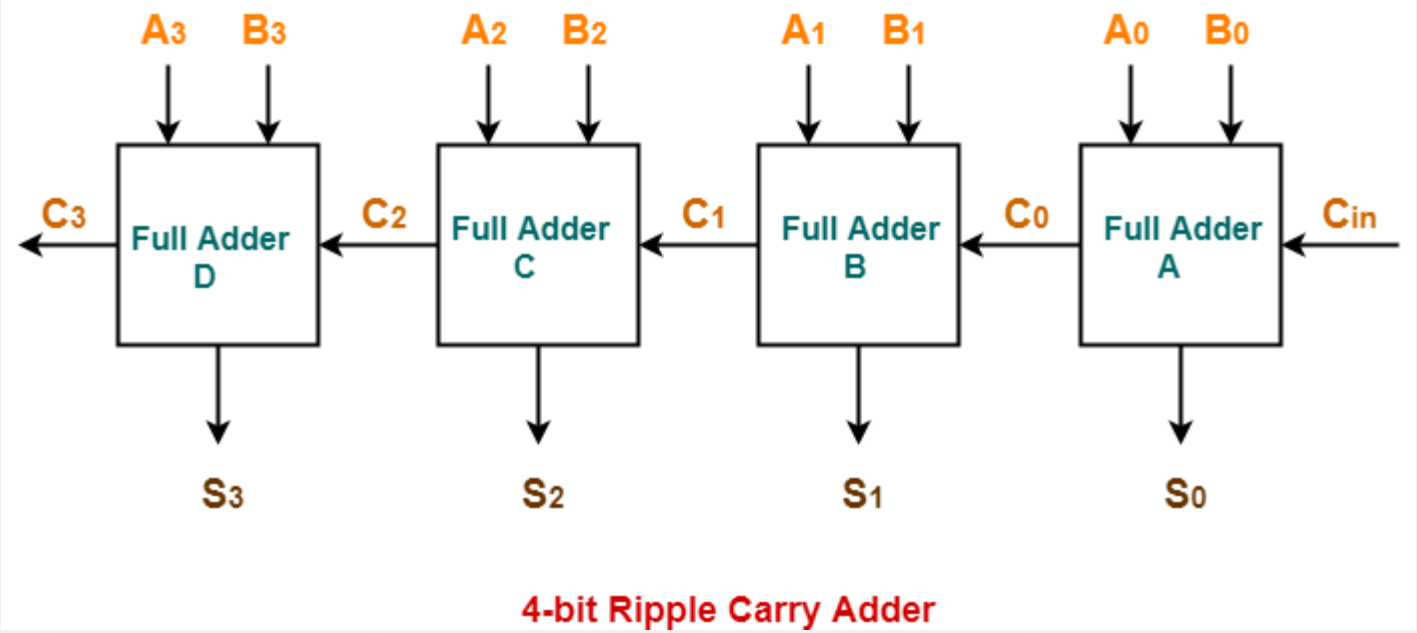
Top-Down Design Methodology



Bottom-Up Design Methodology



Example



4-bit Ripple Carry Adder

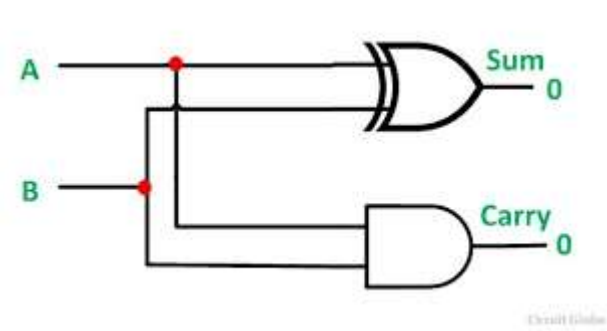


Image Source: circuitglobe.com

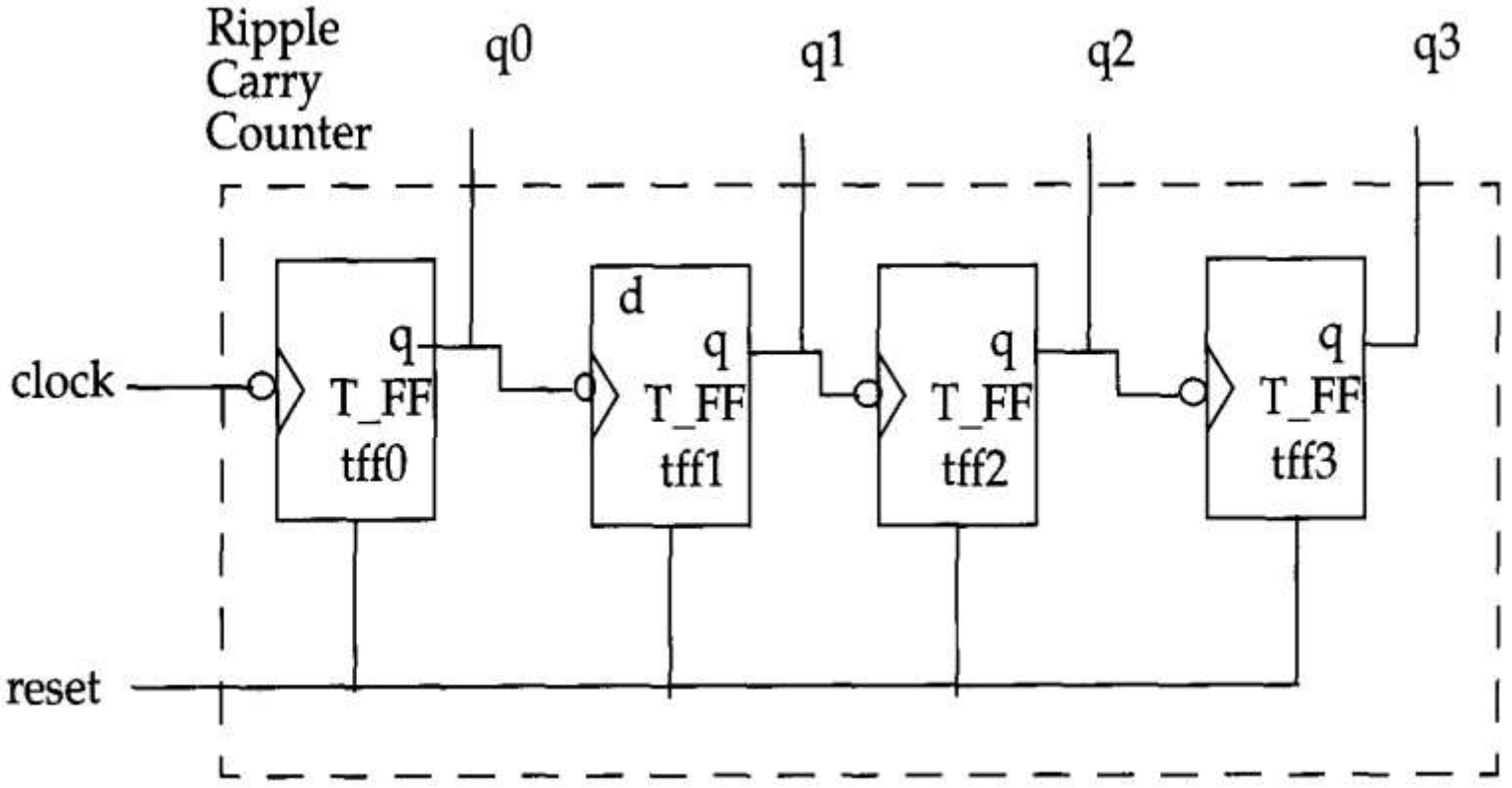
Image Source: gatevidyalay.com

Combination of top-down and bottom-up flows is used.

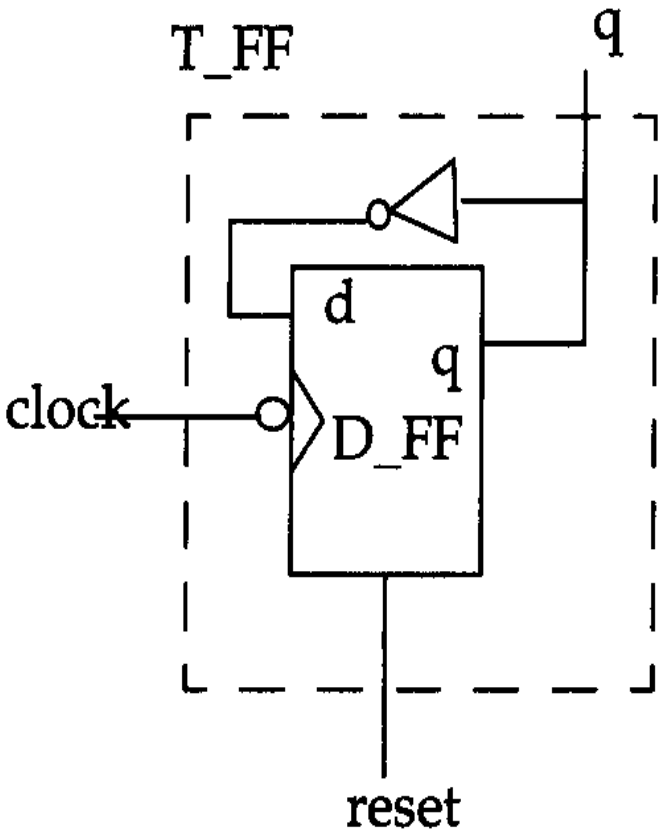
- Design architects define the specifications of the top-level block
- Logic designers decide how the design should be structured by breaking up the functionality into blocks and sub-blocks
- At the same time, circuit designers are designing optimized circuits for leaf-level cells
- They build higher-level cells by using these leaf cells
- The flow meets at an intermediate point where the switch-level circuit designers have created a library of leaf cells by using switches, and the logic level designers have designed from top-down until all modules are defined in terms of leaf cells

Illustration of hierarchical modeling concepts

Example: 4-bit Ripple Carry Counter



reset	q_n	q_{n+1}
1	1	0
1	0	0
0	0	1
0	1	0
0	0	0



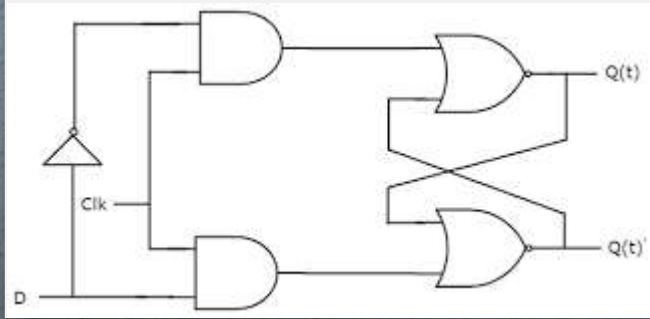
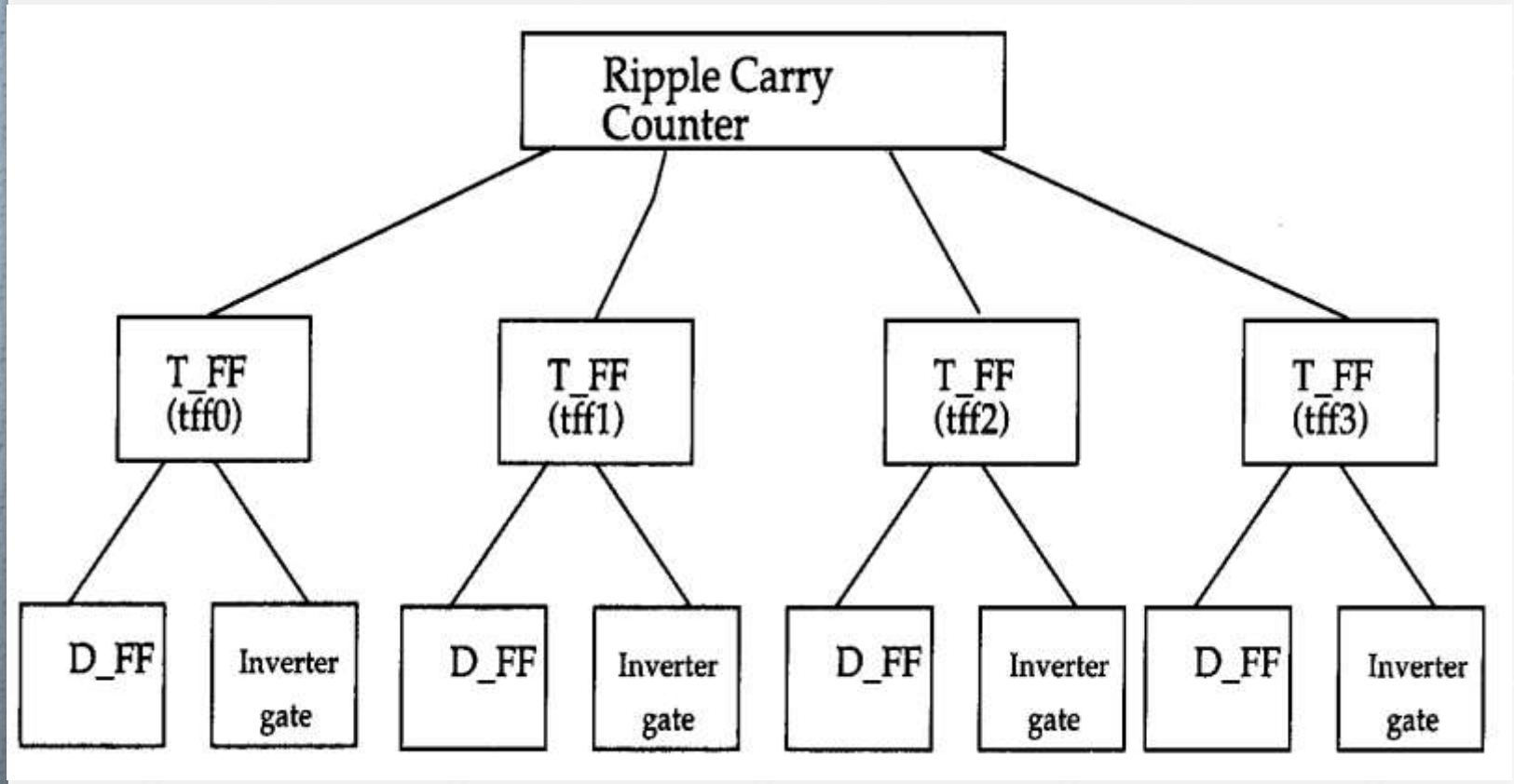


Image Source: learn.circuitverse.org

Design Hierarchical



Modules

- Is the basic building block in Verilog
- It can be an element or a collection of lower level design block
- It provides the necessary functionality to the higher level block through its port interface (i/p & o/p) but hides the internal implementation
- In Verilog, a module is declared by the key word *module*
- Each module definition will end with a keyword *endmodule*
- Each module must have a *module_name* (identifier for the module) and a *module_terminal_list* (which describes the i/p & o/p terminals)

```
module <module_name> (<module_terminal_list>);

...
<module internals>
...
...
endmodule
```

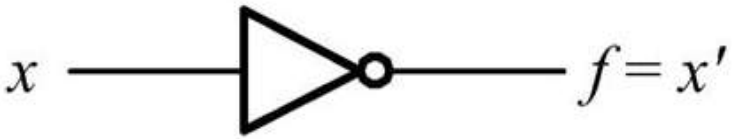
Specifically, the T-flipflop could be defined as a module as follows:

```
module T_FF (q, clock, reset);
.
.
<functionality of T-flipflop>
.
.
endmodule
```

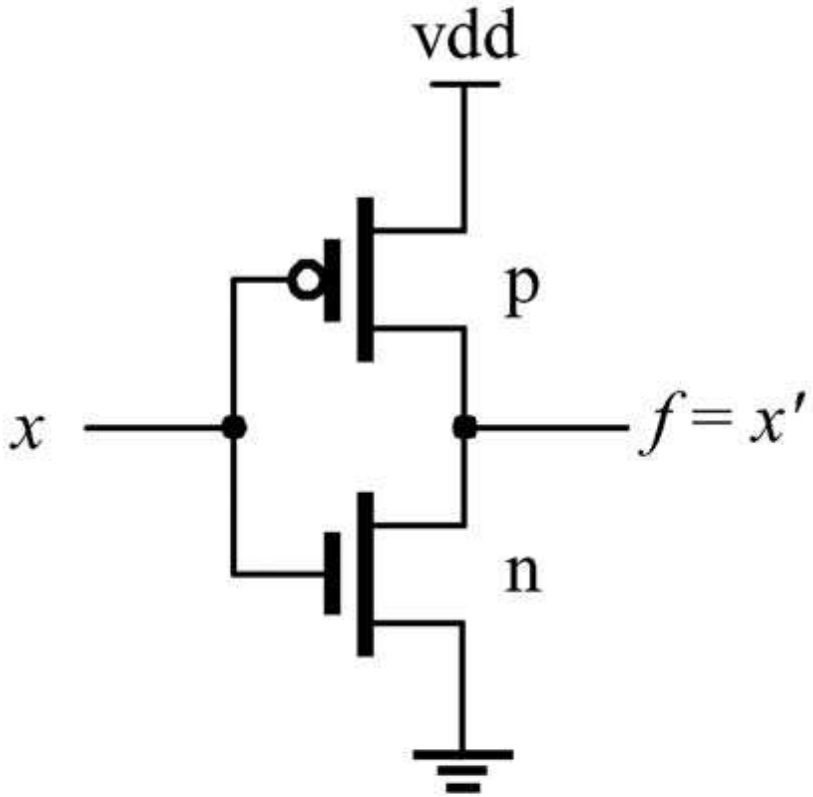
Four Levels of Abstraction

Behavioral or Algorithmic	• Highest level of Abstraction • Implemented in terms of the desired design algorithm without concern for the H/W implementation details
Dataflow	• The designer should be aware of how data flows b/w H/W registers • And how the data is processed in the design
Gate Level	• Implemented in terms of logic gates and interconnection between these gates • Similar to describing a design in terms of a gate level logic diagram
Switch level	• Lowest level of abstraction provided by Verilog • Implemented in terms of switches, storage nodes and the interconnection between them

Inverter Circuit (NOT Gate)



(b) Logic symbol



(a) Circuit

https://www.researchgate.net/

Verilog Code in Four Abstraction Levels for NOT Gate

```
module not_behavioral (y, x);
input  x;
output y;
reg  y;
always@(x)
begin
if(x==1'b0)
y<=1'b1;
else
y<=1'b0;
end
endmodule
```

```
module not_dataflow (y, x);
input x;
output y;
assign y=~x;
endmodule
```

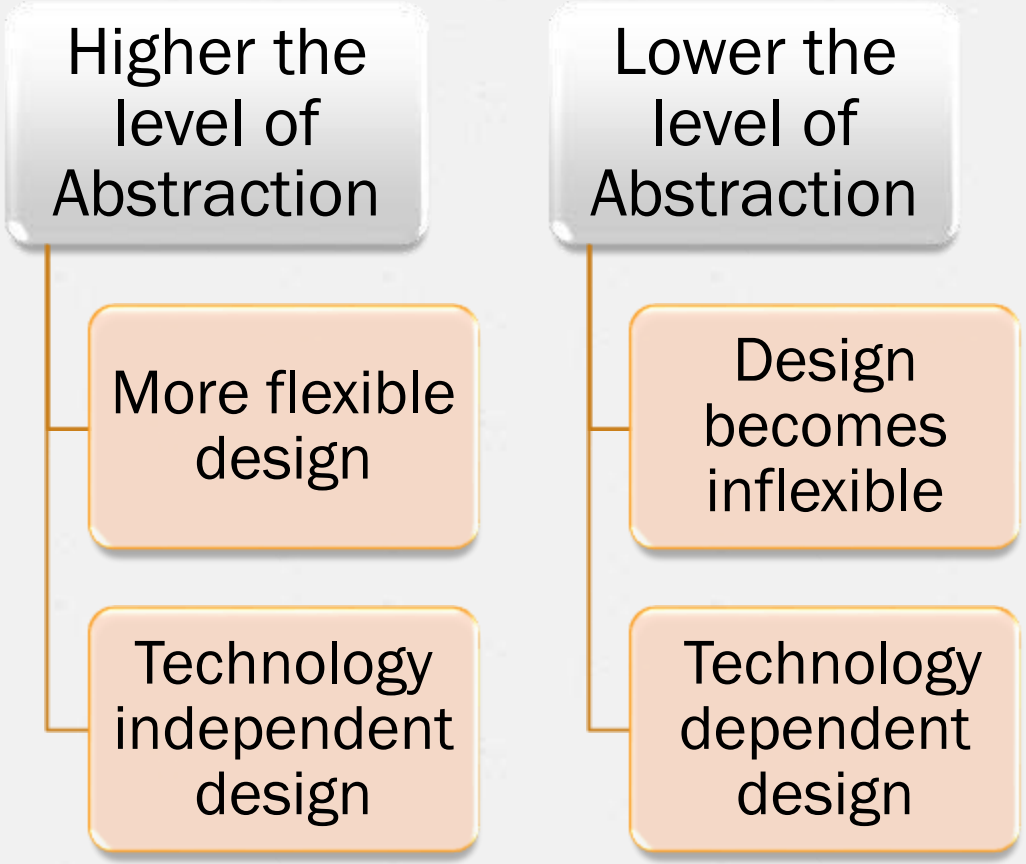
```
module not_gate (y, x);
input x;
output y;
not n1(y, x);
endmodule
```

```
module not_switch (y, x);
input x;
output y;
supply1 vdd;
supply0 gnd;
pmos p1 (y, vdd, x);
nmos n1 (y, gnd, x);
endmodule
```

Verilog is both a Behavioral and a Structural Language

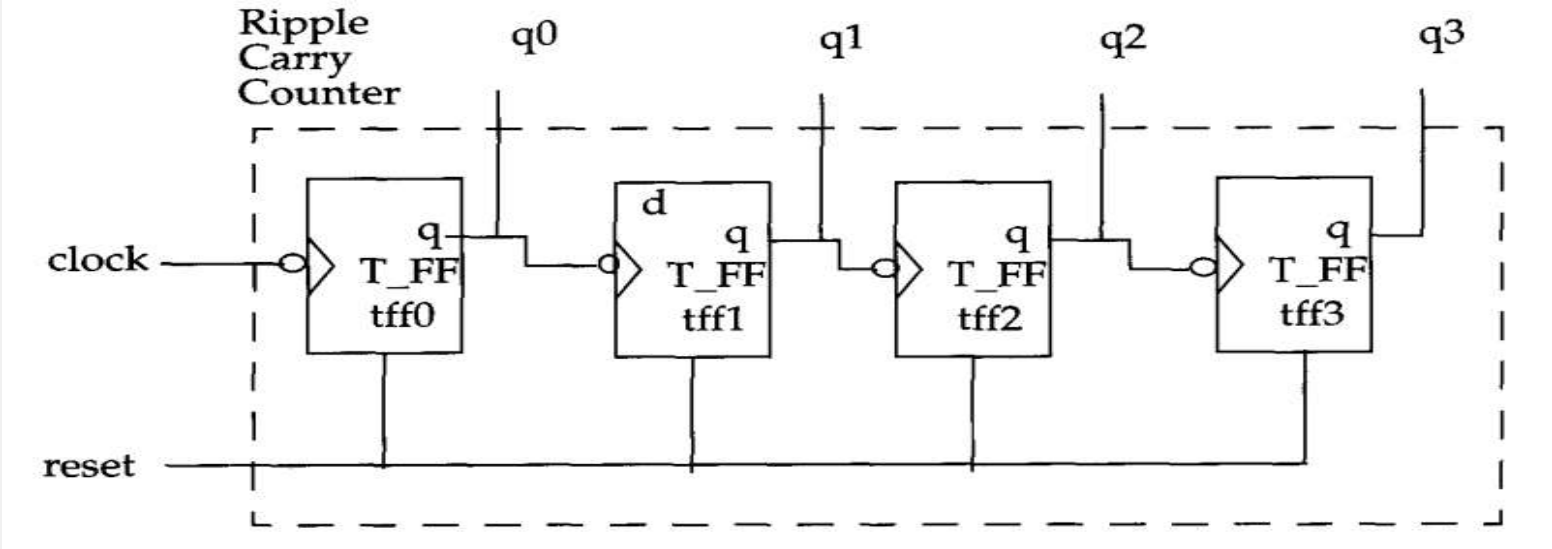
- Internals of each module can be defined at *four* levels of abstraction, depending on the needs of the design
- The module behaves identically with the external environment irrespective of the level of abstraction at which the module is described
- The internals of the module are hidden from the environment
- Thus, the level of abstraction to describe a module can be changed without any change in the environment

Comparison b/w Higher Level & Lower Level Abstraction



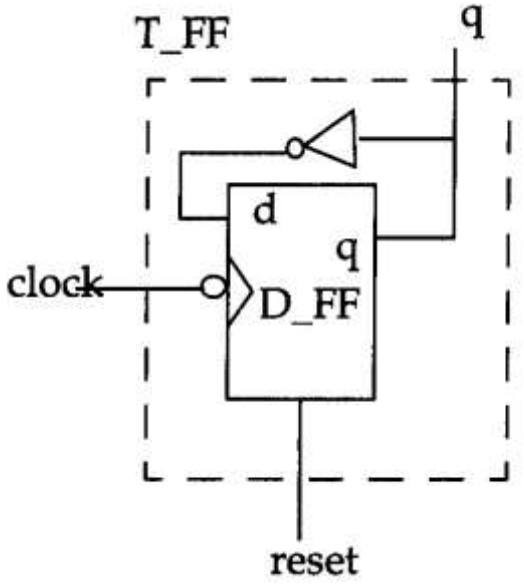
Instances

- The process of creating objects from a module template is called *instantiation*
- And the objects are called *instances*
- Each *instances* must be given a unique name
- Example: Ripple Carry Counter



```
module ripple_carry_counter(q, clk, reset);
output [3:0] q;
input clk, reset;
T-FF tff0 (q[0], clk, reset); // 4 instances
T-FF tff1 (q[1], q[0], reset);// unique names
T-FF tff2 (q[2] ,q[1] , reset) ;
T-FF tff3(q[3] ,q[2], reset) ;
endmodule
```

```
module T_FF(q, clk, reset) ;
output q;
input clk, reset;
wire d;
D-FF dff0 (q, d, clk, reset) ; // Instantiate
not n1(d, q); // not gate is a Verilog primitive
endmodule
```



Nest Modules are illegal in Verilog

- One module definition cannot contain another module definition within the **module** and **endmodule** statements
- Instead, a module definition can incorporate copies of other modules by instantiating them
- Module definitions simply specify how the module will work, its internals, and its interface
- Modules must be instantiated for use in the design

Illegal Module Nesting

An Example:

```
module ripple-carry-counter(q, clk, reset);
output [3:0] q;
input clk, reset;
module T-FF (q, clock, reset) ; // ILLEGAL MODULE NESTING
...
<module T-FF internals>
...
endmodule // END OF ILLEGAL MODULE NESTING
endmodule
```

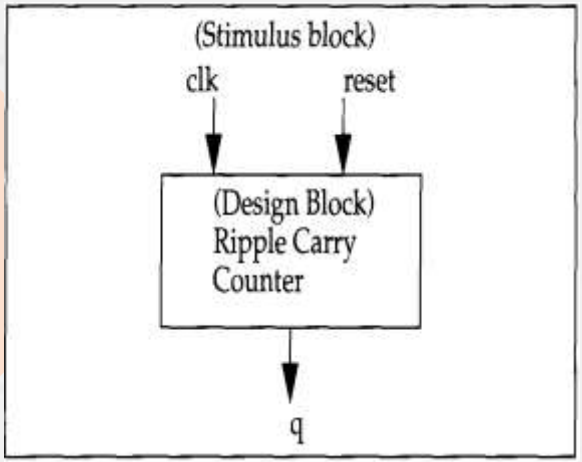
Components of a Simulation

- Once a design block is completed, it must be tested
- The functionality of the design block can be tested by applying stimulus and checking results
- such a block is known as the *stimulus* block.
- It is good practice to keep the stimulus and design blocks separate.
- The stimulus block can be written in Verilog
- A separate language is not required to describe stimulus
- The stimulus block is also commonly called a *test bench*
- Different test benches can be used to thoroughly test the design block

Two Styles of Stimulus Application

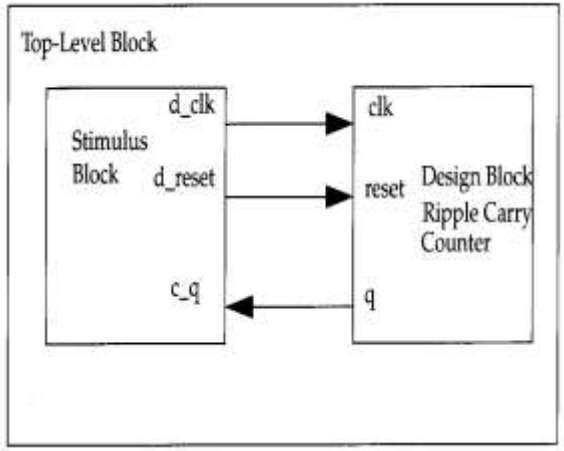
1st Style: Stimulus Block Instantiates the Design Block

- Stimulus block drives the signals in the design block
- Stimulus block becomes the top-level block
- Stimulus manipulates signal clk & reset, and it checks and displays output signal q



2nd Style: Stimulus and Design Blocks Instantiated in a Dummy Top-Level Module

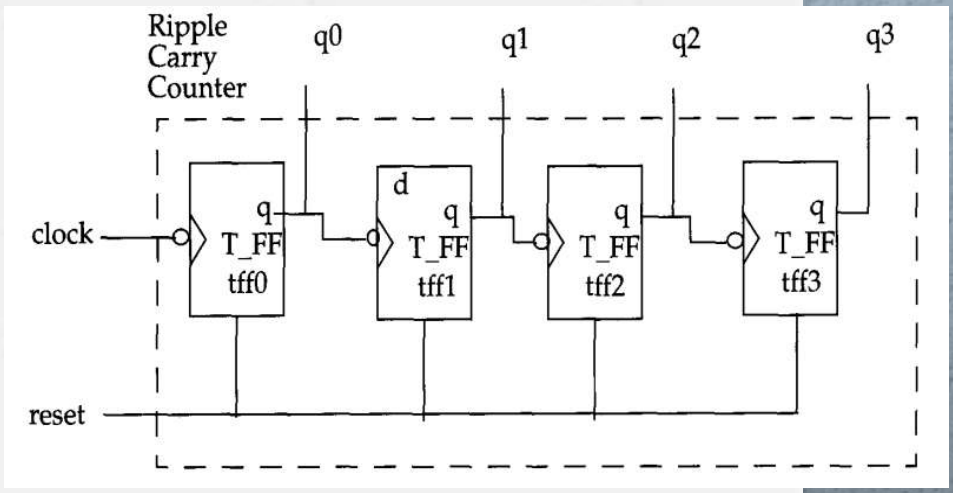
- Stimulus block interacts with the design block only through the interface
- Stimulus module drives the signals d-clk and d-reset, which are connected to the signals clk and reset in the design block
- Stimulus also checks and displays signal c-q, which is connected to the signal q in the design block
- The function of top-level block is simply to instantiate the design and stimulus blocks



Example: Ripple Carry Counter

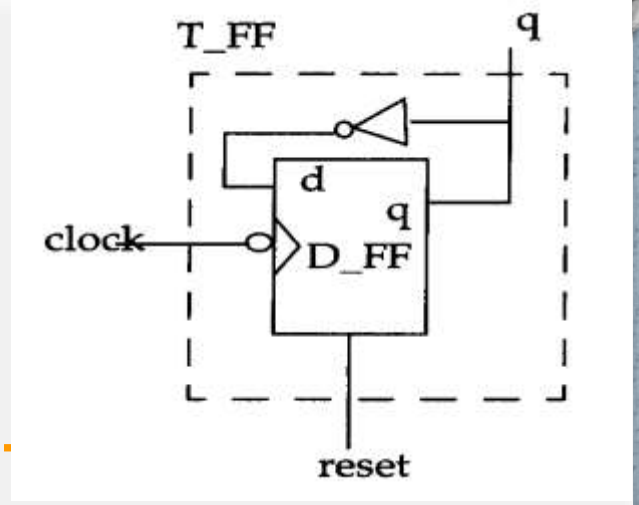
- **Design Block:**

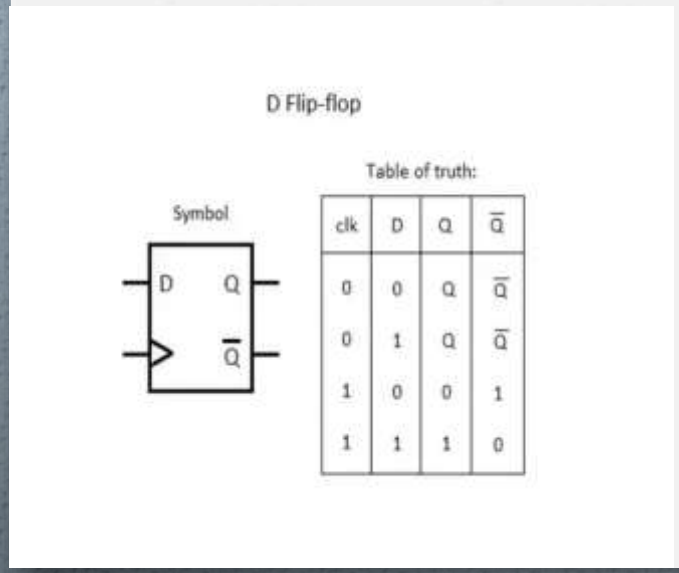
```
module ripple_carry_counter(q, clk, reset) ;
output [3:0] q;
input clk, reset;
T_FF tff0 (q[0], clk, reset);
T_FF tff1 (q[1], q[0], reset);
T_FF tff2 (q[2], q[l], reset);
T_FF tff3 (q[3], q[2], reset);
endmodule
```



```
module T-FF (q, clk, reset) ;
output q;
input clk, reset;
wire d;
D-FF dff0 (q, d, clk, reset) ;
not n1(d, q) ; /*not is a Verilog primitive.
case sensitive*/
endmodule
```

Design Block

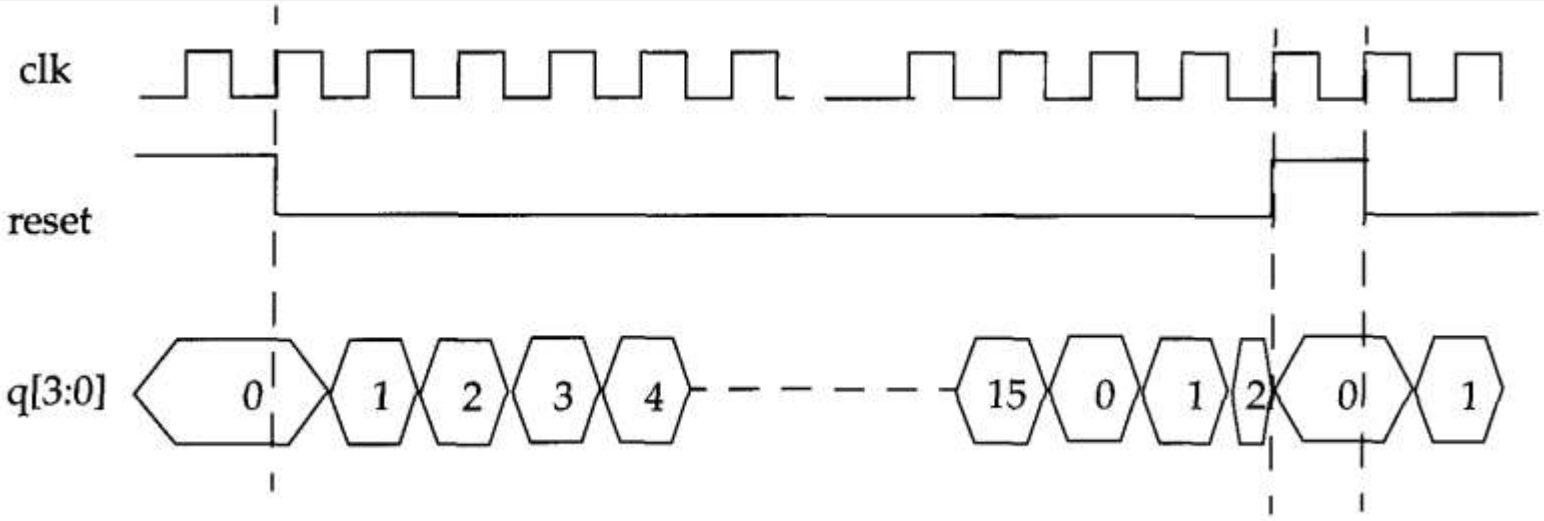





```
module D-FF(q, d, clk, reset) ;
output q;
input d, clk, reset;
reg q;
always @(posedge reset or negedge clk)
if (reset)
q <= 1'b0;
else
q <= d;
endmodule
```

Stimulus Block

- Stimulus block to check if the ripple carry counter design is functioning correctly
- Control the signals clk and *reset* so that the regular function of the ripple carry counter and the asynchronous reset mechanism are both tested
- The cycle time for clk is 10 units; the *reset* signal stays up from time 0 to 15 and then goes up again from time 195 to 205. Output q counts from 0 to 15



Stimulus and Output Waveforms

Stimulus Block

```
module stimulus;
reg clk;
reg reset;
wire[3:0] q;
ripple-carry-counter r1(q, clk, reset); // instantiate the design block
initial
clk = 1'b0; //set clk to 0
always
#5 clk = ~clk; //toggle clk every 5 time units
initial
begin
reset = 1'b1;
#15 reset = 1'b0;
#180 reset = 1'b1;
#10 reset = 1'b0;
#20 $finish; //terminate the simulation
end
// Monitor the outputs
initial
$monitor ($time, " Output q = %d" , q) ;
endmodule
```

Output of Simulation

- 0 Output q = 0
- 20 Output q = 1
- 30 Output q = 2
- 40 Output q = 3
- 50 Output q = 4
- 60 Output q = 5
- 70 Output q = 6
- 80 Output q = 7
- 90 Output q = 8
- 100 output q = 9
- 110 output q = 10
- 120 Output q = 11
- 130 Output q = 12
- 140 Output q = 13
- 150 Output q = 14
- 160 Output q = 15
- 170 Output q = 0
- 180 Output q = 1
- 190 Output q = 2
- 195 Output q = 0
- 210 Output q = 1
- 220 Output q = 2

thank you