KLS VISHWNATHRAO DESHPANDE INSTITUTE OF TECHNOLOGY,
Haliyal – 583 219, Uttara Kannada
(Accredited by NAAC with "A" Grade)

Department of Electronics and Communication Engineering

LABORATORY MANUAL

Course Title	:	**Embedded System Design**
Course Code	:	**BEC601**
Year / Semester	:	**3rd Year / 6th Semester**
Academic Year	:	**2025 – 26**
Course In-Charge	:	**Prof. Nikhil A. Kulkarni**

Student Name	
USN	
Semester / Division	
Roll No.	

College Vision and Mission Statements

Vision
To nurture talent & enrich society through excellence in technical education, research & innovation.
Mission
1. To augment innovative Pedagogy & kindle quest for interdisciplinary learning & to enhance conceptual understanding. 2. To build competence, professional ethics & develop entrepreneurial thinking. 3. To strengthen Industry Institute Partnership & explore global collaborations. 4. To inculcate culture of socially responsible citizenship. 5. To focus on Holistic & Sustainable development.

Department Vision and Mission Statements

Vision	
To bring out talented, skilled, and sustainable Electronics and Communication Engineering Graduates through strong domain expertise to serve the Society with greater Professional Ethics.	
Mission	
1. To create and impart an active learning ambience to accomplish a high degree of Professional competencies 2. To inculcate innovative research and developmental thinking in effective Teaching and Learning processes for solving Societal challenges 3. To deliver the needs and requirements of the latest state of art of the Industry through quality multidisciplinary internship and training programs	
PEOs	
PEO 1:	To be successful in professional career in electronics, communication and allied industries by acquiring the knowledge in the fundamentals of Electronics and Communication Engineering principles and professional skills.
PEO 2:	To be in a position to analyze real life problems and design socially accepted and economically feasible solutions in the respective fields.
PEO 3:	To exhibit good communication skills in their professional career, lead a team with good leadership traits and good interpersonal relationship with the members related to other engineering streams.
PEO 4:	To involve themselves in lifelong learning and professional development by pursuing higher education and participation in research and development activities.
PEO 5:	To demonstrate professional and ethical responsibilities towards their profession, society and the environment.
PSOs	
PSO 1:	An ability to use appropriate modern techniques for analysis, design and development of VLSI and Embedded Systems.
PSO 2:	Understand the architectural specifications of a communication system and determine their performance.

Course Title	**:**	**Embedded System Design**	
Course Code	**:**	**BEC601**	
Year / Semester	**:**	**3rd Year / 6th Semester**	
Academic Year	**:**	**2025 – 26**	
Syllabus	**:**	**Experiments**	**Page No.**
		1. ALP to find the sum of first 10 integer numbers.	**4**
		2. Write an ALP to i) multiply two 16-bit binary numbers.ii) add two 32-bit numbers	**5**
		3. Write an ALP to find factorial of a number.	**6**
		4. Write an ALP to add an array of 16-bit numbers and store the 32-bit result in internal RAM.	**7**
		5. Write an ALP to find the square of a number (1 to 10) using look-up table	**8**
		6. Write an ALP to find the largest/smallest number in an array of 32 numbers.	**9**
		7. Write an ALP to arrange a series of 32-bit numbers in ascending/descending order.	**11**
		8. i) Write an ALP to count the number of ones and zeros in two consecutive memory	**13**
		9. Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.	**15**
		10. Interface a DAC and generate Triangular and Square waveforms.	**19**
		11. Display the Hex digits 0 to F on a 7-segment LED interface, with a suitable delay in between.	**23**
		12. Interface a simple Switch and display its status through Relay, Buzzer and LED	**27**
		13. Virtual Lab Experiment:	**30**
COs	**:**	**CO1:**	Describe the architectural features and instructions of 32-bit microcontroller ARM Cortex M3
		CO2:	Apply the knowledge gained for Programming ARM Cortex M3 for different applications.
CO / PO Mapping	**:**	**COs**	**CO1** / **CO2**
		POs / PSOs	**PO1, PO5, PSO1** / **PO1, PO2, PO5, PSO1**
Course In-Charge	**:**	**Prof. Nikhil A. Kulkarni**	

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
1	2				2								2	
2	2	2			2								2	

Evaluation Details

Students Assessment through CIE (50%) + SEE (50%)

Parameter	Particulars	Marks	Total	Scale Down Marks
CIE	Performance	05	130 (10*13 Expt.)	15
	Viva-Voce	02		
	Journal	03		
LAB IA-1	Write-up+ Conduction+ Result	30	50	10
	Viva	20		
			Total Marks	25

Note:

1. CIE for each lab session will be for 10 Marks
2. Total CIE marks scale down to 15 Marks
3. One lab Internals to be conducted for 50 Marks each
4. Lab Internal marks to be scale down to 10Marks

GETTING STARTED WITH KEIL MICRO VISION SOFTWARE AND FLASH MAGIC

KEIL UVISION4 IDE INSTALLATION

INSTALLATION OF KEILUVISION4 AS FOLLOWS.

1. GO TO **EXE** FOLDER AND THEN UVISION4.2 IN THE CD AND RUN KEIL4 ARM.EXE FILE.

2. **NEXT**

3. CLICK ON THE OPTION "I AGREE TO ALL THE TERMS OF..." AND THEN GIVE NEXT

4. **NEXT**

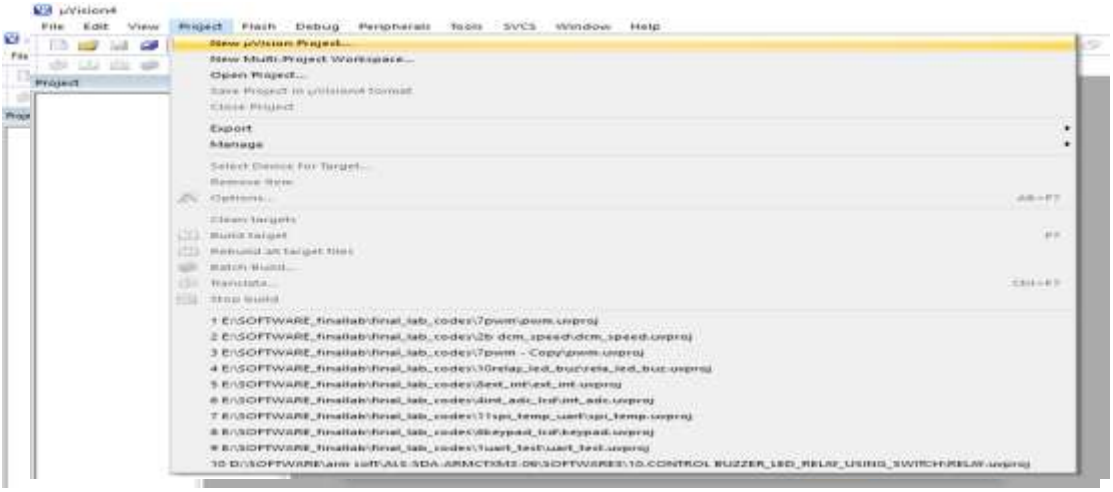
5. GIVE NAME AND THE MAIL ID (IT MIGHT BE ANY MAIL ID) AND THEN NEXT

6. CLICK FINISH TO COMPLETE THE INSTALLATION.

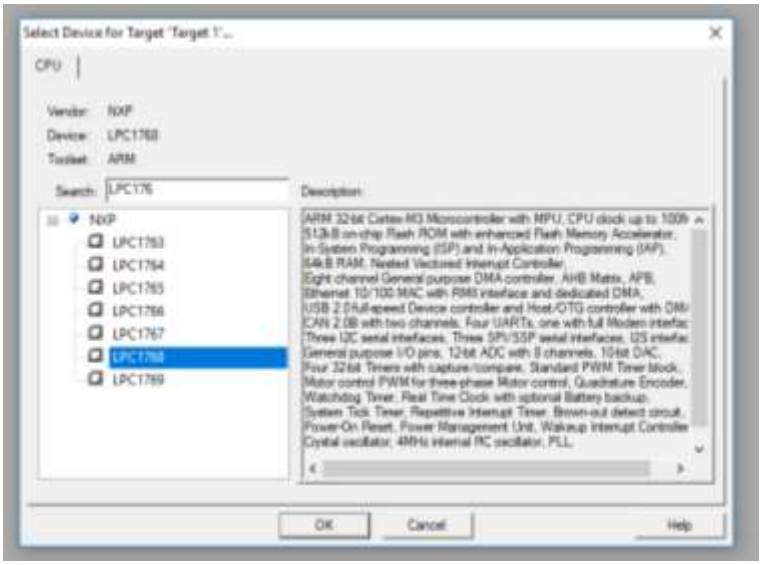
HOW TO USE KEIL 4 SOFTWARE AND FLASH MAGIC

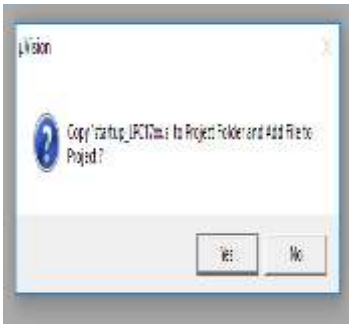
PROJECT CREATION IN KEIL UVISION4 IDE

1. CREATE A FOLDER WITH YOUR USN OR NAME BEFORE CREATING NEW PROJECT.
2. USE SEPARATE FOLDER FOR EACH PROJECT.
3. OPEN KEIL UVISION4 IDE SOFTWARE BY DOUBLE CLICKING ON "KEIL UVISION4" ICON.
4. GO TO "PROJECT" THEN TO "NEW PROJECT" AND SAVE IT WITH A NAME IN THE RESPECTIVE PROJECT FOLDER, ALREADY YOU CREATED.

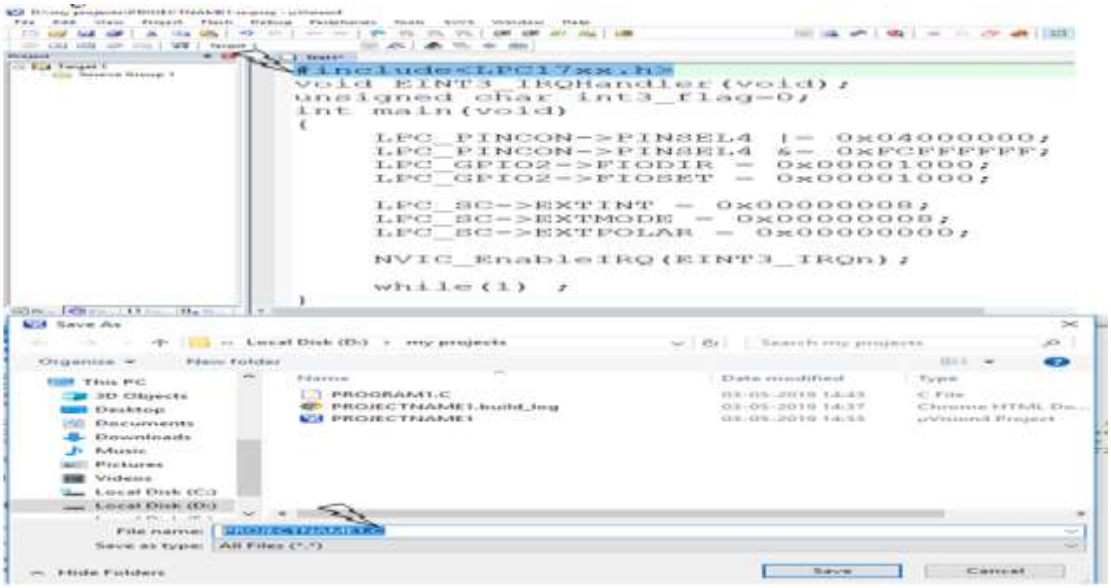


5. SELECT THE DEVICE AS “NXP (FOUNDED BY PHILIPS)” IN THAT “LPC1768” THEN PRESS OK AND THEN PRESS “NO” BUTTON TO ADD “SYSTEM_LPC17XX.S” FILE.





6. GO TO “FILE” IN THAT “NEW” TO OPEN AN EDITOR WINDOW. CREATE YOUR SOURCE FILE AND USE THE HEADER FILE “LPC17XX.H” IN THE SOURCE FILE AND SAVE THE FILE. COLOR SYNTAX HIGHLIGHTING WILL BE ENABLED ONCE THE FILE IS SAVED WITH A RECOGNIZED EXTENSION SUCH AS “.C “.



7. RIGHT CLICK ON “SOURCE GROUP 1” AND SELECT THE OPTION “ADD FILES TO GROUP 'SOURCE GROUP 1' “ADD THE. C SOURCE FILE(S) **[I.E. THE FILE SAVED, WHICH IS IN THE WORKING DIRECTORY]** TO THE GROUP.

8. AGAIN RIGHT CLICK ON **SOURCE GROUP 1 AND SELECT THE OPTION "ADD EXISTING FILES TO GROUP 'SOURCE GROUP 1'"** ADD **THE LIBRARY FILE START.LIB** [IT SHOULD BE COPIED IN TO THE WORKING DIRECTORY OR LOCATION SHOULD BE TRACED] IN : SO IT IS RECOMMENDED TO COPY THE FILE.

9. THEN GO TO "PROJECT" IN THAT "TRANSLATE" TO COMPILE THE FILE (S).
10. GO TO "PROJECT" IN THAT "BUILD TARGET" FOR BUILDING ALL SOURCE FILES SUCH AS ".C",".ASM", ".H", FILES, ETC…THIS WILL CREATE THE HEX FILE IF NO WARNINGS & NO ERRORS.

NOTE:

IN PROJECT WINDOW RIGHT CLICK "TARGET1" AND SELECT "OPTIONS FOR TARGET 'TARGET1'

THEN GO TO OPTION "TARGET" IN THAT TYPE **XTAL 12.0MHZ**

- SELECT IROM1 (STARTING 0×0 SIZE 0×8000).
- SELECT IRAM1 (STARTING 0×10000000 SIZE 0×8000).
- THEN GO TO OPTION "OUTPUT", SELECT "CREATE HEX FILE".THEN GO TO OPTION "LINKER"
- SELECT USE MEMORY LAYOUT FROM TARGET DIALOG.

FLASH MAGIC STEPS

1. OPEN THE FLASH MAGIC SOFTWARE.
2. CLICK ON **BROWSE,** HERE SELECT THE **LOCATION OF THE PROJECT** AND SELCT THE **.HEX** FILE WHICH IS THE PROJECT NAME.
3. SELECT PROPER COMPORT WHICH CAN BE LOCATED FROM DEVICE MANAGER, HERE SEARCH FOR **CH340 USB PORT.**
4. THEN CLICK ON START AND VERIFY THE OUTPUT ON THE BOARD.

EXPERIMENT 1

ALP TO FIND THE SUM OF FIRST 10 INTEGER NUMBERS.

OBJECTIVE:

- TO UNDERSTAND THE ARITHMETIC INSTRUCTIONS AND LOOPING CONCEPT.
- TO KNOW ABOUT THE EXECUTION FLOW OF ASSEMBLY PROGRAM.

TOOLS REQUIRED: KEIL MICRO VISION 4.0

ASSEMBLY PROGRAM:

```
AREA ADDITION, CODE, READONLY
   MOV R0, #10
   MOV R1, #0
SKIP ADD R1, R1, R0
   SUBS R0, #1
   BNE SKIP
STOP B STOP
   END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

EXPERIMENT 2

WRITE AN ALP TO
I) MULTIPLY TWO 16-BIT BINARY NUMBERS.
II) ADD TWO 32-BIT NUMBERS

OBJECTIVE:

- TO UNDERSTAND THE ARITHMETIC INSTRUCTIONS
- TO KNOW ABOUT THE EXECUTION FLOW OF ASSEMBLY PROGRAM.

TOOLS REQUIRED: KEIL MICRO VISION 4.0

ASSEMBLY PROGRAM:

```
AREA ADDITION, CODE, READONLY
  MOV R1, #6400
  MOV R2, #3200
  MUL R3, R1, R2
 LDR R0,=0X11111111
 LDR R1,=0X22222222
 LDR R2,=0X33333333
 LDR R3,=0X44444444
 ADDS R4,R1,R3
 ADC R5,R0,R2
NOP
 NOP
 NOP
 END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

EXPERIMENT 3
ALP TO FIND FACTORIAL OF A NUMBER

OBJECTIVE:

- TO UNDERSTAND THE ARITHMETIC INSTRUCTIONS AND BRANCH CONTROL INSTRUCTIONS.
- TO UNDERSTAND ABOUT THE LINKING OF MULTIPLE OBJECT FILES IN A PROGRAM.

TOOLS REQUIRED: KEIL MICRO VISION 4.0

ASSEMBLY PROGRAM:

```
AREA FACTORIAL , CODE, READONLY
 MOV R0,#4 ; STORE FACTORIAL NUMBER IN R0
 MOV R1,R0 ; MOVE THE SAME NUMBER IN R1
FACT SUBS R1, R1, #1 ; SUBTRACTION
 CMP R1, #1 ; COMPARISON
 BEQ STOP
 MUL R3,R0,R1; ; MULTIPLICATION
 MOV R0,R3 ; RESULT
 BNE FACT ; BRANCH TO THE LOOP IF NOT EQUAL
STOP NOP
 NOP
 NOP
 END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

EXPERIMENT 4

WRITE AN ALP TO ADD AN ARRAY OF 16-BIT NUMBERS AND STORE THE 32-BIT RESULT IN INTERNAL RAM.

OBJECTIVE:

- TO UNDERSTAND THE MEMORY READ / WRITE OPERATION.

TOOLS REQUIRED: KEIL MICRO VISION 4.0

ASSEMBLY PROGRAM:

```
AREA ARRAY, CODE, READONLY
   MOV R5, #6
   MOV R0, #0
   LDR R1, =VALUE
LOOP LDRH R3, [R1], #2
   ADD R0, R0, R3
   SUBS R5, R5, #1
   CMP R5, #0
   BNE LOOP
   LDR R4, =RESULT
   STR R0, [R4]
STOP B STOP
VALUE DCD 0X0000,0X1111,0X2222,0X3333,0XAAAA,0XBBBB,0XCCCC
AREA INFO, DATA, READWRITE
RESULT DCD 0X00000000
END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

EXPERIMENT 5

WRITE AN ALP TO FIND THE SQUARE OF A NUMBER (1TO10) USING LOOK-UP TABLE

OBJECTIVE:

- TO UNDERSTAND THE APPLICATIONS OF LOOKUP TABLE.

TOOLS REQUIRED: KEIL MICRO VISION 4.0

ASSEMBLY PROGRAM:

```
AREA ARRAY, CODE, READONLY
   AREA SQUARE, CODE, READONLY
   LDR R0, =TABLE
   MOV R1, #6
   MOV R1, R1, LSL #0X2
   ADD R0, R0, R1
   LDR R3, [R0]
STOP B STOP
TABLE DCD 0X00000000
   DCD 0X00000001
   DCD 0X00000004
   DCD 0X00000009
   DCD 0X00000010
   DCD 0X00000019
   DCD 0X00000024
   DCD 0X00000031
   DCD 0X00000040
   DCD 0X00000051
   DCD 0X00000064
   END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

EXPERIMENT 6

WRITE AN ALP TO FIND THE LARGEST/SMALLEST NUMBER IN AN ARRAY OF 32 NUMBERS.

OBJECTIVE:

- TO UNDERSTAND THE ARITHMETIC INSTRUCTIONS AND BRANCH CONTROL INSTRUCTIONS AND SORTING METHOD.
- TO UNDERSTAND THE MEMORY READ / WRITE OPERATION.

TOOLS REQUIRED: KEIL MICRO VISION 4.0

ASSEMBLY PROGRAM:

1) LARGEST

```
    AREA LARGEST, CODE, READONLY
    MOV R5, #6
    LDR R1, =VALUE
    LDR R2, [R1], #4
LOOP LDR R3, [R1], #4
    CMP R2, R3
    BHI LOOP1
    MOV R2, R3
LOOP1 SUBS R5, R5, #1
    CMP R5, #0
    BNE LOOP
    LDR R4, =RESULT
    STR R2, [R4]
STOP B STOP
VALUE DCD
0X44444444,0X22222222,0X11111111,0X33333333,0XAAAAAAAA,0X8888888,0X9
9999999
AREA INFO, DATA, READWRITE
RESULT DCD 0X00000000
END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

2) SMALLEST

```
        AREA SMALLEST, CODE, READONLY
        MOV R5, #6
        LDR R1, =VALUE
        LDR R2, [R1], #4
LOOP    LDR R3, [R1], #4
        CMP R2, R3
        BLS LOOP1
        MOV R2, R3
LOOP1   SUBS R5, R5, #1
        CMP R5, #0
        BNE LOOP
        LDR R4, =RESULT
        STR R2, [R4]
STOP    B STOP
VALUE   DCD
0X44444444,0X22222222,0X11111111,0X33333333,0XAAAAAAAA,0X8888888,0X9
9999999
        AREA INFO, DATA, READWRITE
RESULT  DCD 0X00000000
        END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

EXPERIMENT 7

WRITE AN ALP TO ARRANGE A SERIES OF 32-BIT NUMBERS IN ASCENDING/DESCENDING ORDER.

OBJECTIVE:

- TO UNDERSTAND THE ARITHMETIC INSTRUCTIONS AND BRANCH CONTROL INSTRUCTIONS AND SORTING METHOD.
- TO UNDERSTAND THE MEMORY READ / WRITE OPERATION.

TOOLS REQUIRED: KEIL MICRO VISION 4.0

ASSEMBLY PROGRAM:

ASCENDING

```
AREA SORTCODE,CODE,READONLY
   MOV R5,#4
LOOP1 MOV R6,R5
        LDR R1,=0X40000000
LOOP LDR R2,[R1],#4
        LDR R3,[R1]
        CMP R2,R3
        BLS LOOP2  ; BLS FOR ASCENDING AND BHI FOR DESCENDING
        STR R2,[R1],#-4
        STR R3,[R1]
        ADD R1,#4
LOOP2   SUBS R6,R6,#1
        CMP R6,#0
        BNE LOOP
        SUBS R5,R5,#1
        CMP R5,#0
        BNE LOOP1
STOP B STOP
        END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

DESCENDING

```
AREA SORTCODE,CODE,READONLY
   MOV R5,#4
LOOP1 MOV R6,R5
        LDR R1,=0X40000000
LOOP LDR R2,[R1],#4
        LDR R3,[R1]
        CMP R2,R3
        BHI LOOP2  ; BLS FOR ASCENDING AND BHI FOR DESCENDING
        STR R2,[R1],#-4
        STR R3,[R1]
        ADD R1,#4
LOOP2   SUBS R6,R6,#1
        CMP R6,#0
        BNE LOOP
        SUBS R5,R5,#1
        CMP R5,#0
        BNE LOOP1
STOP B STOP
        END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

EXPERIMENT 8

I)WRITE AN ALP TO COUNT THE NUMBER OF ONES AND ZEROS IN TWO CONSECUTIVE MEMORY LOCATIONS.
II) WRITE AN ALP TO SCAN A SERIES OF 32-BIT NUMBERS TO FIND HOW MANY ARE NEGATIVE.

OBJECTIVE:

- TO UNDERSTAND THE LOAD STORE INSTRUCTIONS AND BRANCH CONTROL INSTRUCTIONS.
- TO UNDERSTAND THE MEMORY READ / WRITE OPERATION.

TOOLS REQUIRED: KEIL MICRO VISION 4.0

ASSEMBLY PROGRAM:

```
  AREA ONEZERO, CODE, READONLY
   MOV R2,#0 ;ONE'S COUNTER
   MOV R3,#0 ;ZERO'S COUNTER
   MOV R7,#2
   LDR R6,=VALUE
LOOP MOV R1,#32
   LDR R0,[R6],#4
LOOP0 MOVS R0,R0,ROR #1
   BHI ONES
ZEROS ADD R3,R3,#1
   B LOOP1
ONES ADD R2,R2,#1
LOOP1 SUBS R1,R1,#1
   BNE LOOP0
   SUBS R7,R7,#1
   CMP R7,#0
   BNE LOOP
STOP B STOP
VALUE DCD 0X00000003, 0X00000002
   END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

ASSEMBLY PROGRAM

```
AREA NEGATIVE,CODE,READONLY
    MOV R5,#7;
    MOV R2,#0;
    LDR R4,=VALUE;
LOOP LDR R1,[R4],#4;
    ANDS R1,R1,#1 <<31;
    BHI FOUND
    B LOOP1
FOUND ADD R2,R2,#1;
LOOP1 SUBS R5,R5,#1;
    CMP R5,#0;
    BNE LOOP
    NOP
VALUE DCD
0X82345678,0XABCD1111,0X23320123,0XABCDABCD,0X456798FA,0XF1234BCD,0X
AB123456
    END
```

RESULT:

REGISTER / MEMORY	BEFORE EXECUTION VALUE	AFTER EXECUTION VALUES

EXPERIMENT 9

INTERFACE A STEPPER MOTOR AND ROTATE IT IN CLOCKWISE AND ANTI-CLOCKWISE DIRECTION.

OBJECTIVE:

- TO UNDERSTAND THE INTERFACING AND ROTATING THE STEPPER MOTOR IN REQUIRED DIRECTION WITH REQUIRED ANGLE WITH DAC APPLICATION.

TOOLS REQUIRED:

SL. NO	ITEM	QUANTITY	SPECIFICATIONS
1	COMPUTER SYSTEM	1	PROCESSOR : PENTIUM PROCESSOR AND ABOVE RAM :1GB
2	SOFTWARE	-	KEIL MICRO-VISION VER. 4, FLASH MAGIC
3	BOARD	1	LPC 1768 ARM CORTEX BOARD
4	CABLE	1	RS232 TO USB
5	POWER SUPPLY	1	5V

EXPERIMENT SETUP IN LPC1768 BOARD

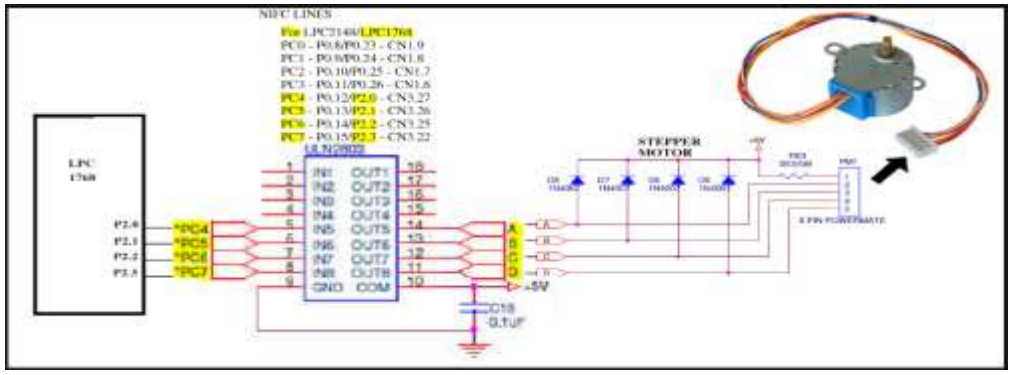


FIG 9.1

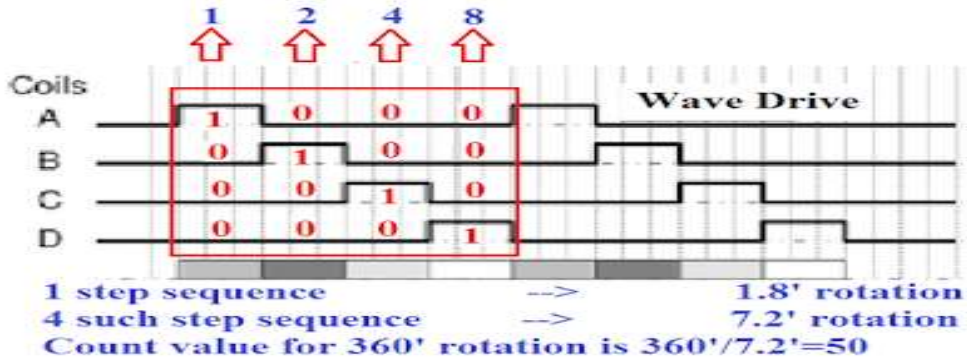


FIG 9.2

CONFIGURATION DETAILS

1. **LPC_PINCON->PINSEL4 = 0X00000000; //P2.0 TO P2.3 GPIO**

Table 84. Pin function select register 4 (PINSEL4 - address 0x4002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2.0	GPIO Port 2.0	PWM1.1	TXD1	Reserved	00
3:2	P2.1	GPIO Port 2.1	PWM1.2	RXD1	Reserved	00
5:4	P2.2	GPIO Port 2.2	PWM1.3	CTS1	Reserved [2]	00
7:6	P2.3	GPIO Port 2.3	PWM1.4	DCD1	Reserved [2]	00
9:8	P2.4	GPIO Port 2.4	PWM1.5	DSR1	Reserved [2]	00

2. **LPC_GPIO2->FIODIR = 0X0000000F; //P2.0 TO P2.3 OUTPUT**

AS THE P2.3 TO P2.0, ARE THE FIRST 4BITS HENCE TO CONFIGURE THEM AS OUTPUT THEY MUST BE SET TO VALUE 1 USING FIODIR, HENCE P2.0 TO P2.3 WILL HAVE THE VALUES 1111 WHICH IS F AND REMAINING 28 BITS WILL BE 0.

NOTE: CHANGING THE COUNT VALUE FROM 50 TO 25, IT WILL CHANGE THE ANGLE FROM 360 TO 180 AND SIMILARLY CHANGING TO 12.5 OR 13, THE ANGLE WILL BE 90.

C CODE:

```
#INCLUDE <LPC17XX.H>
VOID CLOCK_WISE(VOID);
VOID ANTI_CLOCK_WISE(VOID);
UNSIGNED LONG INT VAR1,VAR2;
UNSIGNED INT I=0,J=0,K=0;

  INT MAIN(VOID)
  {
        LPC_PINCON->PINSEL4 = 0X00000000;        //P2.0 TO P2.3 GPIO
        LPC_GPIO2->FIODIR = 0X0000000F;          //P2.0 TO P2.3
OUTPUT
        WHILE(1)
        {
        FOR(J=0;J<50;J++)    //50 TIMES IN CLOCK WISE ROTATION
                CLOCK_WISE();
```

```
FOR(K=0;K<65000;K++);              //DELAY TO SHOW  ANTI_CLOCK ROTATION

FOR(J=0;J<50;J++)          //50 TIMES IN  ANTI CLOCK WISE ROTATION
ANTI_CLOCK_WISE();

FOR(K=0;K<65000;K++);                    //DELAY TO SHOW CLOCK ROTATION
                }
                                         //END OF WHILE(1)
        }
                                         //END OF MAIN
        VOID CLOCK_WISE(VOID)
        {
                    VAR1 = 0X00000001;                    //FOR CLOCKWISE
          FOR(I=0;I<=3;I++)                          //FOR A B C D STEPPING
                {
                 LPC_GPIO2->FIOCLR =  0X0000000F;
                LPC_GPIO2->FIOSET =  VAR1;
                     VAR1 = VAR1<<1;                  //FOR CLOCKWISE
         FOR(K=0;K<4000;K++);                         //FOR STEP SPEED
VARIATION
          }}
        VOID ANTI_CLOCK_WISE(VOID)
        {
                    VAR1 = 0X0000008;          //FOR ANTICLOCKWISE
         FOR(I=0;I<=3;I++)                          //FOR A B C D STEPPING
        {
                LPC_GPIO2->FIOCLR =  0X0000000F;
                        LPC_GPIO2->FIOSET =  VAR1;
                        VAR1 = VAR1>>1;          //FOR ANTICLOCKWISE
        FOR(K=0;K<4000;K++);                         //FOR STEP SPEED
VARIATION
         }}
```

SAMPLE RESULT:

OBSERVATIONS:

EXPERIMENT 10

INTERFACE A DAC AND GENERATE TRIANGULAR AND SQUARE WAVEFORMS.

OBJECTIVE:

- TO UNDERSTAND THE INTERFACING APPLICATION TO CONVERT DIGITAL SIGNAL TO ANALOG SIGNAL WITH THE REQUIRED WAVE SHAPE.

TOOLS REQUIRED:

SL. NO	ITEM	QUANTITY	SPECIFICATIONS
1	COMPUTER SYSTEM	1	PROCESSOR : PENTIUM PROCESSOR AND ABOVE RAM :1GB
2	SOFTWARE	-	KEIL MICRO-VISION VER. 4, FLASH MAGIC
3	BOARD	1	LPC 1768 ARM CORTEX BOARD
4	CABLE	1	RS232 TO USB
5	POWER SUPPLY	1	5V

EXPERIMENT SETUP IN LPC1768 BOARD

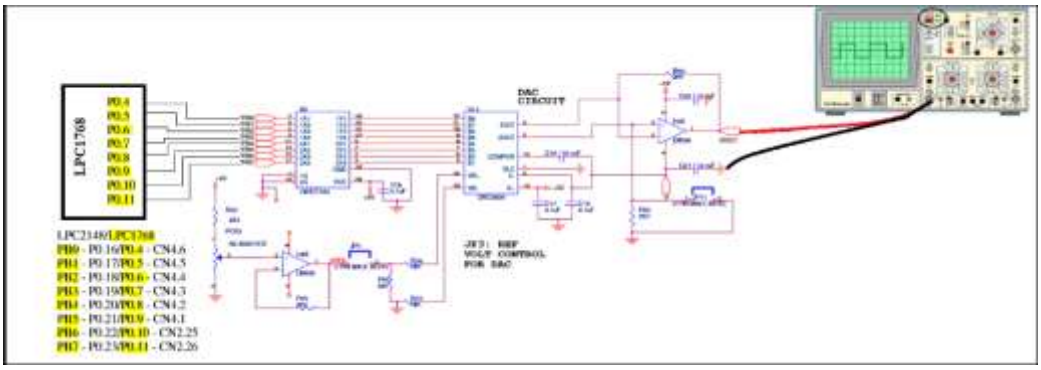


FIG 10.1

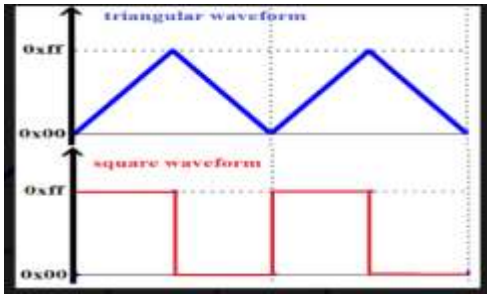


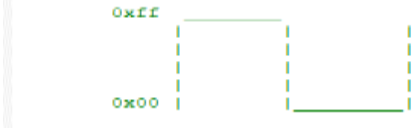
FIG 10.2

CONFIGURATION DETAILS

Table 80. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit description

PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1	00
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1	00
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved	00
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved	00
9:8	P0.4[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0	00
11:10	P0.5[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1	00
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0	00
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1	00
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2	00
19:18	P0.9	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2.3	00
21:20	P0.10	GPIO Port 0.10	TXD2	SDA2	MAT3.0	00
23:22	P0.11	GPIO Port 0.11	RXD2	SCL2	MAT3.1	00
29:24	-	Reserved	Reserved	Reserved	Reserved	0
31:30	P0.15	GPIO Port 0.15	TXD1	SCK0	SCK	00

C CODE:

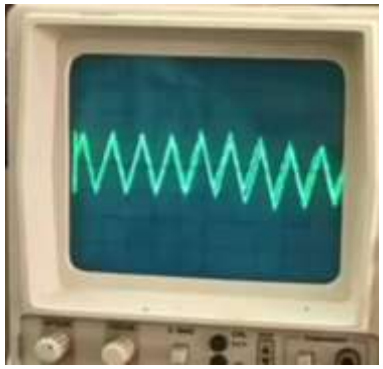



```
// PROGRAM TO GENERATE SQUARE WAVE WITH DAC INTERFACE
/*****************************************************************************
* CONTROLLER    : LPC1768
 * PROJECT : ALS-SDA-ARMCTXM3-06
 * DESCRIPTION    : THIS EXAMPLE EXPLAINS ABOUT HOW SQAURE WAVE IS GENERATED.P0.4 TO P0.11 ARE USED TO GET THE DIGITAL VALUES.

 #INCLUDE <LPC17XX.H>
VOID DELAY(VOID);
INT MAIN ()
{
LPC_PINCON->PINSEL0 = 0X00000000;                    // CONFIGURE P0.4 TO P0.11 AS GPIO
   LPC_GPIO0->FIODIR   = 0X00000FF0 ;                // CONFIGURE P0.4 TO P0.11 AS OUTPUT
         LPC_GPIO0->FIOMASK = 0XFFFFF00F;
      WHILE(1)
   {
      LPC_GPIO0->FIOPIN  = 0X00000FF0 ;
    DELAY();
```

```
    LPC_GPIO0->FIOCLR  = 0X00000FF0 ;
    DELAY();
   }
}
 VOID DELAY(VOID)
{
      UNSIGNED INT I=0;
      FOR(I=0;I<=9500;I++);                              //3MS DELAY
}

// PROGRAM TO GENERATE TRIANGULAR WAVE WITH DAC INTERFACE
/*****************************************************************
* CONTROLLER    : LPC1768
 * PROJECT          : ALS-SDA-ARMCTXM3-06
 * DESCRIPTION    : THIS EXAMPLE EXPLAINS ABOUT HOW
TRIANGULAR WAVE IS GENERATED.P0.4 TO P0.11 ARE USED TO GET
THE DIGITAL VALUES
*****************************************************************
***/
#INCLUDE <LPC17XX.H>
INT MAIN ()
{
      UNSIGNED LONG INT TEMP=0X00000000;
      UNSIGNED INT I=0;

  LPC_PINCON->PINSEL0 = 0X00000000;// CONFIGURE P0.4 TO P0.11 AS
GPIO

LPC_GPIO0->FIODIR   = 0X00000FF0 ; // CONFIGURE P0.4 TO P0.11 AS
OUTPUTS
      LPC_GPIO0->FIOMASK  = 0XFFFFF00F;
       WHILE(1)
   {
/*OUTPUT 0 TO FE*/
      FOR(I=0;I!=0XFF;I++)
      {
      TEMP=I;
      TEMP = TEMP << 4;
      LPC_GPIO0->FIOPIN = TEMP;
      }
      /* OUTPUT FF TO 1*/
      FOR(I=0XFF; I!=0;I--)
      {
```

```
            TEMP=I;
            TEMP = TEMP << 4;
            LPC_GPIO0->FIOPIN = TEMP;
            }
            }  //END OF WHILE(1)
      }//END OF MAIN()
```

SAMPLE RESULT:



OBSERVATIONS:

EXPERIMENT 11

DISPLAY THE HEX DIGITS 0 TO F ON A 7-SEGMENT LED INTERFACE, WITH AN APPROPRIATE DELAY.

OBJECTIVE:

- TO UNDERSTAND THE INTERFACING APPLICATION OF 7 SEGMENT FOR DISPLAYING THE REQUIRED INFORMATION THROUGH THE SPECIFIC PERIPHERAL.

TOOLS REQUIRED:

SL. NO	ITEM	QUANTITY	SPECIFICATIONS
1	COMPUTER SYSTEM	1	PROCESSOR : PENTIUM PROCESSOR AND ABOVE RAM :1GB
2	SOFTWARE	-	KEIL MICRO-VISION VER. 4, FLASH MAGIC
3	BOARD	1	LPC 1768 ARM CORTEX BOARD
4	CABLE	1	RS232 TO USB
5	POWER SUPPLY	1	5V

EXPERIMENT SETUP IN LPC1768 BOARD

	h	g	f	e	d	c	b	a	hex value
0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	1	1	1	1	6F
A	0	1	1	1	0	1	1	1	77
b	0	1	1	1	1	1	0	0	7C
C	0	0	1	1	1	0	0	1	39
d	0	1	0	1	1	1	1	0	5E
E	0	1	1	1	1	0	0	1	79
F	0	1	1	1	0	0	0	1	71

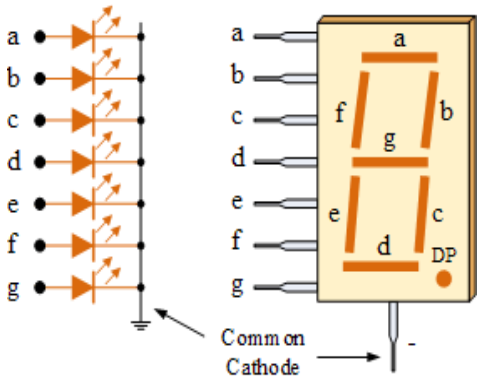


FIG 11.1

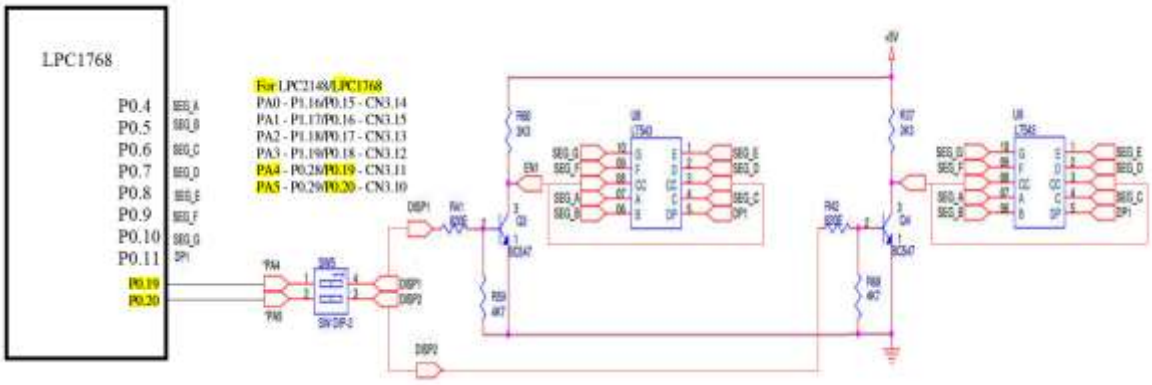


FIG 11.2

CONFIGURATION DETAILS

#DEFINE ALLDISP 0X00180000 //SELECT ALL DISPLAY

Table 81. Pin function select register 1 (PINSEL1 - address 0x4002 C004) bit description

PINSEL1	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.16	GPIO Port 0.16	RXD1	SSEL0	SSEL	00
3:2	P0.17	GPIO Port 0.17	CTS1	MISO0	MISO	00
5:4	P0.18	GPIO Port 0.18	DCD1	MOSI0	MOSI	00
7:6	P0.19[1]	GPIO Port 0.19	DSR1	Reserved	SDA1	00
9:8	P0.20[1]	GPIO Port 0.20	DTR1	Reserved	SCL1	00

#DEFINE DATAPORT 0X00000FF0 //P0.4 TO P0.11: DATA LINES CONNECTED TO DRIVE SEVEN SEGMENTS

Table 80. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit description

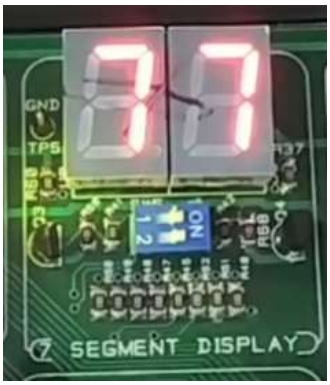
PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1	00
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1	00
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved	00
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved	00
9:8	P0.4[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0	00
11:10	P0.5[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1	00
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0	00
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1	00
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2	00
19:18	P0.9	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2.3	00
21:20	P0.10	GPIO Port 0.10	TXD2	SDA2	MAT3.0	00
23:22	P0.11	GPIO Port 0.11	RXD2	SCL2	MAT3.1	00
29:24	-	Reserved	Reserved	Reserved	Reserved	0
31:30	P0.15	GPIO Port 0.15	TXD1	SCK0	SCK	00

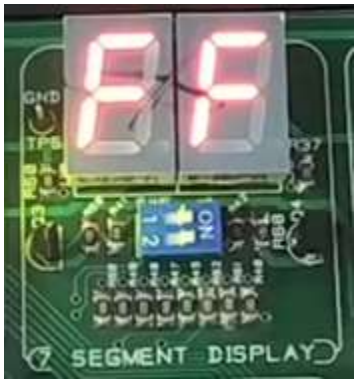
C CODE:

```
/* DESCRIPTION      :\\\\\\\\\\\\\\\\DISPLAY ARE CONNECTED IN COMMON
CATHODE MODE\\\\\\\\\\\\\\\\\\\\
      PORT0 CONNECTED TO DATA LINES OF ALL 7 SEGMENT DISPLAYS

A
       ----
      F|  G |B
       |----|
      E|    |C
       ----  . DOT
        D
      A = P0.04              B = P0.05   C = P0.06   D = P0.07
      E = P0.08      F = P0.09   G = P0.10    DOT = P0.11
      */
#INCLUDE <LPC17XX.H>
UNSIGNED INT DELAY, COUNT=0, SWITCHCOUNT=0,J;
      UNSIGNED INT DISP[16]={0X000003F0, 0X00000060, 0X000005B0,
      0X000004F0,
      0X00000660,0X000006D0, 0X000007D0, 0X00000070, 0X000007F0,
      0X000006F0, 0X00000770,0X000007C0,  0X00000390, 0X000005E0,
      0X00000790, 0X00000710 };
#DEFINE ALLDISP   0X00180000   //SELECT ALL DISPLAY
#DEFINE DATAPORT 0X00000FF0 //P0.4 TO P0.11 : DATA LINES CONNECTED
TO DRIVE SEVEN SEGMENTS
INT MAIN (VOID)
{
      LPC_PINCON->PINSEL0 = 0X00000000;
      LPC_PINCON->PINSEL1 = 0X00000000;
      LPC_GPIO0->FIODIR =          0X00180FF0;
         WHILE(1)
      {
LPC_GPIO0->FIOSET |= ALLDISP;
LPC_GPIO0->FIOCLR =0X00000FF0;         // CLEAR THE DATA LINES TO 7-
SEGMENT DISPLAYS
LPC_GPIO0->FIOSET = DISP[SWITCHCOUNT];   // GET THE 7-SEGMENT
DISPLAY VALUE FROM THE ARRAY
 FOR(J=0;J<3;J++)
FOR(DELAY=0;DELAY<30000;DELAY++);                  // DELAY
SWITCHCOUNT++;
IF(SWITCHCOUNT == 0X10); // 0 TO F HAS BEEN DISPLAYED ? GO BACK TO 0
          {
           SWITCHCOUNT = 0;
```

```
LPC_GPIO0->FIOCLR =    0X00180FF0;
}}}
```

SAMPLE RESULT:





OBSERVATIONS:

EXPERIMENT 12

INTERFACE A SIMPLE SWITCH AND DISPLAY ITS STATUS THROUGH RELAY, BUZZER AND LED

OBJECTIVE:

- TO UNDERSTAND THE APPLICATION AND MECHANISM OF INTERRUPT PROCESS.

TOOLS REQUIRED:

SL. NO	ITEM	QUANTITY	SPECIFICATIONS
1	COMPUTER SYSTEM	1	PROCESSOR : PENTIUM PROCESSOR AND ABOVE RAM :1GB
2	SOFTWARE	-	KEIL MICRO-VISION VER. 4, FLASH MAGIC
3	BOARD	1	LPC 1768 ARM CORTEX BOARD
4	CABLE	1	RS232 TO USB
5	POWER SUPPLY	1	5V

EXPERIMENT SETUP IN LPC1768 BOARD

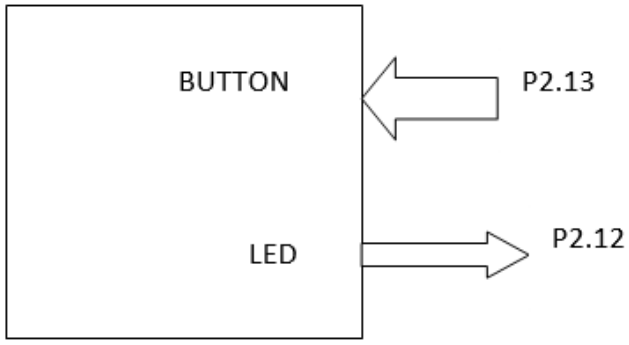


FIG 12.1

CONFIGURATION DETAILS

```
LPC_PINCON->PINSEL4 |= 0X04000000;          //P2.13 AS EINT3
LPC_PINCON->PINSEL4 &= 0XFCFFFFFF;          //P2.12 GPIO FOR LED
```

C CODE:

```
#INCLUDE<LPC17XX.H>
VOID EINT3_IRQHANDLER(VOID);
UNSIGNED CHAR INT3_FLAG=0;
INT MAIN(VOID)
        {
LPC_PINCON->PINSEL4 |= 0X04000000;          //P2.13 AS EINT3
LPC_PINCON->PINSEL4 &= 0XFCFFFFFF;          //P2.12 GPIO FOR LED
LPC_GPIO2->FIODIR = 0X00001000;             //P2.12 IS ASSIGNED OUTPUT
LPC_GPIO2->FIOSET = 0X00001000;             //INITIALL LED IS KEPT ON
LPC_SC->EXTINT = 0X00000008;
//WRITING 1 CLEARES THE INTERRUPT, GET SET IF THERE IS INTERRUPT
LPC_SC->EXTMODE = 0X00000008;
//EINT3 IS INITIATED AS EDGE SENITIVE, 0 FOR LEVEL SENSITIVE
LPC_SC->EXTPOLAR = 0X00000000;
//EINT3 IS FALLING EDGE SENSITIVE, 1 FOR RISING EDGE
//ABOVE REGISTERS, BIT0-EINT0, BIT1-EINT1, BIT2-EINT2,BIT3-EINT3
NVIC_ENABLEIRQ(EINT3_IRQN);                 //CORE_CM3.H
        WHILE(1) ;
        }
        VOID EINT3_IRQHANDLER(VOID)
        {
        LPC_SC->EXTINT = 0X00000008;
//CLEARES THE INTERRUPT
    IF(INT3_FLAG == 0X00)                   //WHEN FLAG IS '0' OFF
THE LED
        {
            LPC_GPIO2->FIOCLR = 0X00001000;
            INT3_FLAG = 0XFF;
        }
        ELSE                        //WHEN FLAG IS FF ON THE LED
        {
            LPC_GPIO2->FIOSET = 0X00001000;
            INT3_FLAG = 0;
        }}
```

SAMPLE RESULT:

OBSERVATIONS:

EXPERIMENT 13

VIRTUAL EXPERIMENT: HN-PART1: INTERFACE A LED MATRIX AND DISPLAY A NUMBER ON THE MATRIX.

URL: http://vlabs.iitkgp.ac.in/rtes/exp11/index.html

OBJECTIVE:

- THIS EXPERIMENT SHOWS HOW TO INTERFACE AN LED MATRIX WITH A MICRO-CONTROLLER AND USE THE SAME TO DISPLAY PATTERNS ON THE MATRIX WITH A DELAY

TOOLS REQUIRED:

HARDWARE:

LEDS

ATMEGA16 MICROCONTROLLER

CRYSTAL OSCILATOR

CAPACITORS

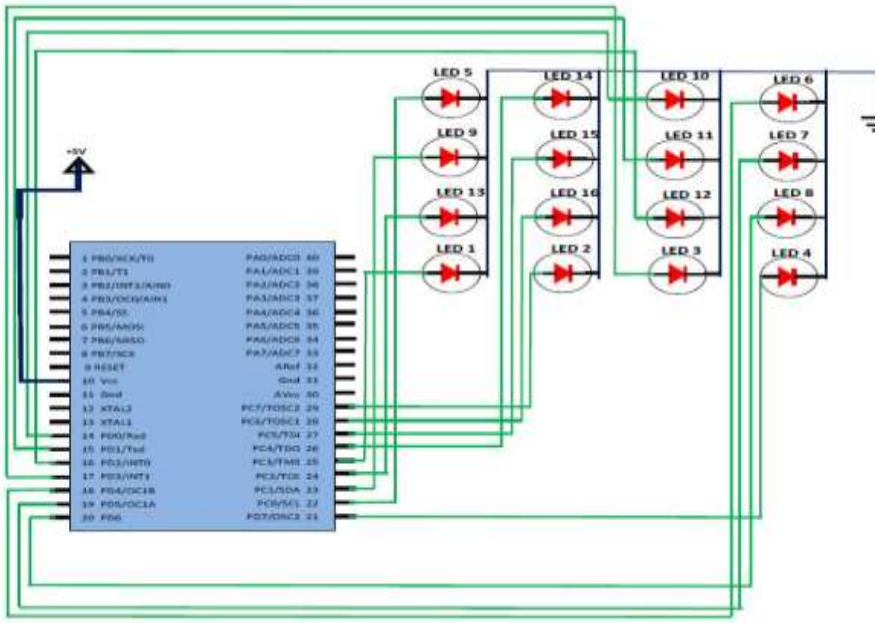
RESISTORS

SOFTWARE:

PROTEUS

AVR STUDIOEXPERIMENT SETUP IN LPC1768 BOARD

PROCEDURE



Step 1: Start

Step 2: Set PORT C and PORT D of ATmega16 as output ports.

Step 3: Refresh the PORTS with 0x00H, before sending the bit pattern for ZERO.

Step 4: Send the required bit pattern to the output PORTS to activate the concerned LEDs of the matrix to represent ZERO

Step 5: Delay

Step 6: Refresh the PORTS with 0x00H, before sending the bit pattern for ONE.

Step 7: send the required bit pattern to the output PORTS to activate the concerned LEDs of the matrix to represent ZERO

Step 8: Goto to Step 3.(Infinite Loop)

CODE:

```
#define F_CPU 1000000UL  // 1 MHz

#include <avr/io.h>
#include <utildelay.h>


int main(void)
{

DDRC=0XFF; //setting portC for output
DDRD=0XFF; //setting portD for output

while(1)
  {
   PORTC=0X00;
   PORTD=0X00;

   PORTC=0X9F;
   PORTD=0XF9;

  _delay_ms(10000);

   PORTC=0X00;
   PORTD=0X00;


   PORTC=0xA0;
   PORTD=0X8F;

   _delay_ms(10000);
   }
}
```

OBSERVATIONS:

KLS VISHWNATHRAO DESHPANDE INSTITUTE OF TECHNOLOGY,

Haliyal – 583 219, Uttara Kannada

(Accredited by NAAC with "A" Grade)

Department of Electronics and Communication Engineering

LABORATORY MANUAL

Course Title	:	**VLSI Design and Testing Lab**
Course Code	:	**BECL606**
Year / Semester	:	**3rd Year / 6th Semester**
Academic Year	:	**2025 - 26**
Course In-Charge	:	**Prof. Deepak Sharma**

College Vision and Mission Statements

Vision
To nurture talent & enrich society through excellence in technical education, research & innovation.

Mission
1. To augment innovative pedagogy & kindle quest for interdisciplinary learning & to enhance conceptual understanding. 2. To build competence, professional ethics & develop entrepreneurial thinking. 3. To strengthen industry institute partnership & explore global collaborations. 4. To inculcate culture of socially responsible citizenship. 5. To focus on holistic & sustainable development.

Department Vision, Mission,PEOs and PSOs Statements

Vision
To bring out talented, skilled, and sustainable Electronics and Communication Engineering Graduates through strong domain expertise to serve the Society with greater Professional Ethics.

Mission
1. To create and impart an active learning ambience to accomplish a high degree of Professional competencies 2. To inculcate innovative research and developmental thinking in effective Teaching and Learning processes for solving Societal challenges 3. To deliver the needs and requirements of the latest state of art of the Industry through quality multidisciplinary internship and training programs

PEOs	
PEO 1:	To be successful in professional career in electronics, communication and allied industries by acquiring the knowledge in the fundamentals of Electronics and Communication Engineering principles and professional skills.
PEO 2:	To be in a position to analyze real life problems and design socially accepted and economically feasible solutions in the respective fields.
PEO 3:	To exhibit good communication skills in their professional career, lead a team with good leadership traits and good interpersonal relationship with the members related to other engineering streams.
PEO 4:	To involve themselves in lifelong learning and professional development by pursuing higher education and participation in research and development activities.
PEO 5:	To demonstrate professional and ethical responsibilities towards their profession, society and the environment.
PSOs	
PSO 1:	An ability to use appropriate modern techniques for analysis, design and development of VLSI and Embedded Systems.
PSO 2:	Understand the architectural specifications of a communication system and determine their performance.

Course Title	**:**	VLSI Design and Testing Lab		
Course Code	**:**	BECL606		
Year / Semester	**:**	3rd Year / 6th Semester		
Academic Year	**:**	2025 – 26		
Syllabus	**:**	**Sl. No.**	**Contents**	**Page No.**
		1.	Design a 4-Bit Adder • Write a Verilog description • Verify the Functionality using Test-bench • Synthesize the design by setting proper constraints and generate the gate level netlist. • From the report generated identify Critical path, Maximum delay, Total number of cells, Power requirement and Total area required	1
		2.	4-Bit Shift and add Multiplier • Write Verilog Code • Verify the Functionality using Test-bench • Synthesize the design by setting proper constraints and obtain the gate level netlist. • From the report generated identify Critical path, Maximum delay, Total number of cells, • Power requirement and Total area required	5
		3.	32-Bit ALU Supporting 4-Logical and 4-Arithmetic operations, using case and if statement for ALU Behavioral Modeling • Write Verilog description • Verify functionality using Test-bench • Synthesize the design targeting suitable library and by setting area and timing constraints • Tabulate the Area, Power and Delay for the Synthesized netlist • Identify Critical path	8
		4.	Flip-Flops (D, SR and JK) • Write the Verilog description • Verify the Functionality using Test-bench • Synthesize the design by setting proper constraints and obtain the gate level netlist. • From the report gate level netlist identify Critical path, Maximum delay, Total number of cells, Power requirement and Total area required. • Verify the functionality using Gate level netlist and compare the results at RTL and gate level netlist.	12
		5.	Four-bit Synchronous MOD-N counter with Asynchronous reset • Write Verilog Code • Verify functionality using Test-bench • Synthesize the design targeting suitable library and by setting area and timing constraints • Tabulate the Area, Power and Delay for the Synthesized netlist • Identify Critical path • Verify the functionality using Gate level netlist and compare the results at RTL and gate level netlist.	17
		6.	a) Construct the schematic of CMOS inverter with load capacitance of 0.1pF and set the widths of inverter with Wn =	20

			Wp, Wn = 2Wp, Wn = Wp/2 and length at selected technology. Carry out the following: i. Set the input signal to a pulse with rise time, fall time of 1ns and pulse width of 10ns and the time period of 20ns and plot the input voltage and output voltage of designed inverter? ii. From the simulation result compute tpHL, tpLH and td for all three geometrical settings of width? iii. Tabulate the results of delay and find the best geometry for minimum delay for CMOS inverter. b) Draw layout of inverter with Wp/Wn = 40/20, use optimum layout methods. Verify for DRC and LVS, extract parasitic and perform post layout simulations, compare the results with pre layout simulations and compare the results.	
		7.	Capture the schematic of 2-input CMOS NOR gate having similar delay as that of CMOS inverter computed in experiment above. Verify the functionality of NOR gate and also find out the delay td for all four possible combinations of input vectors. Table the results. Increase the drive strength to 2X and 4X and tabulate the results.	24
		8.	Construct the schematic of the Boolean Expression Y= AB+CD+E using CMOS Logic. Verify the functionality of the expression find out the delay td for some combination of input vectors. Tabulate the results.	27
		9.	a) Construct the schematic of Common Source Amplifier with PMOS Current Mirror Load and find its transient response and AC response? Measure the Unit Gain Bandwidth (UGB), amplification factor by varying transistor geometries, study the impact of variation in width to UGB. b) Draw Layout of common source amplifier, use optimum layout methods. Verify for DRC & LVS, extract parasitic and perform post layout simulations, compare the results with pre layout simulations. Record the observations.	30
		10.	a) Construct the schematic of two-stage operational amplifier and measure the following: i. Unity gain Bandwidth ii. dB Bandwidth iii. Gain Margin and phase margin with and without coupling capacitance iv. Use the op-amp in the inverting and non-inverting configuration and verify its functionality. v. Study the UGB, 3dB bandwidth, gain and power requirement in op-amp by varying the stage wise transistor geometries and record the observations. b) Draw layout of two-stage operational amplifier with minimum transistor width set to 300 (in 180/90/45 nm technology), choose appropriate transistor geometries as per the results obtained in part a. Use optimum layout methods. Verify for DRC and LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations and perform the comparative analysis.	34
			Demonstration Experiments (For CIE)	
		11.	UART • Write Verilog description • Verify the Functionality using Test-bench • Synthesize the design targeting suitable library and by setting area and timing constraints	39

			• Tabulate the Area, Power and Delay for the Synthesized netlist, Identify Critical path	
		12.	Design and characterize 6T binary SRAM cell and measure the following: • Read Time, Write Time, SNM, Power • Draw Layout of 6T SRAM, use optimum layout methods. Verify for DRC & LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.	47
		13.	Virtual Experiment: Schematic Design of Pass Transistor Logic & Multiplexer	50
CLOs	**:**	• Design, model, simulate and verify CMOS digital circuits. • Perform ASIC design flow and understand the process of synthesis, synthesis constraints and evaluating the synthesis reports to obtain optimum gate level netlist. • Perform RTL-GDSII flow and understand the stages in ASIC design		
COs	**:**	**CO1:**	Design and simulate combinational and sequential digital circuits using Verilog HDL.	
		CO2:	Understand the synthesis process of digital circuits using EDA tool.	
		CO3:	Perform ASIC design flow and understand the process of synthesis, synthesis constraints and evaluating the synthesis reports to obtain optimum gate level net list	
		CO4:	Design and simulate basic CMOS circuits like inverter, common source amplifier, differential amplifier, SRAM	
		CO5:	Perform RTL - GDSII flow and understand the stages in ASIC design	

CO / PO Mapping	**:**	**COs**	**CO1**	**CO2**	**CO2**	**CO4**	**CO5**
		POs / PSOs	**PO1, PO2, PO5, PSO1**	**PO1, PO2, PO5, PSO1**	**PO1, PO2, PO5, PSO1**	**PO1, PO2, PO5, PSO1**	**PO1, PO2, PO5, PSO1**
Course In-Charge	**:**	**Prof. Deepak Sharma**					

CO's And PO's Mapping Matrix

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
1	1	1			3								1	
2	1	1			3								1	
3	2	2			3								1	
4	1	1			3								1	
5	1	1			3								1	

1. ***Evaluation*:**

Students Assessment through CIE (50%) + SEE (50%)

Parameter	Particulars	Marks	Total	Scale Down Marks
CIE	Performance	05	130 (10*13 Expt)	30
	Viva-Voce	02		
	Journal	03		
LAB IA-1	Write-up + Conduction + Acceptable Result + Procedural Knowledge	60	100	20
	Viva	40		
LAB IA-2	Write-up+ Conduction+ Result	60	100	
	viva	40		
			Total Marks	50

Note:

1. CIE for each lab session will be for 10 Marks
2. Total CIE marks scale down to 30 Marks
3. Two lab Internals to be conducted for 100 Marks each
4. Lab Internal marks to be scale down to 20Marks

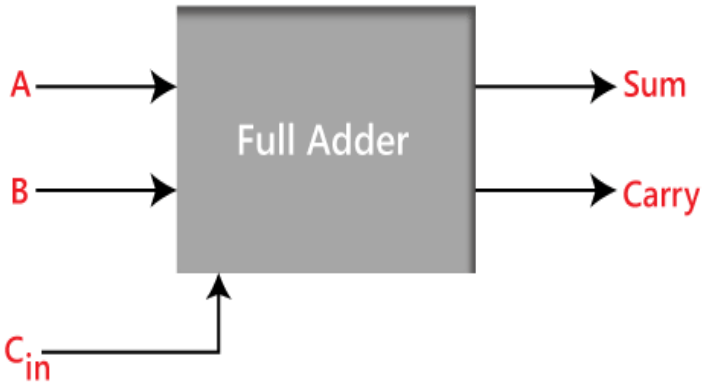
EXPERIMENT 1 – 4 BIT ADDER

Objectives:

- To write a Verilog code for 4bit adder and verify the functionality using Test bench.
- To Synthesize, Analyze Reports and Netlist, Critical Path and Max Operating Frequency.
- To find the total number of cells, power requirement and total area requirement from the report generated.

Software / Tool: Cadence/ Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim) Synthesis: Genus

Block Diagram:



A	B	C	SUM OUT	CARRY OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig. 1.1: Block Diagram of Full Adder and Truth Table

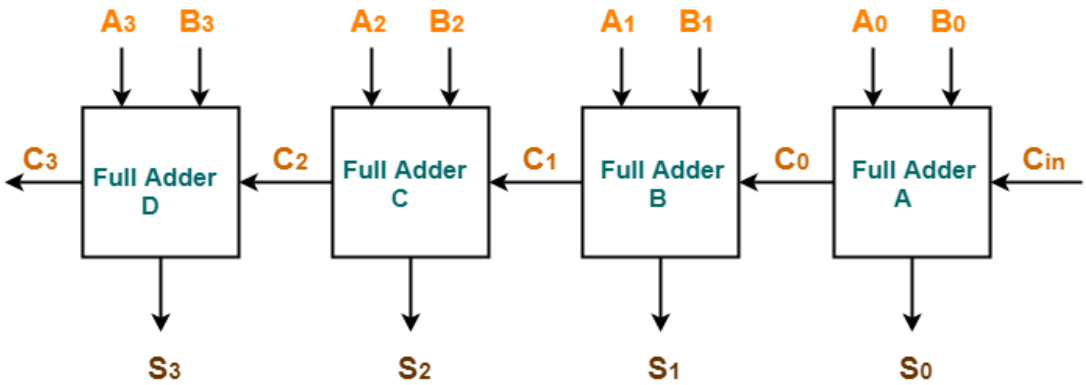


Fig. 1.2: Block Diagram of 4-bit Adder

Verilog code for full Adder:

```
module full_adder( A,B,CIN,S,COUT);
input A,B,CIN;
output S,COUT;
assign S = A^B^CIN;
assign COUT = (A&B) | (CIN&(A^B));
endmodule
```

Verilog code for 4 bit Adder:

```
module four_bit_adder(A,B,C0,S,C4);
input [3:0] A,[3:0] B,C0;
output [3:0] S,C4;
wire C1,C2,C3;
full_adder fa0 (A[0],B[0],C0,S[0],C1);
full_adder fa1 (A[1],B[1],C1,S[1],C2);
full_adder fa2 (A[2],B[2],C2,S[2],C3);
full_adder fa3 (A[3],B[3],C3,S[3],C4);
endmodule
```

Creating Test bench:

Test Bench for 4 bit adder

```
module test_4_bit;
reg [3:0] A;
reg [3:0] B;
reg C0;
wire [3:0] S;
wire C4;
four_bit_adderdut(A,B,C0,S,C4);
initial begin
A = 4'b0011;B=4'b0011;C0 = 1'b0; #10;
A = 4'b1011;B=4'b0111;C0 = 1'b1; #10;
A = 4'b1111;B=4'b1111;C0 = 1'b1; #10;
end
initial
#50 $finish;
endmodule
```

Theory:

- A full adder is a combinational circuit that performs the arithmetic sum of three input bits Ai, addend Bi and carry in C in from the previous adder.
- Its results contain the sum Si and the carryout, C out to the next stage. So, to design a 4-bit adder circuit we start by designing the 1 –bit full adder then connecting the four 1-bit full adders to get the 4-bit adder
- For the 1-bit full adder, the design begins by drawing the Truth Table for the three input and the corresponding output SUM and CARRY.

Procedure:

1. Write the code and save it in a folder.
2. Run the simulation and note down the results.

Waveforms:

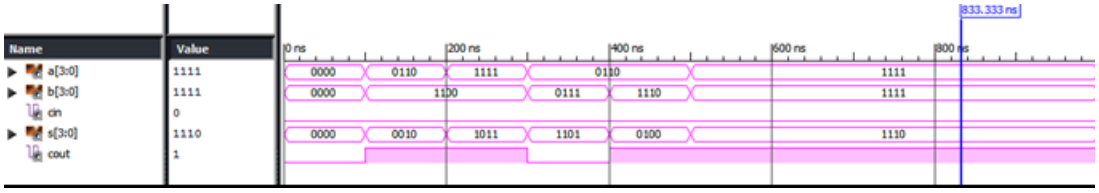


Fig. 1.3: Output Waveform of 4-bit adder

b) Synthesize the design using Constraints and obtain the netlist

Step 1: Getting Started

Make sure you close out all the Incisive tool windows first.
Synthesis requires three files as follows,

- Library Files (.lib)
- Verilog/VHDL Files (.v or .vhdl or .vhd)
- SDC (Synopsis Design Constraint) File (.sdc)

Step 2: Performing Synthesis

The library files are present in the below path,
/home/install/FOUNDRY/digital/<Technology_Node_number>nm/dig/lib/
The Available technology nodes are 180nm , 90nm and 45nm.
In the terminal, initialize the tools with the following commands if a new terminal is being used.

- csh
- source /home/install/cshrc

The tool used for Synthesis is “Genus”. Hence, type “genus -gui” to open the tool.
The following are commands to proceed,

1. read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdl {4bi_adder.v} //Reading multiple Verilog Files
3. elaborate
4. set_top_modulefour_bit_adder //Differentiating Top & Sub Module
5. set_dont_use *XL //Dont Use Cells with High Driving Strength
6. set_dbsyn_generic_effort medium //Setting effort medium
7. set_dbsyn_map_effort medium
8. set_dbsyn_opt_effort medium
9. syn_generic
10. syn_map
11. syn_opt //Performing Synthesis Mapping and Optimisation
12. report_timing -unconstrained >adder_timing.rep
    //Generates Timing report for worst datapath and dumps into file
    //-unconstrained is to be given as no timing constraints are given
13. report_area>adder_area.rep //Generates Synthesis Area report and dumps into a file
14. report_power>adder_power.rep //Generates Power Report [Pre-Layout]
15. write_hdl>adder_netlist.v //Creates readable Netlist File
16. write_sdc>adder_sdc.sdc //Creates Block Level SDC
17. report_qor>adder_qor.rpt // Critical slack path

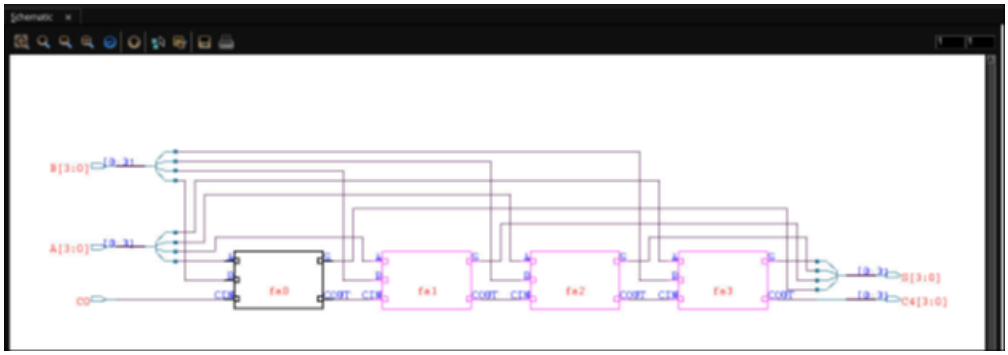


Fig. 1.4: Schematic capture of 4-bit Adder

Observation

Sl. No.	Parameters	Values
1	Critical Path	
2	Maximum Delay	
3	Total Number of cells	
4	Power requirement	
5	Total Area required	

EXPERIMENT 2 – 4-Bit Shift and add Multiplier

Objectives:

- To Write a verilog code for 4-bit Shift and add Multiplier.
- To Verify the Functionality using Test Bench.
- To Synthesize the design by setting proper constraints and obtain the netlist.
- From the report generated identify Critical path, Maximum delay, Total number of cells, Power requirement and Total area required.

Software / Tool: Cadence/ Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim) Synthesis: Genus

Block Diagram:

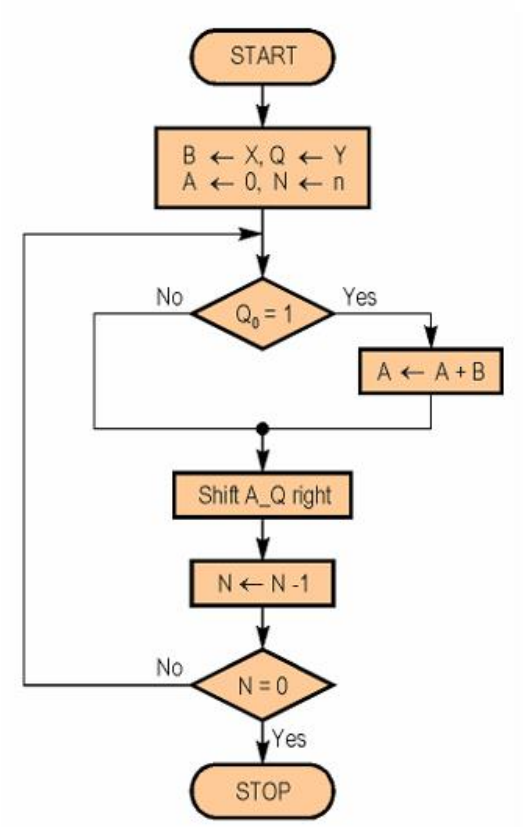


Fig. 2.1: Flow Chart of Shift and Add Multiplier

Theory:

Shift-and-add multiplication is similar to the multiplication performed by pa per and pencil. This method adds the multiplicand X to itself Y times, where Y de notes the multiplier. To multiply two numbers by paper and pencil, the algorithm is to take the digits of the multiplier one at a time from right to left, multiplying the multi plicand by a single digit of the multiplier and placing the intermediate product in the appropriate positions to the left of the earlier results.

As an example, consider the multiplication of two unsigned 4-bit numbers, 8 (1000) and 9 (1001).

```
Multiplicand        1000 ×
Multiplier          1001
                    1000
                   0000
                  0000
                 1000
Product         1001000
```

In the case of binary multiplication, since the digits are 0 and 1, each step of the multiplication is simple. If the multiplier digit is 1, a copy of the multiplicand (1 × multiplicand) is placed in the proper positions; if the multiplier digit is 0, a number of 0 digits (0 × multiplicand) are placed in the proper positions.

Code:

Verilog code for Shift and Add Multiplier:	**Test Bench for Shift and Add Multiplier**

module shiftnadd(a,b,product);
input [3:0]a;
input [3:0]b;
output [7:0]product;
reg[7:0]temp_pdt;
integer i;
always@(*)
begin
temp_pdt=0;
for(i=0;i<4;i=i+1)
begin
if(b[i])
temp_pdt = temp_pdt+(a<<i);
end
end
assign product = temp_pdt;
endmodule
	module shiftnadd_tb;
	reg [3:0]a,b;
	wire [7:0]product;
	
	shiftnadd dut(a,b,product);
	
	initial
	begin
	a=4'b0000;b=4'b0000; #10;
	a=4'b0011;b=4'b0010; #10;
	a=4'b1111;b=4'b0001; #10;
	a=4'b0010;b=4'b0100; #10;
	a=4'b1111;b=4'b1111; #10;
	$finish;
	end
	endmodule

Procedure:

1. Write the code and save in a folder.
2. Run the simulation and note down the results.

Waveforms:

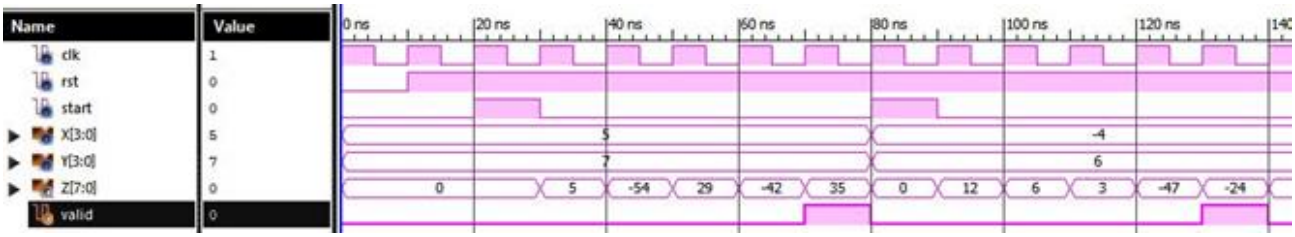


Fig. 2.2: Output Waveform of Multiplier

b) Synthesize the design using Constraints and obtain the netlist.

1. read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdl {alu_32bit_if.v (OR) alu_32bit_case.v} //Choose any one
3. elaborate
4. read_sdcconstraints_top.sdc //Optional-Reading Top Level SDC

5. set_dbsyn_generic_effort medium //Setting effort medium
6. set_dbsyn_map_effort medium
7. set_dbsyn_opt_effort medium
8. syn_generic
9. syn_map
10. syn_opt
//Performing Synthesis Mapping and Optimisation
11. report_timing>alu_timing.rep
//Generates Timing report for worst datapath and dumps into file
12. report_area>alu_area.rep
//Generates Synthesis Area report and dumps into a file
13. report_power>uart_power.rep
//Generates Power Report [Pre-Layout]
14. write_hdl>uart_netlist.v
//Creates readable Netlist File
15. write_sdc>uart_sdc.sdc //Creates Block Level SDC

Observation

Sl. No.	Parameters	Values
1	Critical Path	
2	Maximum Delay	
3	Total Number of cells	
4	Power requirement	
5	Total Area required	

EXPERIMENT 3 – 32-BIT ALU

Objectives:

- To write a verilog code for 32-bit ALU supporting four logical and four arithmetic operations by using case statement and if statement for ALU behavioral modeling.
- To verify the Functionality using Test Bench.
- To Synthesize the design targeting suitable library and by setting area and timing constraints
- To tabulate the Area, Power and Delay for the Synthesized netlist.
- To identify Critical path.

Software / Tool: Cadence/ Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim) Synthesis: Genus

Block Diagram:

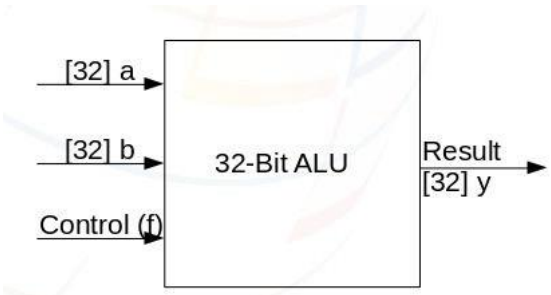


Fig. 3.1: Block Diagram of 32-bit ALU

Verilog code for 32-bit ALU using CASE:	**Creating Test bench:**

module alu_32bit_case(y,a,b,f);
input [31:0]a;
input [31:0]b;
input [2:0]f;
output reg [31:0]y;
always@(*)
begin
case(f)
3'b000:y=a&b; //AND Operation
3'b001:y=a|b; //OR Operation
3'b010:y=~(a&b); //NAND Operation
3'b011:y=~(a|b); //NOR Operation
3'b100:y=a+b; //Addition
3'b101:y=a-b; //Subtraction
3'b110:y=a*b; //Multiply
3'b111:y=a/b; //Division
default:y=32'bx;
endcase
end
endmodule
	Test Bench for 32-bit ALU
	module alu_32bit_tb_case;
	reg [31:0]a;
	reg [31:0]b;
	reg [2:0]f;
	wire [31:0]y;
	alu_32bit_case test2(.y(y),.a(a),.b(b),.f(f));
	initial
	begin
	a=32'h00000000;
	b=32'hFFFFFFFF;
	#10 f=3'b000;
	#10 f=3'b001;
	#10 f=3'b010;
	#10 f=3'b011;
	#10 f=3'b100;
	#10 f=3'b101;
	#10 f=3'b110;
	#10 f=3'b111;
	end
	initial
	#50 $finish;
	endmodule

Verilog code for 32-bit ALU using IF:	**Test Bench for 32-bit ALU**
module alu_32bit_if(y,a,b,f);	module alu_32bit_tb_if;

```
input [31:0]a;
input [31:0]b;
input [2:0]f;
output reg [31:0]y;
always@(*)
begin
if(f==3'b000)
y=a&b; //AND Operation
else if (f==3'b001)
y=a|b; //OR Operation
else if (f==3'b010)
y=a+b; //Addition
else if (f==3'b011)
y=a-b; //Subtraction
else if (f==3'b100)
y=a*b; //Multiply
else if (f==3'b101)
y=a^b; //XOR Operation
else if (f==3'b110)
y=~(a&b); //NAND Operation
else if (f==3'b111)
y=a/b; //Division Operation
else
y=32'bx;
end
endmodule
```

```
reg [31:0]a;
reg [31:0]b;
reg [2:0]f;
wire [31:0]y;
alu_32bit_if test(.y(y),.a(a),.b(b),.f(f));
initial
begin
a=32'h00000000;
b=32'hFFFFFFFF;
#10 f=3'b000;
#10 f=3'b001;
#10 f=3'b010;
#10 f=3'b011;
#10 f=3'b100;
#10 f=3'b101;
#10 f=3'b110;
#10 f=3’b111;
end
initial
#50 $finish;
endmodule
```

Theory:

- The ALU will take in two 32-bit values, and control line.
- An Arithmetic unit does the following tasks like addition subtraction, multi-fiction and logical operations.
- As the input is given in32 bit we get 32-bit output.
- The arithmetic will show only one output at a time, so a selector is necessary to select one of the operator.

Procedure:

1. Write the code and save in a folder.
2. Run the simulation and note down the results.

Waveforms:

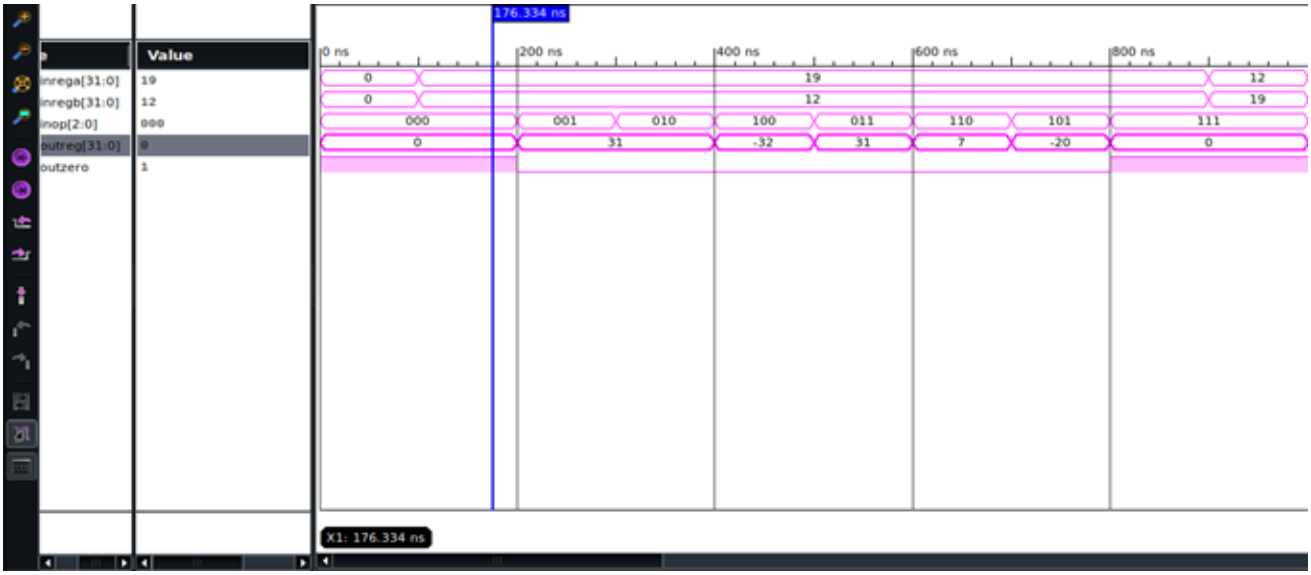


Fig 3.2: Output Waveform of 32-bit ALU

b) Synthesize the design using Constraints and obtain the netlist.

1. read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdl {alu_32bit_if.v (OR) alu_32bit_case.v} //Choose any one
3. elaborate
4. read_sdcconstraints_top.sdc //Optional-Reading Top Level SDC
5. set_dbsyn_generic_effort medium //Setting effort medium
6. set_dbsyn_map_effort medium
7. set_dbsyn_opt_effort medium
8. syn_generic
9. syn_map
10. syn_opt //Performing Synthesis Mapping and Optimisation
11. report_timing>alu_timing.rep //Generates Timing report for worst datapath and dumps into file
12. report_area>alu_area.rep //Generates Synthesis Area report and dumps into a file
13. report_power>uart_power.rep //Generates Power Report [Pre-Layout]
14. write_hdl>uart_netlist.v //Creates readable Netlist File
15. write_sdc>uart_sdc.sdc //Creates Block Level SDC

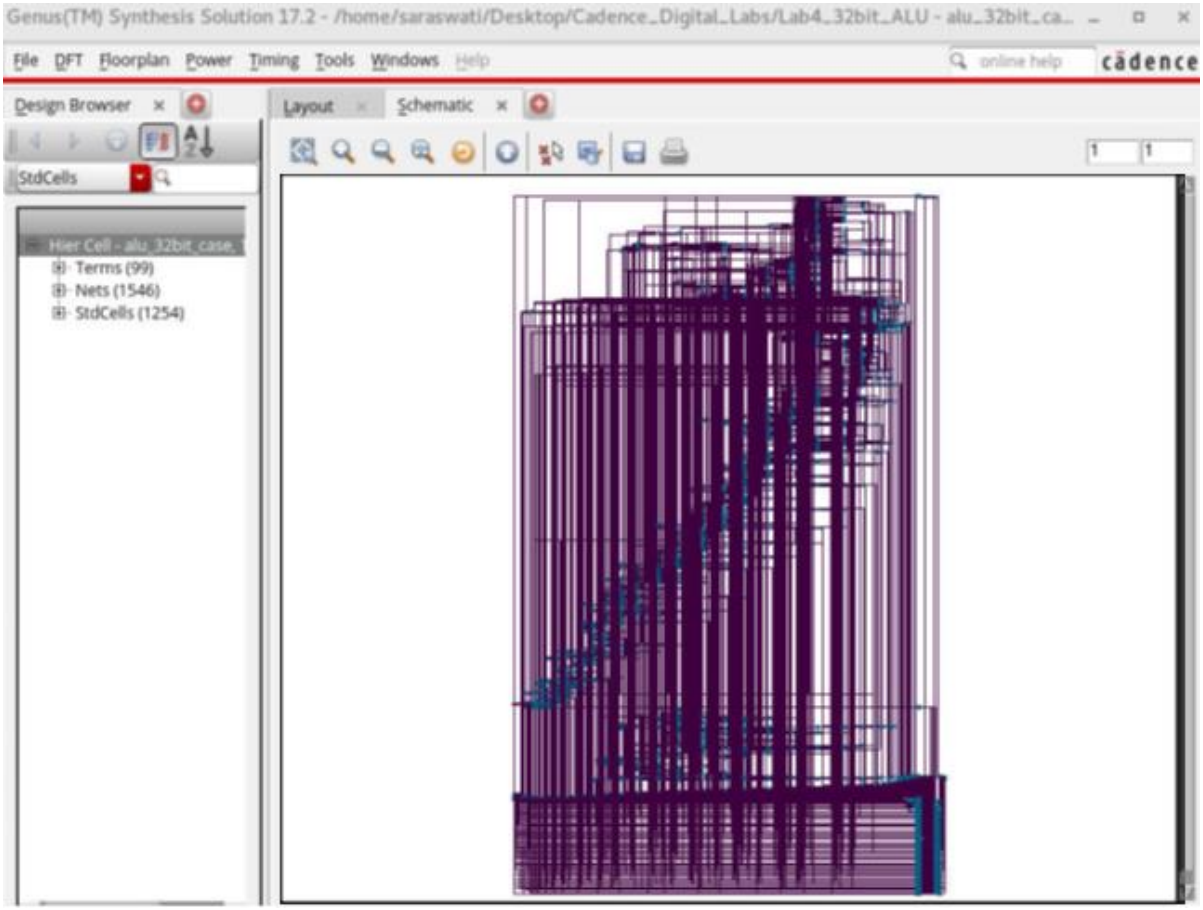


Fig. 3.3: Schematic capture of 32-bit ALU

Observation

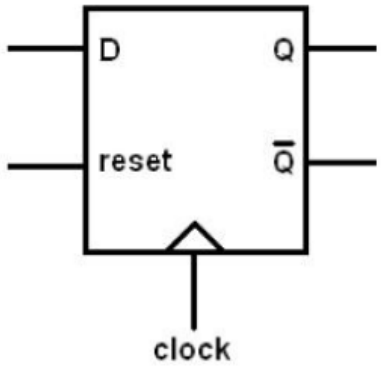
Sl. No.	Parameters	Values
1	Critical Path	
2	Maximum Delay	
3	Power requirement	
4	Total Area required	

EXPERIMENT 4 – FLIP FLOPS

Objective: To write a verilog code for Latch and Flip-flops (D, SR, JK), Synthesize the design and compare the synthesis report.

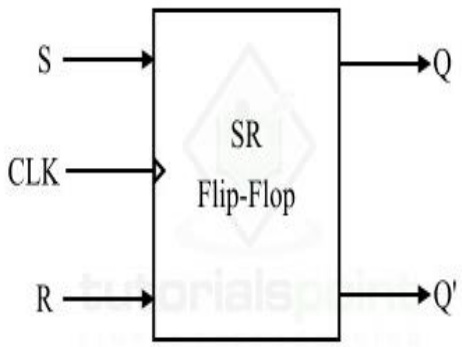
Software / Tool: Cadence/ Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)
Synthesis: Genus

Block Diagram:



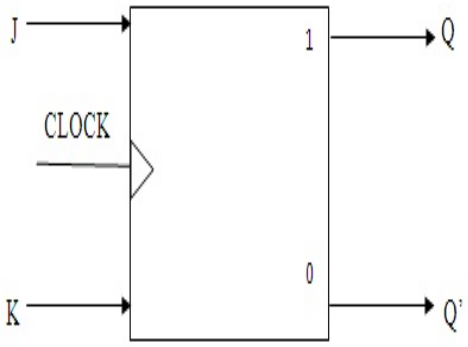
Input			Output	
D	reset	clock	Q	Q'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

Fig. 4.1: Block Diagram and Truth Table of D Flipflop



INPUTS			OUTPUT	STATE
CLK	S	R	Q	
X	0	0	No Change	Previous
↑	0	1	0	Reset
↑	1	0	1	Set
↑	1	1	-	Forbidden

Fig. 4.2: Block Diagram and Truth Table of SR Flipflop



Trigger	Inputs		Output				Inference
			Present State		Next State		
CLK	J	K	Q	Q'	Q	Q'	
⊠	x	x	-		-		Latched
⊓	0	0	0	1	0	1	No Change
⊓			1	0	1	0	
⊓	0	1	0	1	0	1	Reset
⊓			1	0	0	1	
⊓	1	0	0	1	1	0	Set
⊓			1	0	1	0	
⊓	1	1	0	1	1	0	Toggles
⊓			1	0	0	1	

Fig. 4.3: Block Diagram and Truth Table of JK Flipflop

Verilog code for D Flipflop:

```
module DFF(clk, d,q, qnot);
output reg q;
output qnot;
input clk,d;
always @(posedge clk)
begin
case(d)
1'b0:q<=0;
1'b1:q<=1;
endcase
end
assign qnot = ~q;
endmodule
```

Testbench code for D Flipflop

```
module DFF_tb;
wire q, qnot;
reg clk=0;
reg d=0;
DFF dut (clk, d,q, qnot);
initial
begin
d=1'b0; #5;
d=1'b1; #5;
#25 $finish;
end
always #1 clk=~clk;
endmodule
```

Verilog code for SR Flipflop:

```
module SRFF(S,R,clk,q,qnot);
input S,R,clk;
output reg q;
output qnot;
always @(posedge clk)
begin
case({S,R})
2'b00:q<=q;
2'b01:q<=1'b0;
2'b10:q<=1'b1;
2'b11:q<=1'bx;
endcase
end
assign qnot = ~q;
endmodule
```

Testbench code for SR Flipflop

```
module SRFF_tb;
wire q, qnot;
reg clk=0;
reg S=0;
reg R=0;
SRFF dut (S,R,clk,q,qnot);
initial
begin
S=1'b0; R=1'b0; #5;
S=1'b0; R=1'b1; #5;
S=1'b1; R=1'b0; #5;
S=1'b1; R=1'b1; #5;
#25 $finish;
end
always #1 clk=~clk;
endmodule
```

Verilog code for JK Flipflop:

```
module JKFF(J,K,clk,q,qnot);
input J,K,clk;
output reg q;
output qnot;
always @(posedge clk)
begin
case({J,K})
2'b00:q<=q;
2'b01:q<=1'b0;
2'b10:q<=1'b1;
2'b11:q<=~q;
endcase
end
assign qnot = ~q;
endmodule
```

Testbench code for JK Flipflop

```
module JKFF_tb;
wire q, qnot;
reg clk=0;
reg J=0;
reg K=0;
SRFF dut (J,K,clk,q,qnot);
initial
begin
J=1'b0; K=1'b0; #5;
J=1'b0; K=1'b1; #5;
J=1'b1; K=1'b0; #5;
J=1'b1; K=1'b1; #5;
#25 $finish;
end
always #1 clk=~clk;
endmodule
```

Theory:

- Latches and flip-flops are the basic elements for storing information. One latch or flip-flop can store one bit of information. The main difference between latches and flip-flops is that for latches, their outputs are constantly affected by their inputs as long as the enable signal is asserted.
- When they are enabled, their content changes immediately when their inputs change. Flip-flops, on the other hand, have their content change only either at the rising or falling edge of the enable signal.
- This enable signal is usually the controlling clock signal.
- After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes.
- There are basically four main types of latches and flip-flops: SR, D, and JK.
- The major differences in these flip-flop types are the number of inputs they have and how they change state.
- For each type, there are also different variations that enhance their operations.

Procedure:

1. Write the code and save in a folder.
2. Run the simulation and note down the results.

Waveforms:

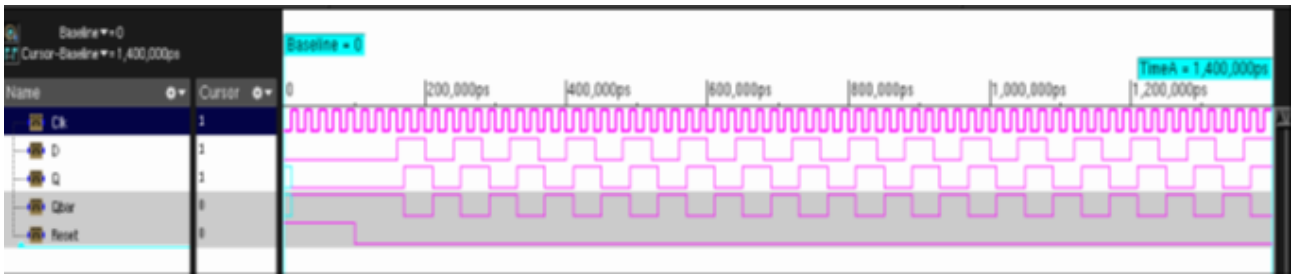


Fig. 4.4: Output Waveform of D Flipflop

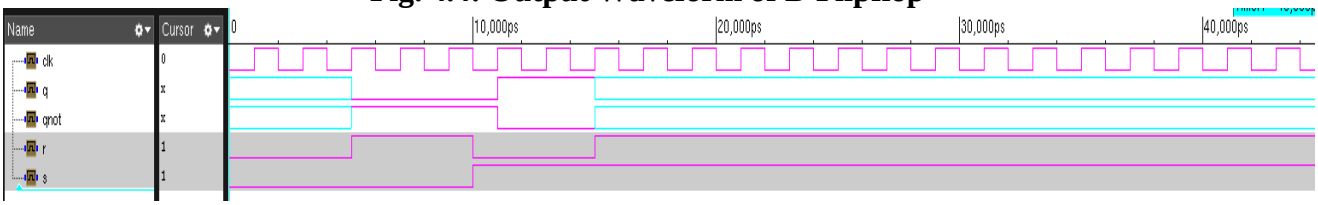


Fig. 4.5: Output Waveform of SR Flipflop

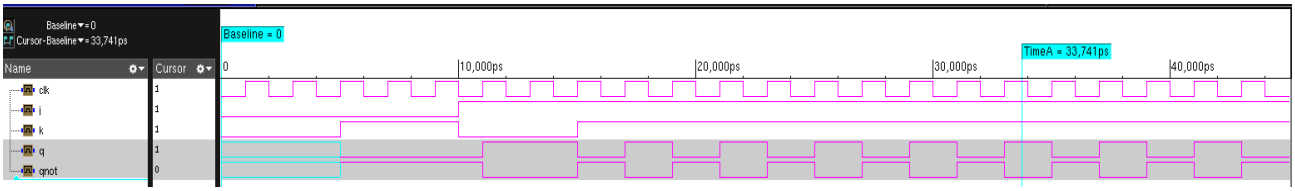


Fig. 4.6: Output Waveform of JK Flipflop

b) Synthesize the design using Constraints and obtain the netlist.

1. read_libs /home/install_run/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdldff.v
3. elaborate
4. read_sdcconstraints_top.sdc
5. set_dbsyn_generic_effort medium
6. set_dbsyn_map_effort medium
7. set_dbsyn_opt_effort medium
8. syn_generic
9. syn_map
10. syn_opt
11. report_timing>dff_timing.rep
12. report_area>dff_area.rep
13. report_power>dff_power.rep
14. write_hdl>dff_netlist.v
15. write_sdc>dff_sdc.sdc

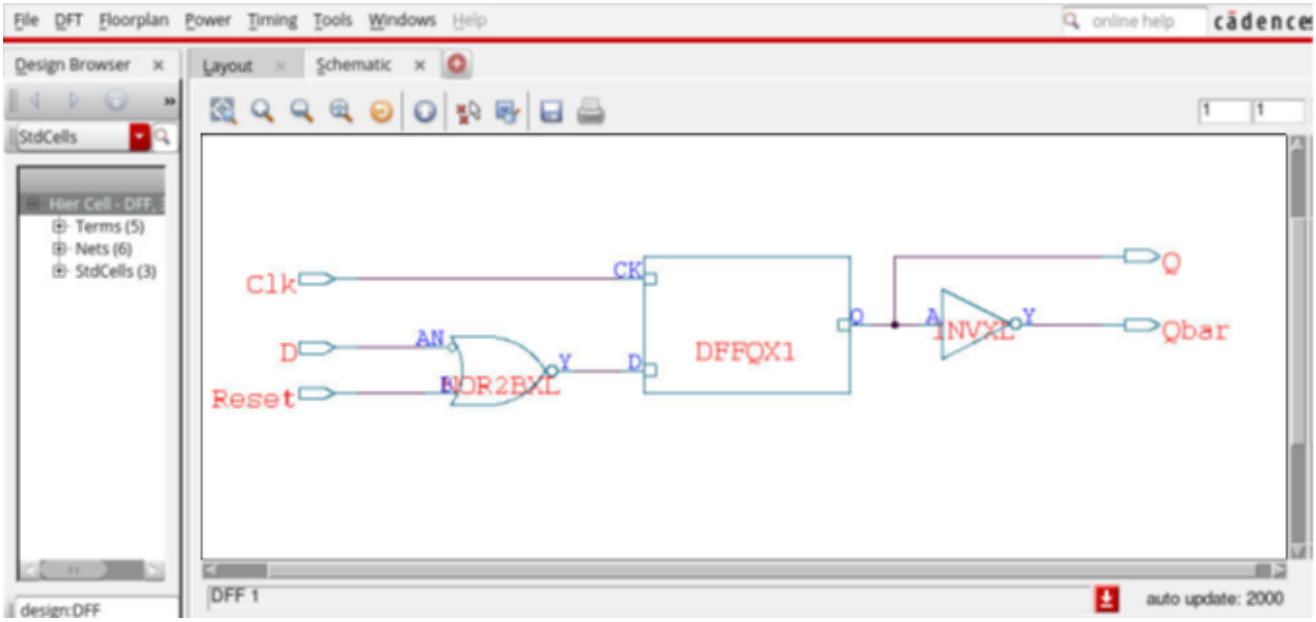


Fig. 4.7: Schematic capture of D Flipflop

Observation

D Flipflop

Sl. No.	Parameters	Values
1	Critical Path	
2	Maximum Delay	
3	Total Number of cells	
4	Power requirement	
5	Total Area required	

SR Flipflop

Sl. No.	Parameters	Values
1	Critical Path	
2	Maximum Delay	
3	Total Number of cells	
4	Power requirement	
5	Total Area required	

JK Flipflop

Sl. No.	Parameters	Values
1	Critical Path	
2	Maximum Delay	
3	Total Number of cells	
4	Power requirement	
5	Total Area required	

EXPERIMENT 5 - FOUR BIT SYNCHRONOUS MOD-N COUNTER WITH ASYNCHRONOUS RESET

Objective: To write a Verilog code for 4-bit Synchronous MOD-N counter with Asynchronous reset

- Verify functionality using Test-bench
- Synthesize the design targeting suitable library and by setting area and timing constraints
- Tabulate the Area, Power and Delay for the Synthesized netlist
- Identify Critical path
- Verify the functionality using Gate level netlist and compare the results at RTL and gate level netlist.

Software / Tool: Cadence/ Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)
Synthesis: Genus

Block Diagram:

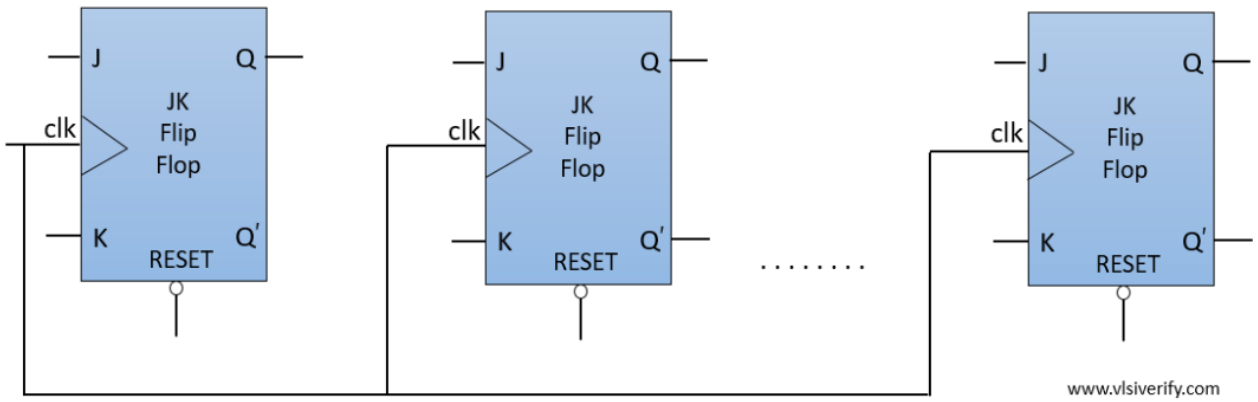


Fig. 5.1: Block diagram of Synchronous MOD-N counter with Asynchronous reset

Theory:

Counters are sequential logic devices that follow a predetermined sequence of counting states triggered by an external clock (CLK) signal. The number of states or counting sequences through which a particular counter advances before returning to its original first state is called the ***modulus*** (MOD). In other words, the modulus (or modulo) is the number of states the counter counts and is the dividing number of the counter.

Modulus Counters, or MOD counters, are defined based on the number of states that the counter will sequence before returning to its original value. For example, a 2-bit counter that counts from 002 to 112 in binary, 0 to 3 in decimal, has a modulus value of 4 (00 → 1 → 10 → 11, and return to 00); therefore, be called a modulo-4, or mod-4, counter. Note also that it has taken four clock pulses to get from 00 to 11.

Code:

Verilog code for 4-bit Synchronous Mod-N Counter with Asynchronous reset:	**Creating Test bench:**
module synchronous_counter #(parameter SIZE=4)(input clk, rst_n,	module tb; reg clk, rst_n; reg up; wire [3:0] cnt;

```
  input up,
  output reg [3:0] cnt
);

  always@(posedge clk) begin
    if(!rst_n) begin
      cnt <= 4'h0;
    end
    else begin
      if(up) cnt <= cnt + 1'b1;
      else cnt <= cnt - 1'b1;
    end
  end
endmodule
```

```
  synchronous_counter(clk, rst_n, up, cnt);

  initial begin
    clk = 0; rst_n = 0;
    up = 1;
    #4; rst_n = 1;
    #80;
    rst_n = 0;
    #4; rst_n = 1; up = 0;
    #50;
    $finish;
  end
  always #2 clk = ~clk;

  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
  end
endmodule
```

Waveform:

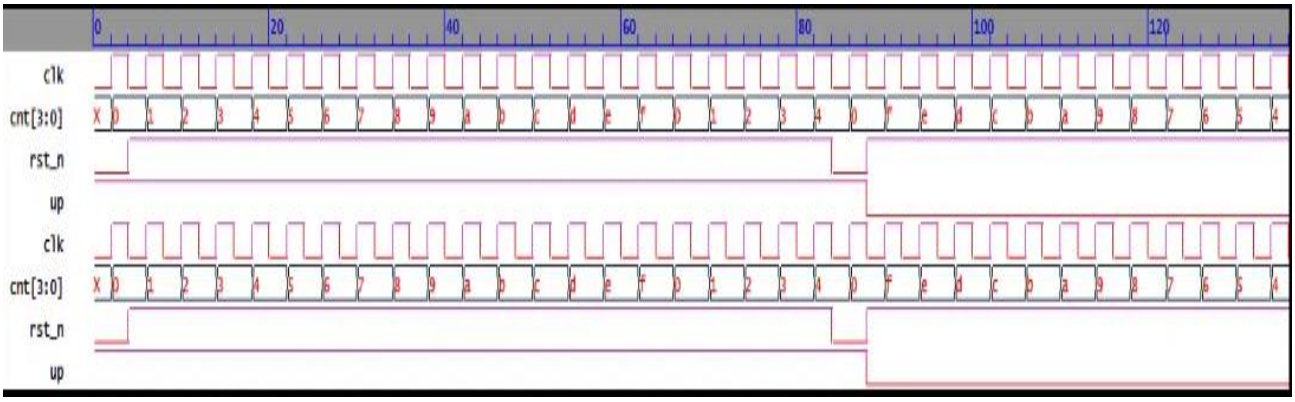


Fig. 5.2: Output Waveform

Synthesize the design using Constraints and obtain the netlist.

1. read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdl {alu_32bit_if.v (OR) alu_32bit_case.v} //Choose any one
3. elaborate
4. read_sdcconstraints_top.sdc //Optional-Reading Top Level SDC
5. set_dbsyn_generic_effort medium //Setting effort medium
6. set_dbsyn_map_effort medium
7. set_dbsyn_opt_effort medium
8. syn_generic
9. syn_map
10. syn_opt //Performing Synthesis Mapping and Optimisation
11. report_timing>alu_timing.rep //Generates Timing report for worst datapath and dumps into file
12. report_area>alu_area.rep //Generates Synthesis Area report and dumps into a file
13. report_power>uart_power.rep //Generates Power Report [Pre-Layout]
14. write_hdl>uart_netlist.v //Creates readable Netlist File
15. write_sdc>uart_sdc.sdc //Creates Block Level SDC

Observation

Sl. No.	Parameters	Values
1	Critical Path	
2	Maximum Delay	
3	Power requirement	
4	Total Area required	

EXPERIMENT 6 – CMOS INVERTER

(a) Capture the Schematic of a CMOS Inverter with Load Capacitance of 0.1Pf and set the widths of Inverter with

(i) $W_N = W_P$

(ii) $W_N = 2W_P$

(iii) $W_N = W_P/2$

and length at selected Technology. Carry out the following:

1. Set the Input Signal to a pulse with Rise Time, Fall Time of 1 ps and Pulse Width of 10 ns, Time Period of 20 ns and plot the input voltage and output voltage of the designed Inverter
2. From the Simulation Results, compute tp_{HL}, tp_{LH} and t_{PD} for all the three geometrical settings of Width
3. Tabulate the results of delay and find the best geometry for minimum delay for CMOS Inverter

Objective: To understand the effect of W/L ratio on delay for CMOS Inverter

Software / Tool: Cadence/ Virtuoso

Circuit Diagram:

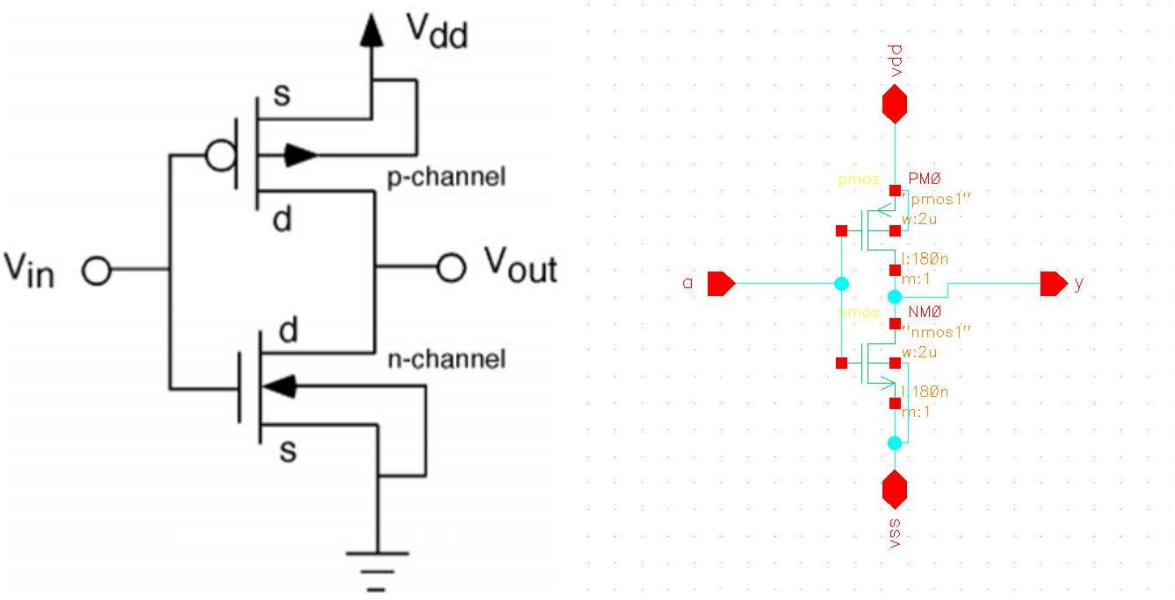


Fig. 6.1: Schematic of CMOS Inverter

Procedure:

1. Invoke the software and create a new folder.
2. Using appropriate libraries design the circuit and save.
3. Create the symbol of the circuit designed and save.
4. Using appropriate libraries design the test circuit and save.
5. Run the simulation for performing Transient and DC Analysis.
6. Calculate and note down the results of tp_{HL}, tp_{LH} and t_{PD} for all the geometrical settings of Width.

7. Repeat the procedure for various W/L ratio for better understanding of effect of W/L ratio on delay for CMOS Inverter

Waveforms:

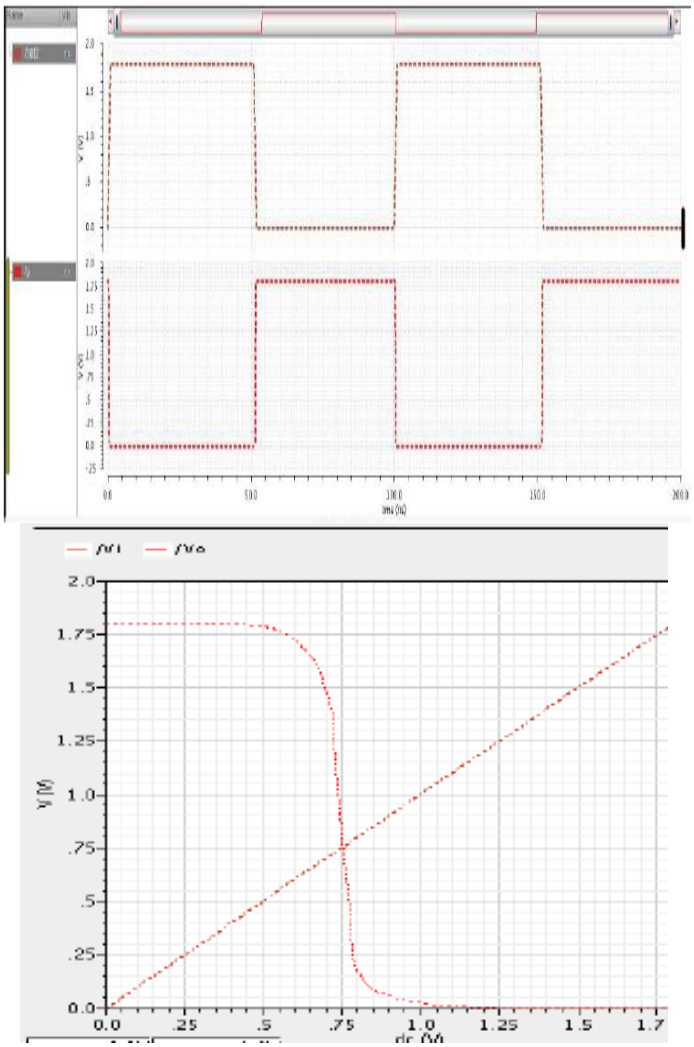


Fig. 6.2: Transient and DC Analysis of CMOS Inverter

Tabulation: Sample results

Table 1: Sample Results of delay for CMOS Inverter

Width Settings	MOSFET	Width	tpLH	tpHL	tpd
Wp =Wn	PMOS	850n	3.233E-10	7.049E-10	5.141E-10
	NMOS	850n			
Wn =2Wp	PMOS	850n	3.337E-10	4.700E-10	4.019E-10
	NMOS	1.7u			
Wn = Wp/2	PMOS	425n	1.141E-09	3.154E-10	7.282E-10
	NMOS	850n			

Observation

Width Settings	MOSFET	Width	tpLH	tpHL	tpd
Wp =Wn	PMOS	850n			
	NMOS	850n			
Wn =2Wp	PMOS	850n			
	NMOS	1.7u			
Wn = Wp/2	PMOS	425n			
	NMOS	850n			

(b) To draw the Layout of CMOS Inverter with Wp/Wn = 40/20 using optimum Layout Methods. Verify for DRC and LVS, extract the Parasitics and perform the Post-Layout Simulations, compare the results with Pre-Layout Simulations and record the observations.

Objective: To understand various terms DRC, ERC, LVS after drawing the Layout by following Lambda based design rules

Software / Tool: Cadence/ Virtuoso, Assura

Layout Diagram:

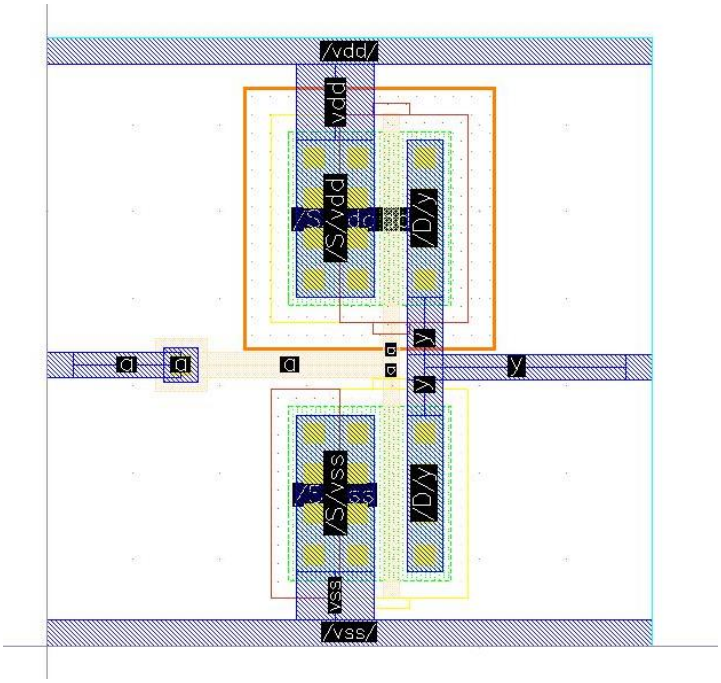


Fig. 6.3: Layout of Inverter

Procedure:

1. Generate Layout by launching Layout XL.
2. Get the layers by connectivity – generate all from source.
3. Connect all layers by following the Lambda based design rules.
4. Do physical verification by running DRC – Design Rule Check, ERC – Electric Rule Check, LVS Layout vs Schematic
5. Extract RC from the Layout

Sample window of LVS:

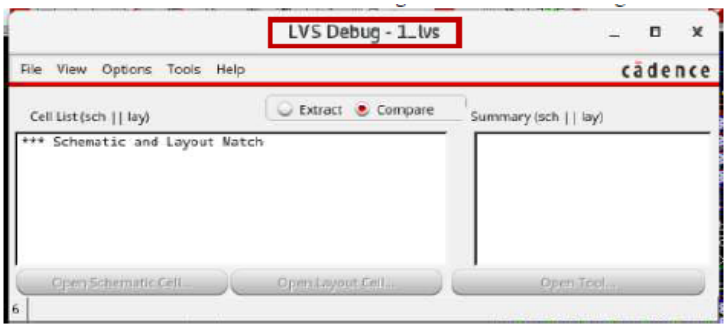


Fig. 6.4: LVS result of CMOS Inverter Layout

EXPERIMENT 7 – CMOS NOR GATE

Objective: Capture the Schematic of a 2 – input CMOS NOR Gate having similar delay as that of CMOS Inverter computed in Expt. – 06. Verify the functionality of the NOR Gate and also find out the delay for all the four possible combinations of input vectors. Table the results. Increase the drive strength to 2X and 4X and tabulate the results.

Objective: To understand the effect of W/L ratio on delay for CMOS NOR Gate.

Software / Tool: Cadence/ Virtuoso

Circuit Diagram:

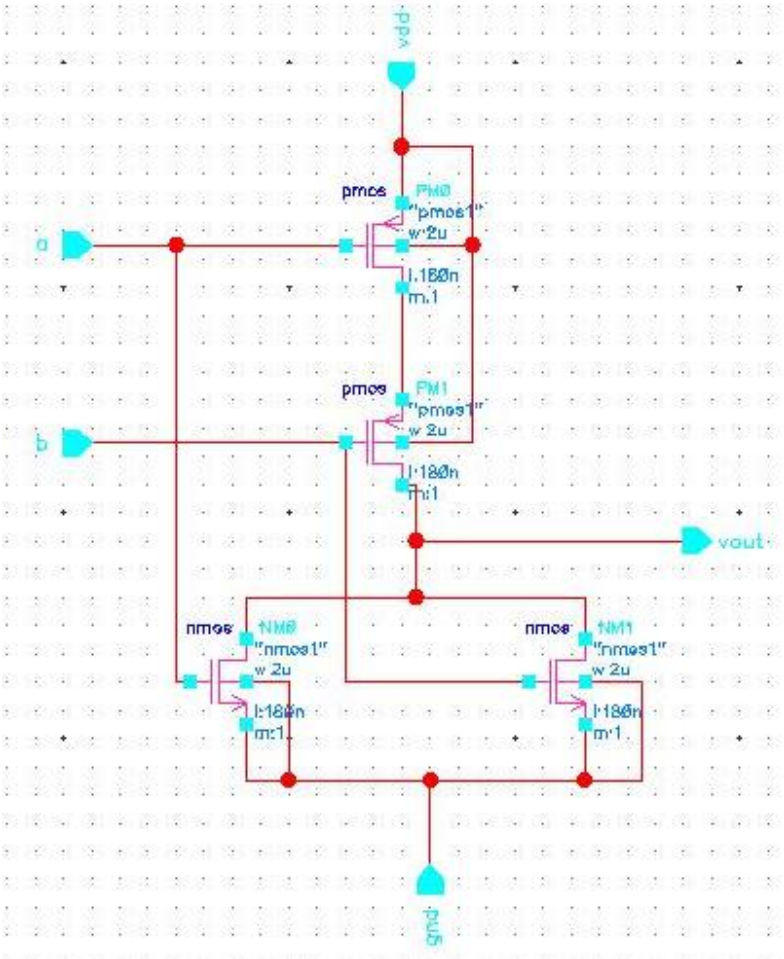


Fig. 7.1: Schematic of CMOS NOR Gate

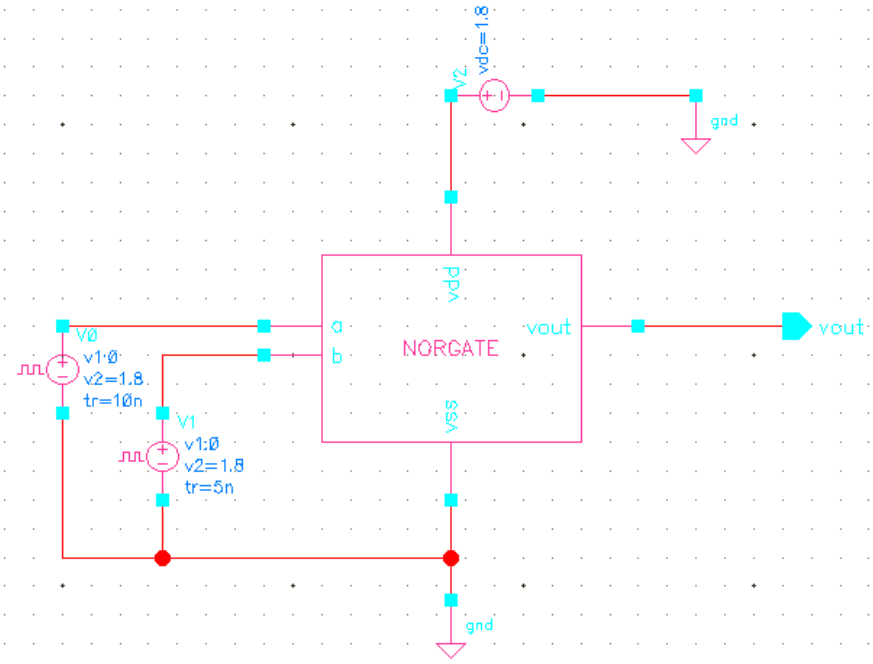


Fig. 7.2: Test Circuit for CMOS NOR Gate

Procedure:

1. Invoke the software and create new folder.
2. Using appropriate libraries design the circuit and save.
3. Create the symbol of the circuit designed and save.
4. Using appropriate libraries design the test circuit and save.
5. Run the simulation for performing Transient Analysis.
6. Calculate and note down the results of tp_{HL}, tp_{LH} and t_{PD} for all the geometrical settings of Width.
7. Repeat the procedure for various W/L ratio for better understanding of effect of W/L ratio on delay for CMOS NOR Gate.

Waveforms:

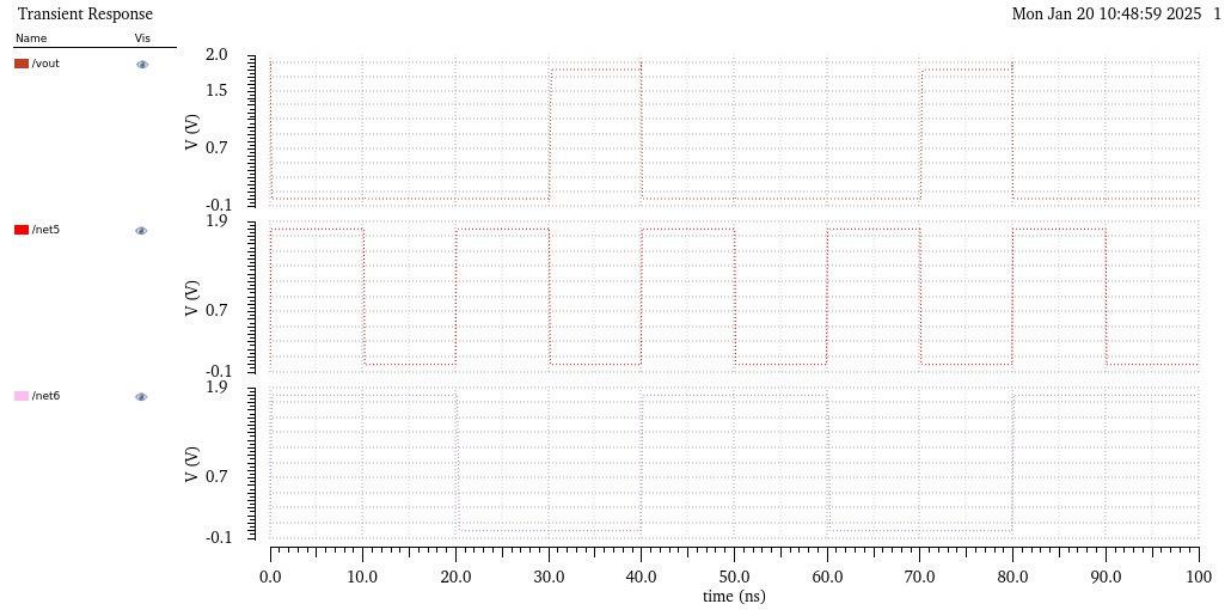


Fig. 7.3: Transient Analysis of CMOS NOR Gate

Table 2: Sample Results of delay for CMOS NOR Gate

MOSFET	Length	Width	tpLH	tpHL	tpd
PMOS	180n	1.5 * 850n = 1.275u	3.720E-10	3.340E-10	3.530E-10
NMOS		2 * 850n = 1.7u			
PMOS	180n	1.5 * 850n * 2 = 2.55u	2.610E-10	2.200E-10	2.400E-10
NMOS		2 * 850n * 2 = 3.4u			
PMOS	180n	1.5 * 850n * 4 = 5.1u	1.900E-10	1.650E-10	1.780E-10
NMOS		2 * 850n * 4 = 6.8u			

Observation

MOSFET	Length	Width	tpLH	tpHL	tpd
PMOS	180n	1.5 * 850n = 1.275u			
NMOS		2 * 850n = 1.7u			
PMOS	180n	1.5 * 850n * 2 = 2.55u			
NMOS		2 * 850n * 2 = 3.4u			
PMOS	180n	1.5 * 850n * 4 = 5.1u			
NMOS		2 * 850n * 4 = 6.8u			

EXPERIMENT 8 – BOOLEAN EXPRESSION Y= (AB+CD+E)' USING CMOS LOGIC

Objective: Capture the Schematic of a Boolean Expression Y= (AB+CD+E)' using CMOS Logic. Verify the functionality of the expression find out the delay td for some combination of input vectors. Tabulate the results.

Software / Tool: Cadence/ Virtuoso

Circuit Diagram:

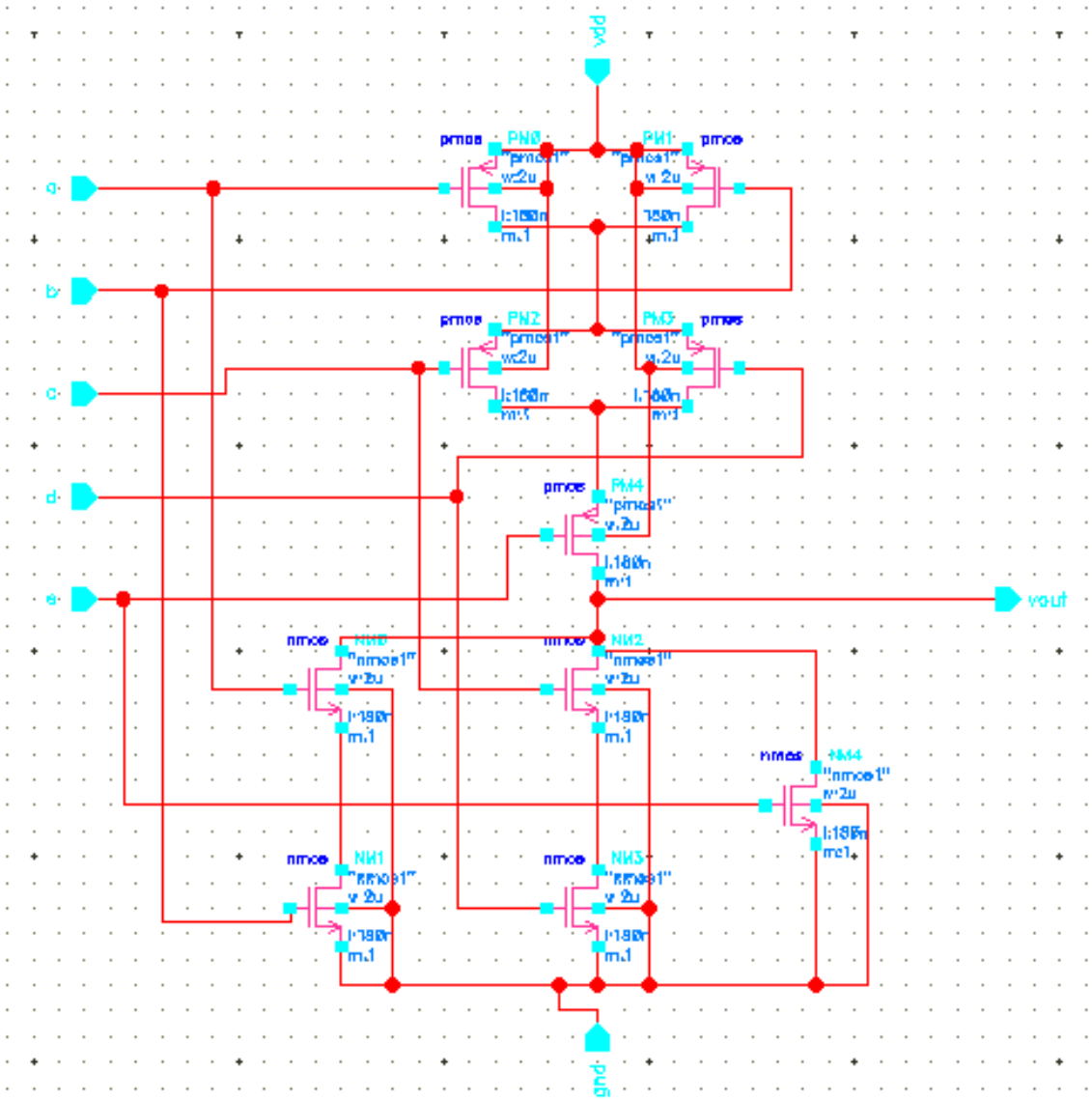


Fig. 8.1: Schematic of Boolean Expression Y = (AB + CD + E)'

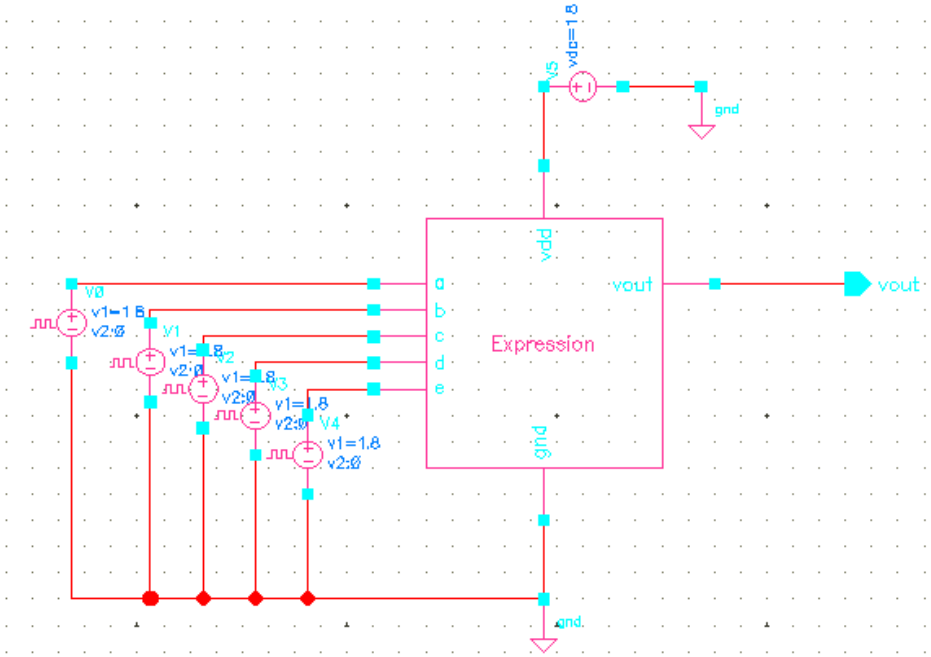


Fig. 8.2: Test Circuit for Boolean Expression Y = (AB + CD + E)'

Procedure:

1. Invoke the software and create new folder.
2. Using appropriate libraries design the circuit and save.
3. Create the symbol of the circuit designed and save.
4. Using appropriate libraries design the test circuit and save.
5. Run the simulation for performing Transient Analysis.
6. Calculate and note down the results of tp_{HL}, tp_{LH} and t_{PD} for all the geometrical settings of Width.

Waveforms:

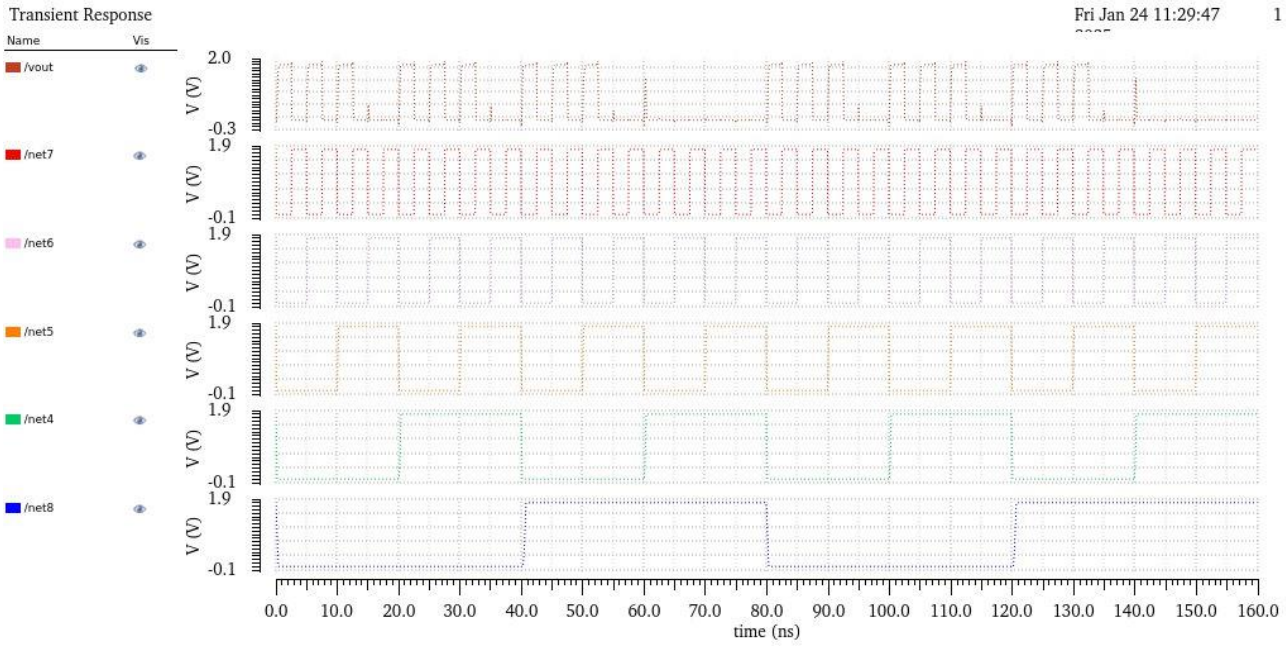


Fig. 8.3: Transient Analysis of Boolean Expression Y = (AB + CD + E)'

Observation

Input		Delay (td)
A	B	
0	0	
0	1	
1	0	
1	1	

EXPERIMENT 9 - COMMON SOURCE AMPLIFIER WITH PMOS CURRENT MIRROR LOAD

(a) Capture the Schematic of a Common Source Amplifier with PMOS Current Mirror Load and find its Transient Response and AC Response. Measure the UGB and Amplification Factor by varying transistor geometries, study the impact of variation in width to UGB.

Objective: To understand the effect of varying transistor geometries on UGB for Common Source Amplifier with PMOS Current Mirror Load

Software / Tool: Cadence/ Virtuoso

Circuit Diagram:

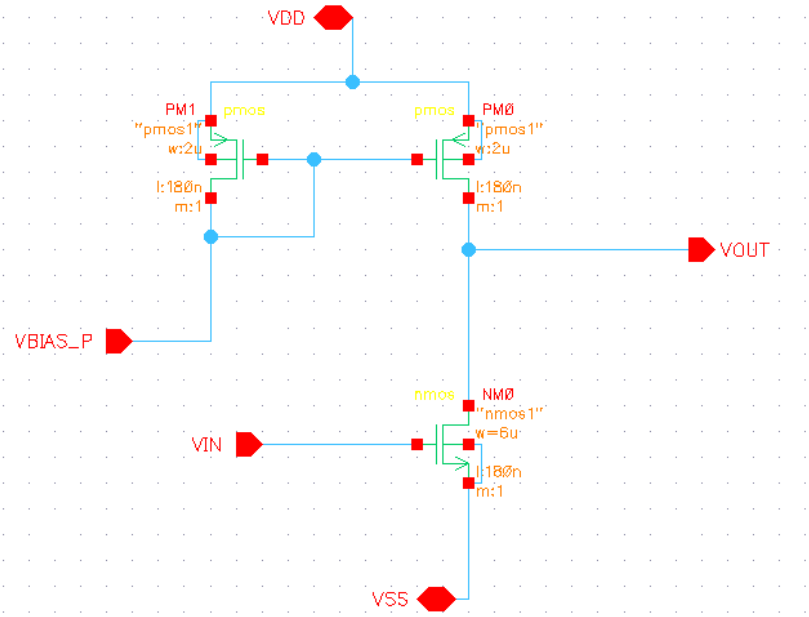


Fig. 9.1: Schematic of Common Source Amplifier with PMOS Current Mirror Load

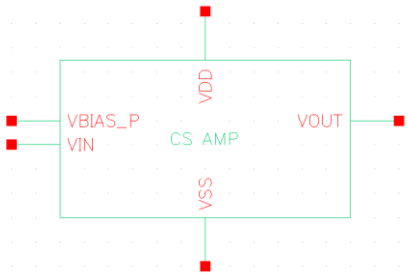


Fig. 9.2: Symbol of Common Source Amplifier with PMOS Current Mirror Load

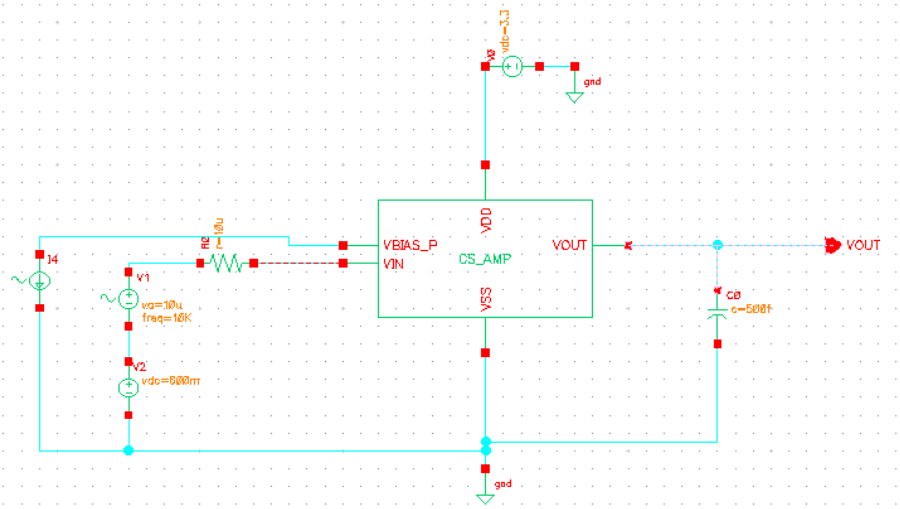


Fig. 9.3: Test Circuit for Common Source Amplifier with PMOS Current Mirror Load

Table 3: Parameters for the devices used in Test Schematic

Library Name	Cell Name	Comments / Properties
analogLib	vdc	DC voltage = 3.3 V (VDD)
analogLib	vdc	DC voltage = VBIAS_N V (Vin)
analogLib	isin	DC current = 100u A (Vbias_P)
analogLib	vsin	AC Magnitude = 1 V, Amplitude = 10u V, Frequency = 10K Hz (Vin)
analogLib	cap	Capacitance = 500f F
analogLib	res	Resistance = 10u Ohms
analogLib	gnd	

Procedure:

1. Invoke the software and create new folder.
2. Using appropriate libraries design the circuit and save.
3. Create the symbol of the circuit designed and save.
4. Using appropriate libraries design the test circuit and save.
5. Run the simulation for performing Transient, DC and AC Analysis.
6. To measure the Gain and Unity Gain Bandwidth, go back to the ADEL window, select "**Results →Direct Plot →AC Magnitude & Phase**". The marker placed on the low frequency part of the response gives the DC Gain, use the bind key "M" to place the marker. Place a horizontal cursor at "0 dB" and the crossing frequency gives the Unity Gain Bandwidth (UGB).

Waveforms:

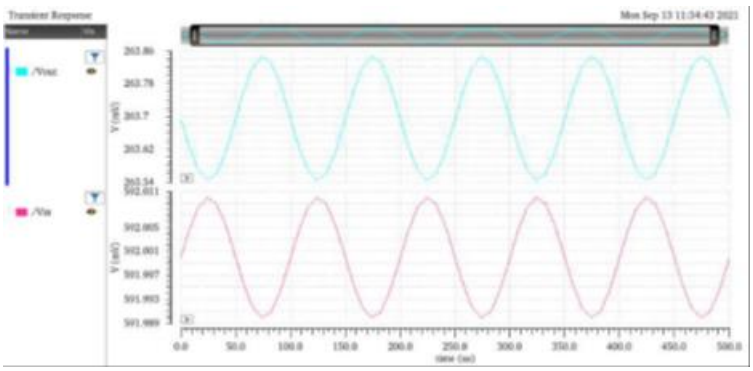


Fig. 9.4: Transient Analysis of Common Source Amplifier with PMOS Current Mirror Load

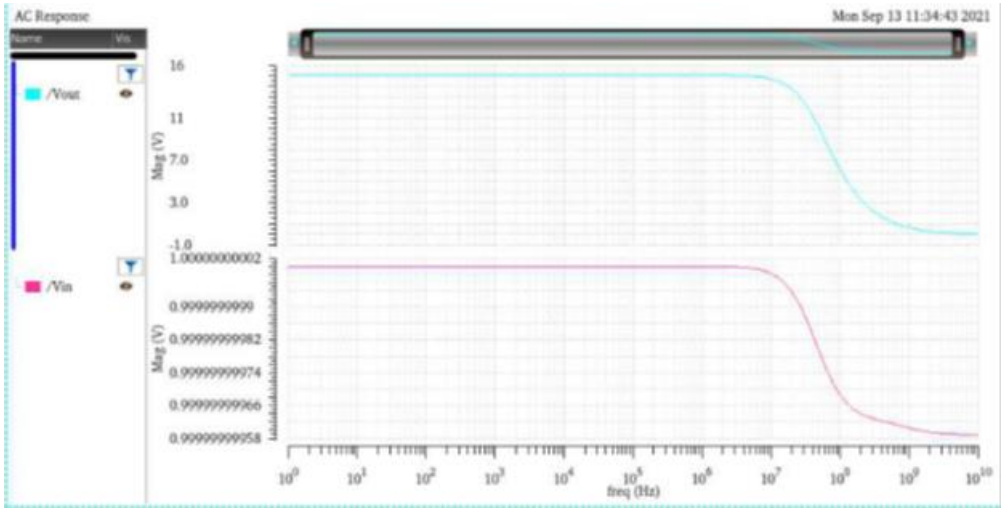


Fig. 9.5: AC Analysis of Common Source Amplifier with PMOS Current Mirror Load

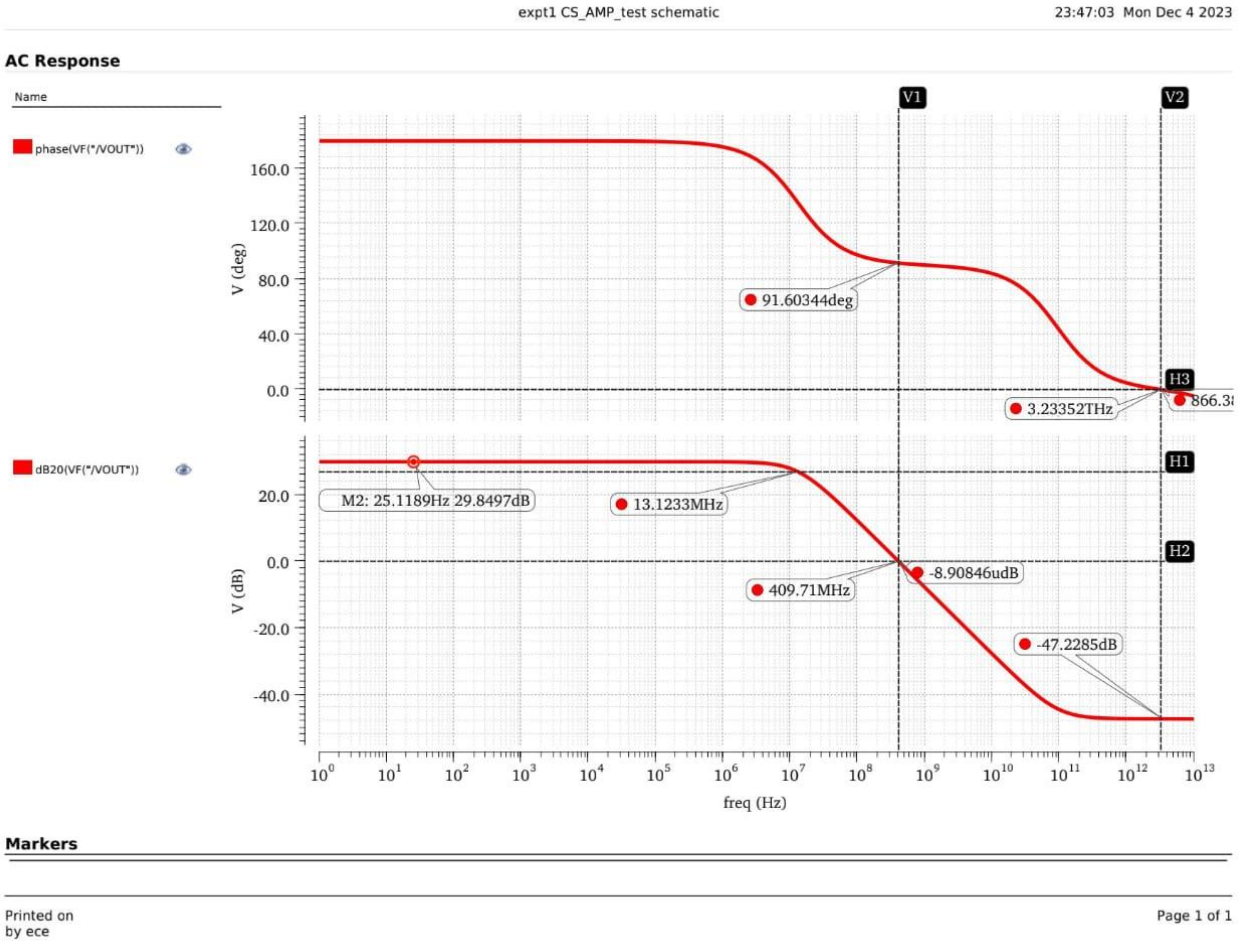


Fig. 9.6: Gain and Phase plot

Observation

Sl. No.	Parameter	Value	
1	UGB		
2	Amplification Factor		

b) Draw the layout of Common Source Amplifier, use optimum layout methods. Verify DRC and LVS, extract the parasitics and perform the post layout simulation, compare the results with pre-layout simulations. Record the observations.

Objective: To understand and perform DRC, ERC, LVS after drawing the Layout by following Lambda based design rules

Software / Tool: Cadence/ Virtuoso, Assura

Layout Diagram:

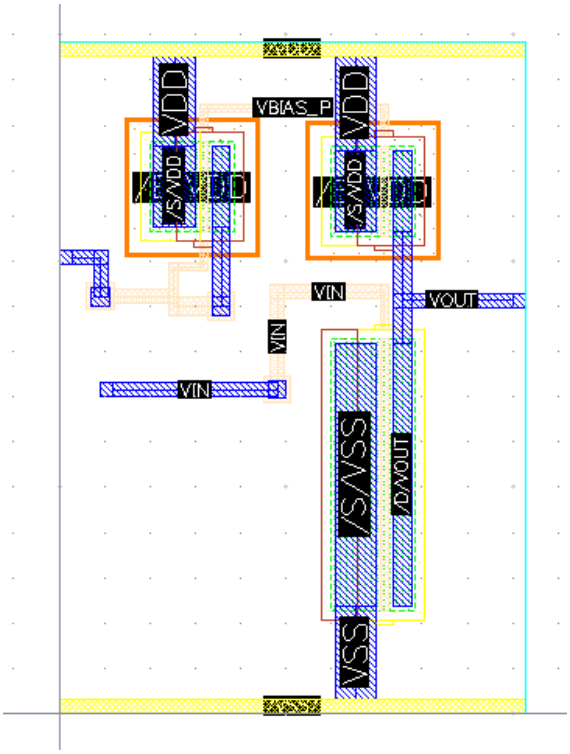


Fig. 9.7: Layout of Common Source Amplifier with PMOS Current Mirror Load

Procedure:

1. Generate Layout by launching Layout XL.
2. Get the layers by connectivity – generate all from source.
3. Connect all layers by following the Lambda based design rules.
4. Do physical verification by running DRC – Design Rule Check, ERC – Electric Rule Check, LVS - Layout vs Schematic
5. Extract RC from the Layout

Sample window of DRC:

Fig. 9.8: DRC result

EXPERIMENT 10 - TWO STAGE OPERATIONAL AMPLIFIER

(a) Capture the Schematic of a 2 – Stage Operational Amplifier and measure the following:

1. **UGB**
2. **dB Bandwidth**
3. **Gain Margin and Phase Margin with and without coupling capacitance**
4. **Use the Op-Amp in the Inverting and Non-Inverting configuration and verify its functionality**
5. **Study the UGB, 3 dB Bandwidth, Gain and Power Requirement in Op-Amp by varying the stage wise transistor geometries and record the observations**

Objective: To understand the design of Op Amp and Study the UGB, 3 dB Bandwidth, Gain and Power Requirement in Op-Amp by varying the stage wise transistor geometries

Software / Tool: Cadence/ Virtuoso

Circuit Diagram: Use 45nm technology

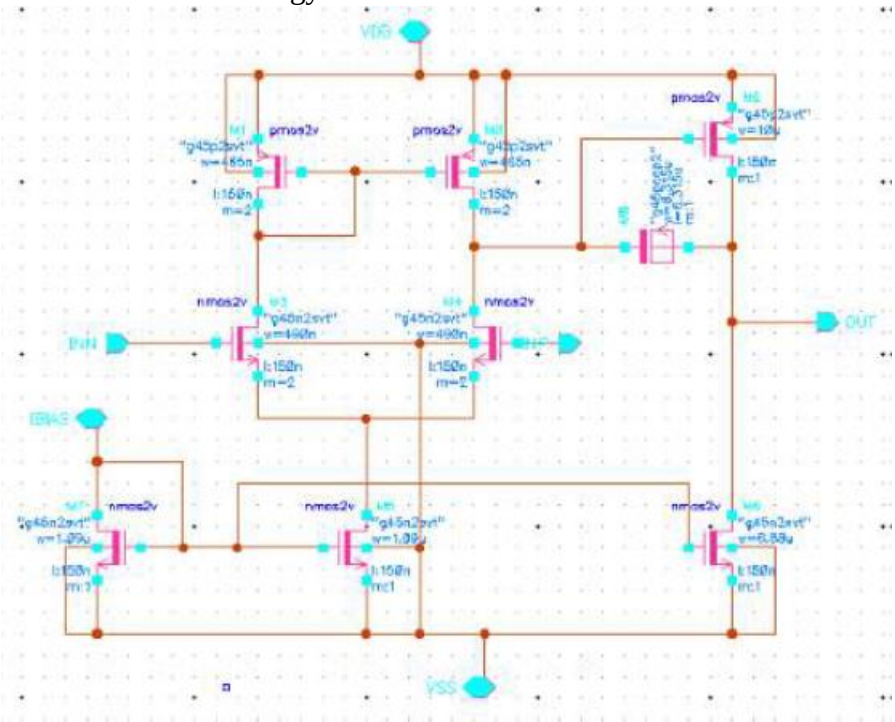


Fig. 10.1: Schematic of Two Stage Op Amp

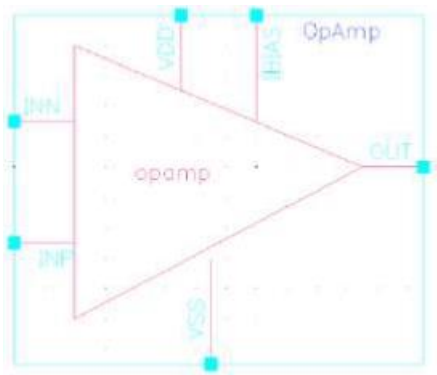


Fig. 10.2: Symbol for 2 – Stage Operational Amplifier

Table 4: Parameters for the devices used in Test

Library Name	Cell Name	Comments / Properties
analogLib	vdc	DC voltage = vdd V
analogLib	vdc	DC voltage = vss V
analogLib	vpulse	Voltage 1 = vdc + 0.3 V, Voltage 2 = vdc - 0.3 V, Period = 10u s, Rise time = 10p s, Fall time = 10p s
analogLib	idc	DC current = ibias A
analogLib	cap	Capacitance = CL F
analogLib	gnd	

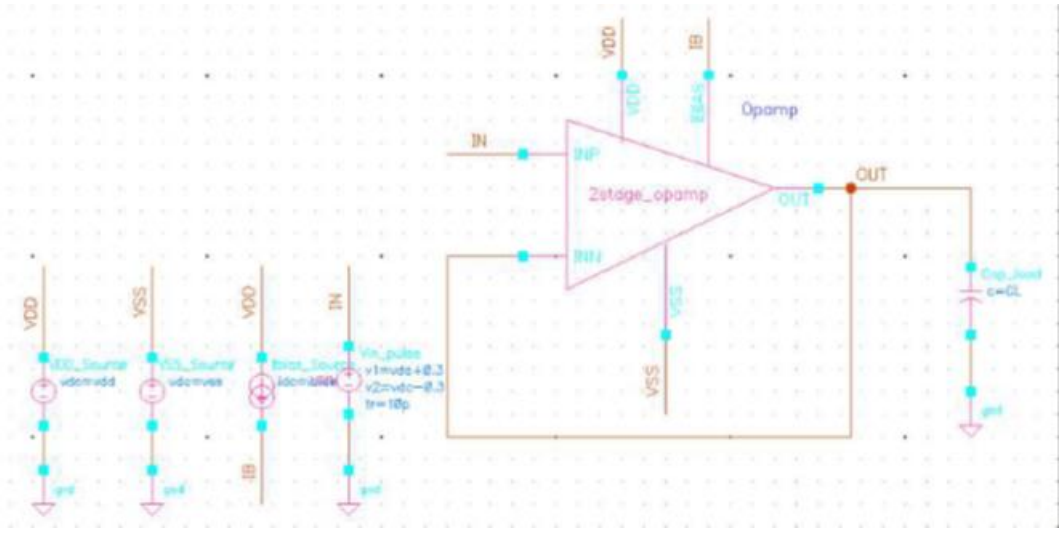


Fig. 10.3: Test Circuit for2 – Stage Operational Amplifier

Procedure

1. Launch ADE Explorer after designing the circuit in schematic editor.
2. Perform transient and DC analysis.
3. Through Virtuoso Visualization & Analysis XL calculator, calculate the parameters and perform AC Analysis.
4. The specification that has to be achieved on simulating the design are as follows:
   - Slew Rate >= 50 MV/s
   - DC Open Loop Gain >= 60 dB (1000 V/V)
   - Unity Gain Bandwidth >= 50 MHz
   - Output Offset <= ± 10 mV
   - Settling Time <= 50 ns

Waveforms

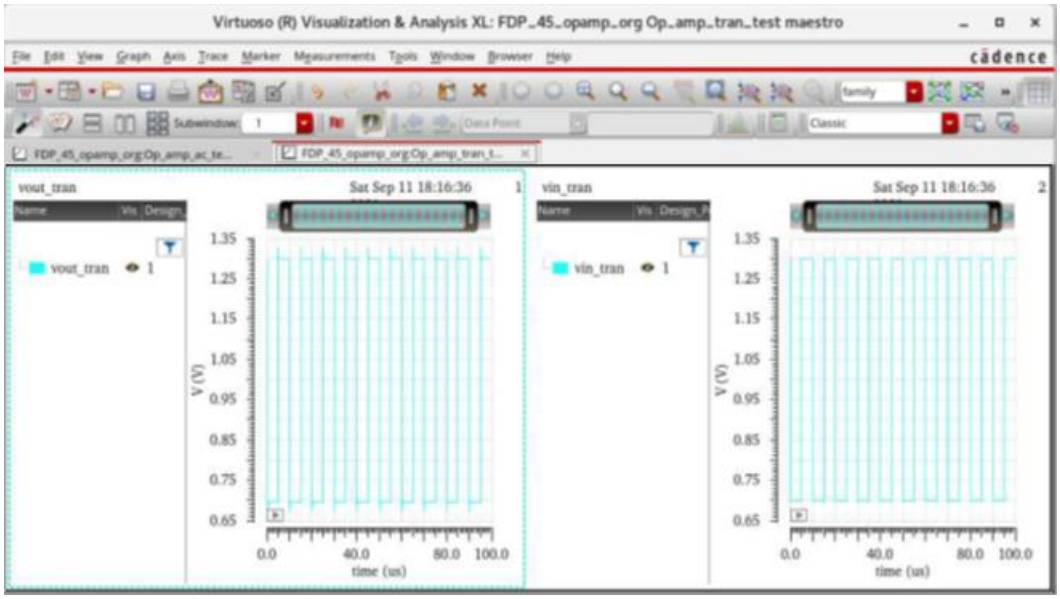


Fig. 10.4: Plotted Waveform for 2 – Stage Operational Amplifier

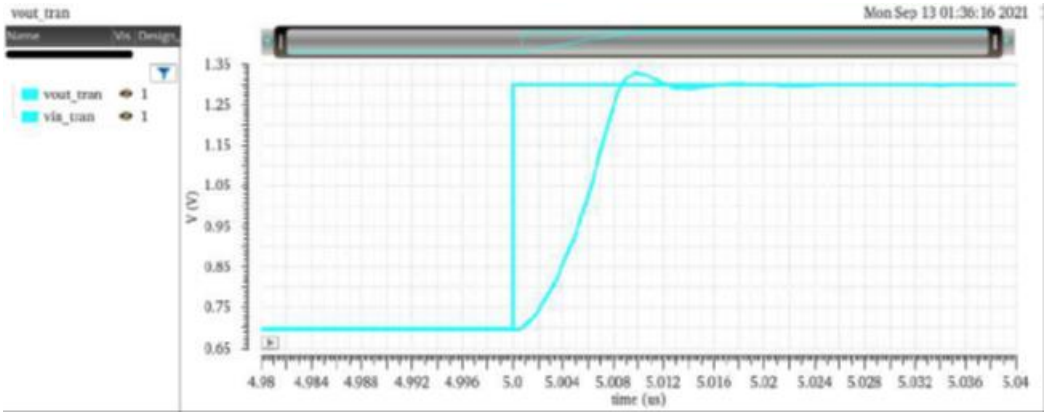


Fig. 10.5: Combined Waveform for 2 – Stage Operational Amplifier

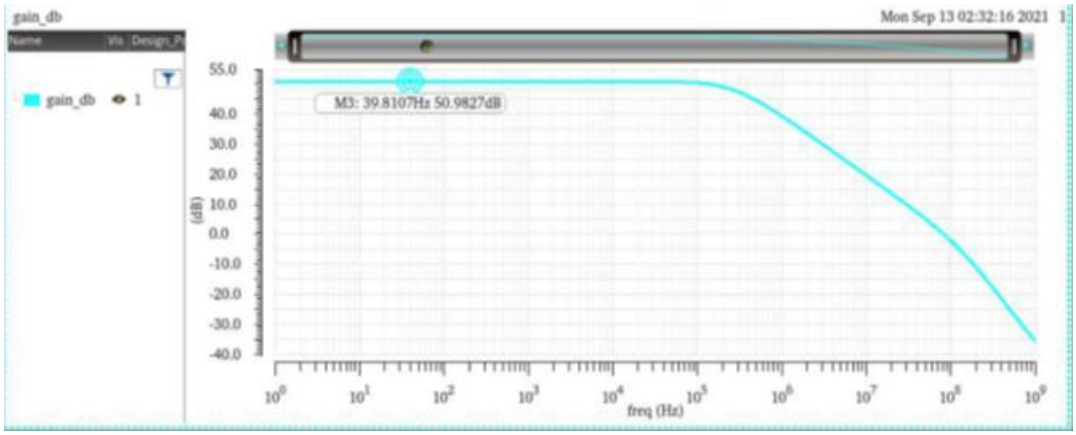


Fig. 10.6: DC Open Loop Gain – 50.98 dB

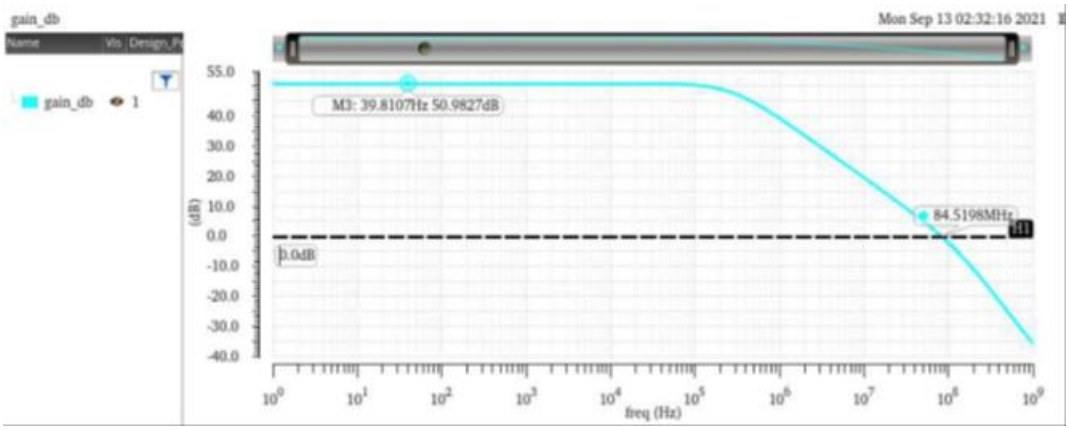


Fig. 10.7: Unity Gain Bandwidth – 84.51M Hz

Observation

Sl. No.	Parameter	Value	
1	UGB		
2	3 dB Bandwidth		
3	Gain margin		
4	Phase margin		
5	Power		

(b) Draw the layout of 2 – stage Operational Amplifier with the maximum transistor width set to 300 (in 180 / 90/ 45n m Technology), choose appropriate transistor geometries as per the results obtained in 4(a). Use optimum layout methods. Verify DRC and LVS, extract the parasitics and perform the post layout simulation, compare the results with pre layout simulations. Record the observations.

Objective: To understand and perform DRC, ERC, LVS after drawing the Layout by following Lambda based design rules

Software / Tool: Cadence/ Virtuoso, Assura

Layout Diagram:

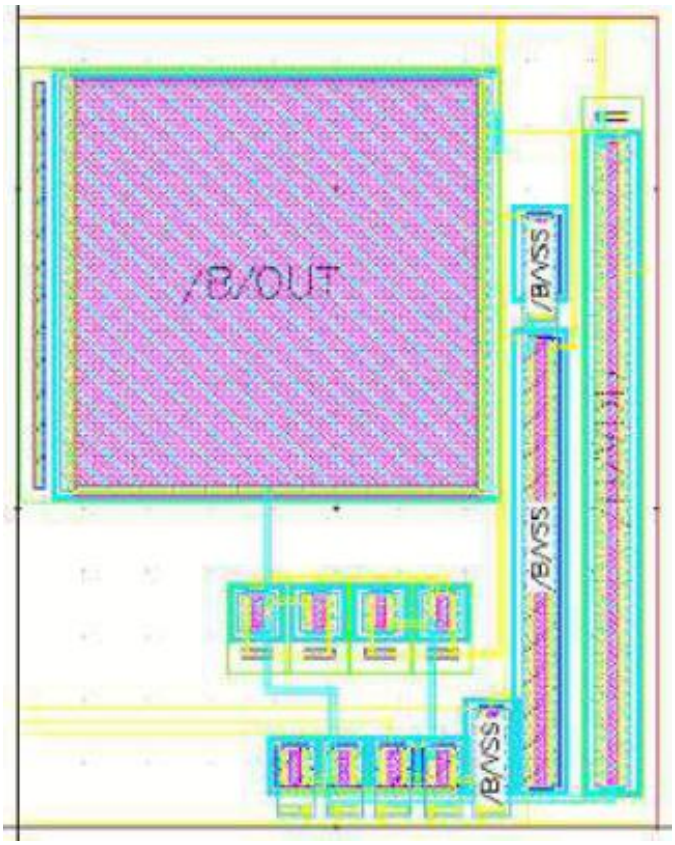


Fig. 10.8: Layout for 2 – Stage Operational Amplifier

Procedure:

1. Generate Layout by launching Layout XL.
2. Get the layers by connectivity – generate all from source.
3. Connect all layers by following the Lambda based design rules.
4. Do physical verification by running DRC – Design Rule Check, ERC – Electric Rule Check, LVS - Layout vs Schematic
5. Extract RC from the Layout

Sample window of DRC:

Fig. 10.9: DRC result

EXPERIMENT 11 – UART (DEMONSTRATION)

Objective: To write a verilog code for UART and carry out the following:

- To Verify the Functionality using test Bench
- Synthesize Design using constraints
- Tabulate Reports using various Constraints
- Identify Critical Path and calculate Max Operating Frequency

Software / Tool: Cadence/ Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)
Synthesis: Genus

Block Diagram:

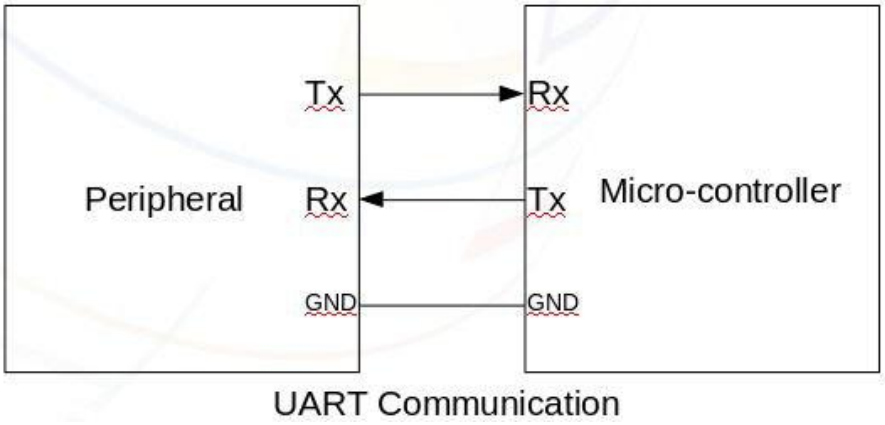


Fig. 11.1: Block Diagram of UART

Verilog code UART:

Transmitter

```
// This code contains the UART Transmitter. This transmitter is able to transmit 8 bits of serial data,
//one start bit, one stop bit and no parity bit. When transmit is complete o_Tx_done will be driven
high //for one clock cycle. Set Parameter CLKS_PER_BIT as follows:
// CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
// Example: 25 MHz Clock, 115200 baud UART
// (25000000)/(115200) = 217
        module UART_TX
        #(parameter CLKS_PER_BIT = 217)
        (inputi_Clock,
        inputi_TX_DV,
        input [7:0] i_TX_Byte,
        outputo_TX_Active,
        outputrego_TX_Serial,
        outputo_TX_Done);
        parameter IDLE = 3'b000;
        parameter TX_START_BIT = 3'b001;
        parameter TX_DATA_BITS = 3'b010;
        parameter TX_STOP_BIT = 3'b011;
        parameter CLEANUP = 3'b100;
        reg [2:0] r_SM_Main = 0;
        reg [7:0] r_Clock_Count = 0;
        reg [2:0] r_Bit_Index = 0;
```

```
reg [7:0] r_TX_Data = 0;
regr_TX_Done = 0;
regr_TX_Active = 0;
always @(posedgei_Clock)
begin
case (r_SM_Main)
IDLE :
begin
o_TX_Serial<= 1'b1; // Drive Line High for Idle
r_TX_Done<= 1'b0;
r_Clock_Count<= 0;
r_Bit_Index<= 0;
if (i_TX_DV == 1'b1)
begin
r_TX_Active<= 1'b1;
r_TX_Data<= i_TX_Byte;
r_SM_Main<= TX_START_BIT;
end
else
r_SM_Main<= IDLE;
end // case: IDLE
// Send out Start Bit. Start bit = 0
TX_START_BIT :
begin
o_TX_Serial<= 1'b0;
// Wait CLKS_PER_BIT-1 clock cycles for start bit to finish
if (r_Clock_Count< CLKS_PER_BIT-1)
begin
r_Clock_Count<= r_Clock_Count + 1;
r_SM_Main<= TX_START_BIT;
end
else
begin
r_Clock_Count<= 0;
r_SM_Main<= TX_DATA_BITS;
end
end // case: TX_START_BIT
// Wait CLKS_PER_BIT-1 clock cycles for data bits to finish
TX_DATA_BITS :
begin
o_TX_Serial<= r_TX_Data[r_Bit_Index];
if (r_Clock_Count< CLKS_PER_BIT-1)
begin
r_Clock_Count<= r_Clock_Count + 1;
r_SM_Main<= TX_DATA_BITS;
end
else
begin
r_Clock_Count<= 0;
// Check if we have sent out all bits
if (r_Bit_Index< 7)
begin
r_Bit_Index<= r_Bit_Index + 1;
```

```
r_SM_Main<= TX_DATA_BITS;
end
else
begin
r_Bit_Index<= 0;
r_SM_Main<= TX_STOP_BIT;
end
end
end // case: TX_DATA_BITS
// Send out Stop bit. Stop bit = 1
TX_STOP_BIT :
begin
o_TX_Serial<= 1'b1;
// Wait CLKS_PER_BIT-1 clock cycles for Stop bit to finish
if (r_Clock_Count< CLKS_PER_BIT-1)
begin
r_Clock_Count<= r_Clock_Count + 1;
r_SM_Main<= TX_STOP_BIT;
end
else
begin
r_TX_Done<= 1'b1;
r_Clock_Count<= 0;
r_SM_Main<= CLEANUP;
r_TX_Active<= 1'b0;
end
end // case: TX_STOP_BIT
// Stay here 1 clock
CLEANUP :
begin
r_TX_Done<= 1'b1;
r_SM_Main<= IDLE;
end
default :
r_SM_Main<= IDLE;
endcase
end
assigno_TX_Active = r_TX_Active;
assigno_TX_Done = r_TX_Done;
endmodule
```

Receiver

```
// This file contains the UART Receiver. This receiver is able to receive 8 bits of serial data, one
start //bit, one stop bit, and no parity bit. When receive is complete o_rx_dv will be driven high for
one //clock cycle. Set Parameter CLKS_PER_BIT as follows:
// CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
// Example: 25 MHz Clock, 115200 baud UART
// (25000000)/(115200) = 217
module UART_RX
#(parameter CLKS_PER_BIT = 217)
(
inputi_Clock,
inputi_RX_Serial,
```

```
outputo_RX_DV,
output [7:0] o_RX_Byte
);
parameter IDLE = 3'b000;
parameter RX_START_BIT = 3'b001;
parameter RX_DATA_BITS = 3'b010;
parameter RX_STOP_BIT = 3'b011;
parameter CLEANUP = 3'b100;
reg [7:0] r_Clock_Count = 0;
reg [2:0] r_Bit_Index = 0; //8 bits total
reg [7:0] r_RX_Byte = 0;
regr_RX_DV = 0;
reg [2:0] r_SM_Main = 0;
// Purpose: Control RX state machine
always @(posedgei_Clock)
begin
case (r_SM_Main)
IDLE :
begin
r_RX_DV<= 1'b0;
r_Clock_Count<= 0;
r_Bit_Index<= 0;
if (i_RX_Serial == 1'b0) // Start bit detected
r_SM_Main<= RX_START_BIT;
else
r_SM_Main<= IDLE;
end
// Check middle of start bit to make sure it's still low
RX_START_BIT :
begin
if (r_Clock_Count == (CLKS_PER_BIT-1)/2)
begin
if (i_RX_Serial == 1'b0)
begin
r_Clock_Count<= 0; // reset counter, found the middle
r_SM_Main<= RX_DATA_BITS;
end
else
r_SM_Main<= IDLE;
end
else
begin
r_Clock_Count<= r_Clock_Count + 1;
r_SM_Main<= RX_START_BIT;
end
end // case: RX_START_BIT
// Wait CLKS_PER_BIT-1 clock cycles to sample serial data
RX_DATA_BITS :
begin
if (r_Clock_Count< CLKS_PER_BIT-1)
begin
r_Clock_Count<= r_Clock_Count + 1;
r_SM_Main<= RX_DATA_BITS;
```

```
end
else
begin
r_Clock_Count<= 0;
r_RX_Byte[r_Bit_Index] <= i_RX_Serial;
// Check if we have received all bits
if (r_Bit_Index< 7)
begin
r_Bit_Index<= r_Bit_Index + 1;
r_SM_Main<= RX_DATA_BITS;
end
else
begin
r_Bit_Index<= 0;
r_SM_Main<= RX_STOP_BIT;
end
end
end // case: RX_DATA_BITS
// Receive Stop bit. Stop bit = 1
RX_STOP_BIT :
begin
// Wait CLKS_PER_BIT-1 clock cycles for Stop bit to finish
if (r_Clock_Count< CLKS_PER_BIT-1)
begin
r_Clock_Count<= r_Clock_Count + 1;
r_SM_Main<= RX_STOP_BIT;
end
else
begin
r_RX_DV<= 1'b1;
r_Clock_Count<= 0;
r_SM_Main<= CLEANUP;
end
end // case: RX_STOP_BIT
// Stay here 1 clock
CLEANUP :
begin
r_SM_Main<= IDLE;
r_RX_DV<= 1'b0;
end
default :
r_SM_Main<= IDLE;
endcase
end
assigno_RX_DV = r_RX_DV;
assigno_RX_Byte = r_RX_Byte;
endmodule // UART_RX
```

Creating Test bench:

Test Bench for UART

```
// Thistestbench will exercise the UART RX.
// It sends out byte 0x37, and ensures the RX receives it correctly.
```

```
`timescale 1ns/10ps
`include "uart_tx.v"
`include "uart_rx.v"
module UART_TB ();
// Testbench uses a 25 MHz clock
// Want to interface to 115200 baud UART
// 25000000 / 115200 = 217 Clocks Per Bit.
parameterc_CLOCK_PERIOD_NS = 40;
parameterc_CLKS_PER_BIT = 217;
parameterc_BIT_PERIOD = 8600;
regr_Clock = 0;
regr_TX_DV = 0;
wirew_TX_Active, w_UART_Line;
wirew_TX_Serial;
reg [7:0] r_TX_Byte = 0;
wire [7:0] w_RX_Byte;
UART_RX #(.CLKS_PER_BIT(c_CLKS_PER_BIT)) UART_RX_Inst
(.i_Clock(r_Clock),
.i_RX_Serial(w_UART_Line),
.o_RX_DV(w_RX_DV),
.o_RX_Byte(w_RX_Byte)
);
UART_TX #(.CLKS_PER_BIT(c_CLKS_PER_BIT)) UART_TX_Inst
(.i_Clock(r_Clock),
.i_TX_DV(r_TX_DV),
.i_TX_Byte(r_TX_Byte),
.o_TX_Active(w_TX_Active),
.o_TX_Serial(w_TX_Serial),
.o_TX_Done()
);
// Keeps the UART Receive input high (default) when
// UART transmitter is not active
assignw_UART_Line = w_TX_Active ?w_TX_Serial : 1'b1;
always
#(c_CLOCK_PERIOD_NS/2) r_Clock<= !r_Clock;
// Main Testing:
initial
begin
// Tell UART to send a command (exercise TX)
@(posedger_Clock);
@(posedger_Clock);
r_TX_DV<= 1'b1;
r_TX_Byte<= 8'h3F;
@(posedger_Clock);
r_TX_DV<= 1'b0;
end
endmodule
```

Theory:

- The UART is “Universal Asynchronous Receiver/Transmitter”, and it is an inbuilt IC with in a microcontroller but not like a communication protocol (I2C & SPI).
- The main function of UART is to serial data communication. In UART, the communication

between two devices can be done in two ways namely serial data communication and parallel data communication.

- The transmitter section includes three blocks namely transmit hold register, shift register and also control logic.
- Likewise, the receiver section includes a receive hold register, shift register, and control logic.
- These two sections are commonly provided by a baud-rate-generator. This generator is used for generating the speed when the transmitter section & receiver section has to transmit or receive the data.

Procedure:

1. Write the code and save in a folder.
2. Run the simulation and note down the results.

Waveforms:

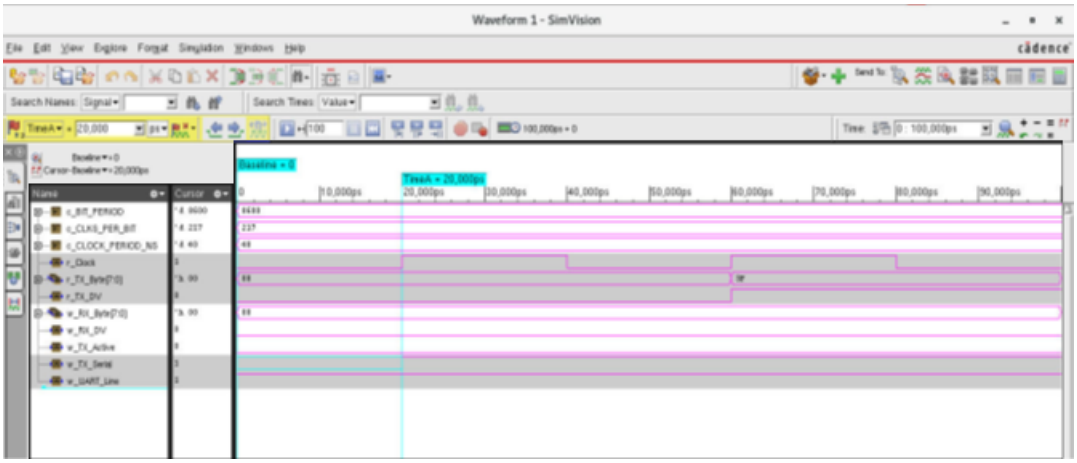


Fig. 11.2: Output Waveform of UART

b) Synthesize the design targeting suitable library and by setting area and timing

1. read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdl {uart_tx.v / uart_rx.v} //Choose any one
3. elaborate
4. read_sdcconstraints_top.sdc //Reading Top Level SDC
5. set_dbsyn_generic_effort medium //Setting effort medium
6. set_dbsyn_map_effort medium
7. set_dbsyn_opt_effort medium
8. syn_generic
9. syn_map
10. syn_opt //Performing Synthesis Mapping and Optimisation
11. report_timing>uart_timing.rep
    //Generates Timing report for worst datapath and dumps into file
12. report_area>uart_area.rep
    //Generates Synthesis Area report and dumps into a file
13. report_power>uart_power.rep
    //Generates Power Report [Pre-Layout]
14. report_qor>uart_qor.rep
15. write_hdl>uart_netlist.v
    //Creates readable Netlist File
16. write_sdc>uart_sdc.sdc

//Creates Block Level SDC

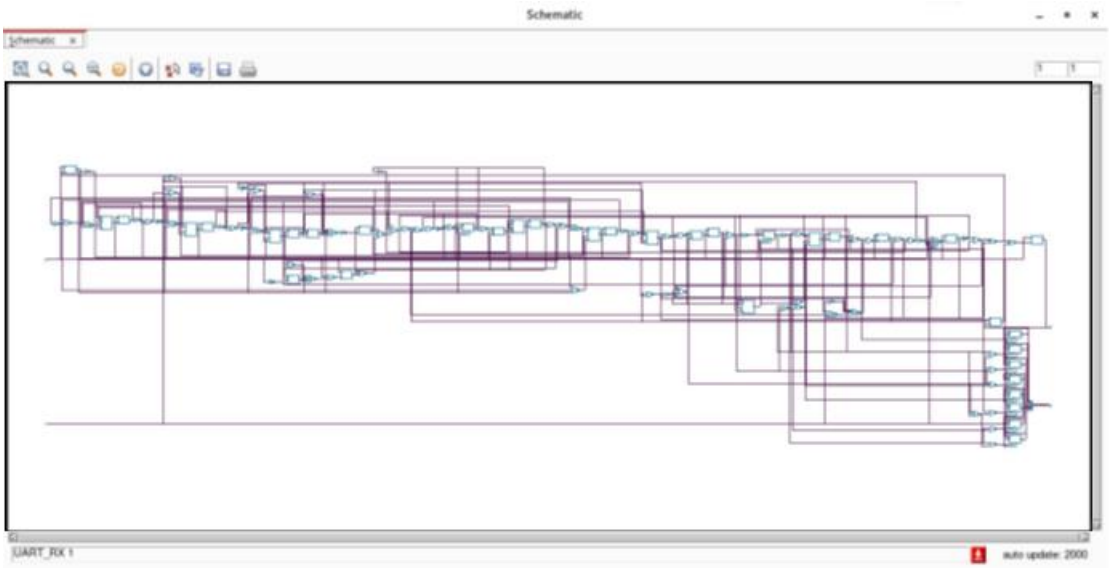


Fig. 11.3: Schematic capture of UART

EXPERIMENT 12 – 6T SRAM (DEMONSTRATION)

Objective: To Design and characterize 6T binary SRAM cell and measure the following:

- Read Time, Write Time, SNM, Power
- Draw Layout of 6T SRAM, use optimum layout methods. Verify for DRC & LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.

Software / Tool: Cadence/ Virtuoso

Circuit Diagram:

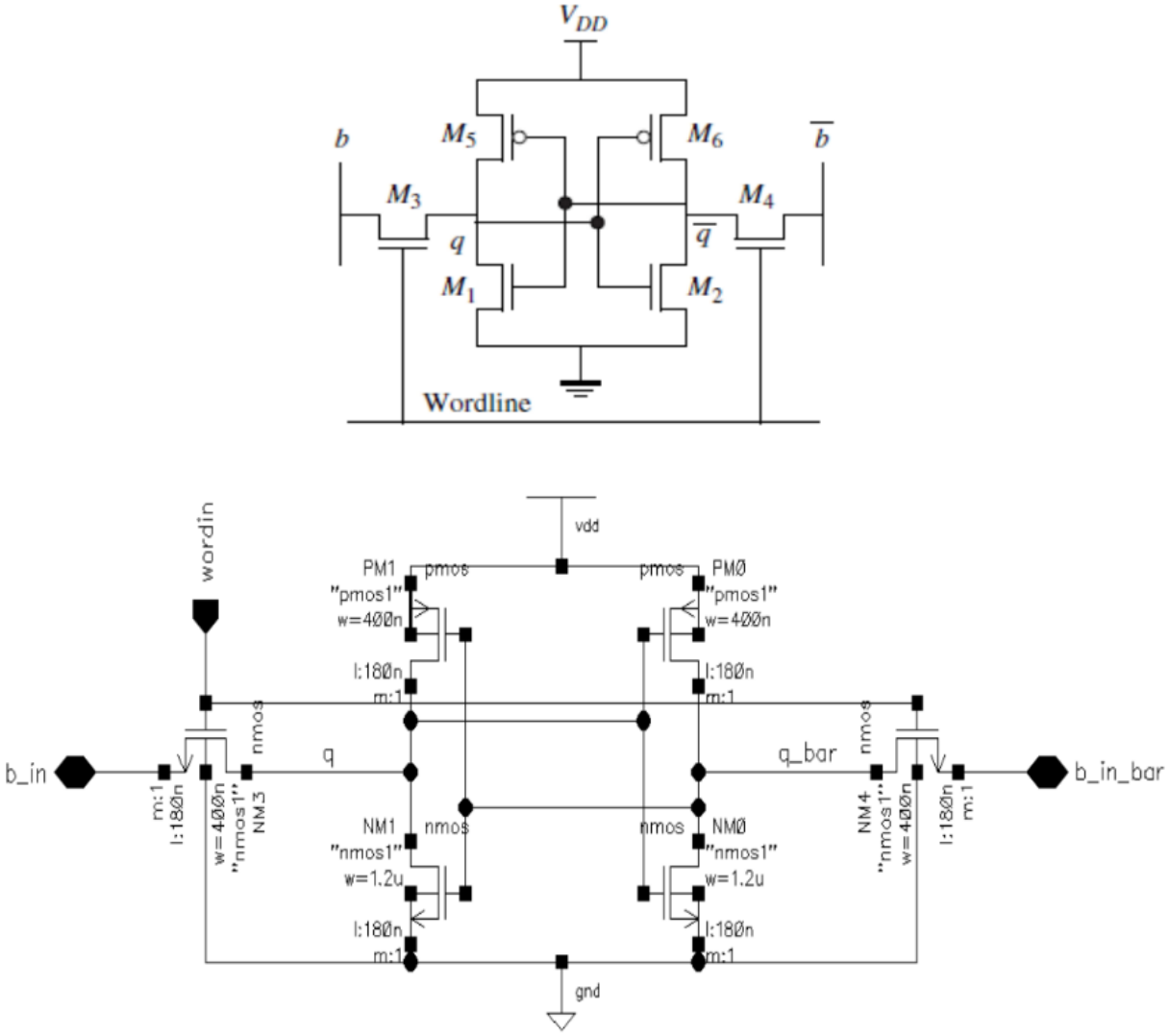


Fig. 12.1: Schematic of 6T SRAM

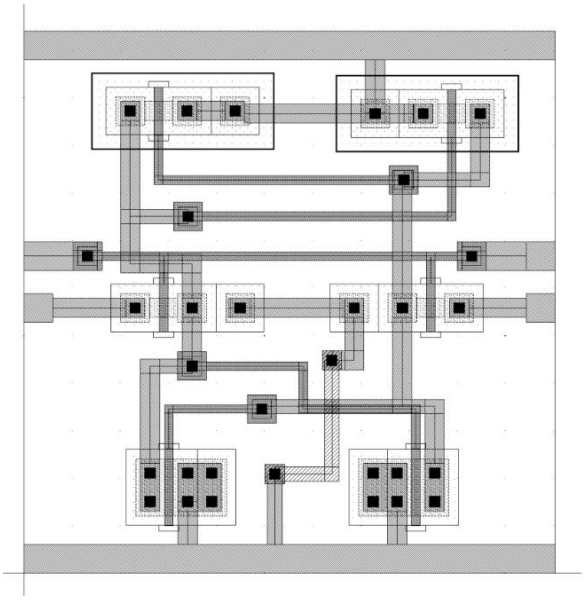


Fig. 12.2: Layout of 6T SRAM

Procedure:

1. Invoke the software and create new folder.
2. Using appropriate libraries design the circuit and save.
3. Create the symbol of the circuit designed and save.
4. Using appropriate libraries design the test circuit and save.
5. Run the simulation for performing Transient Analysis.
6. Verify Read and Write operation by simulating

Waveforms:

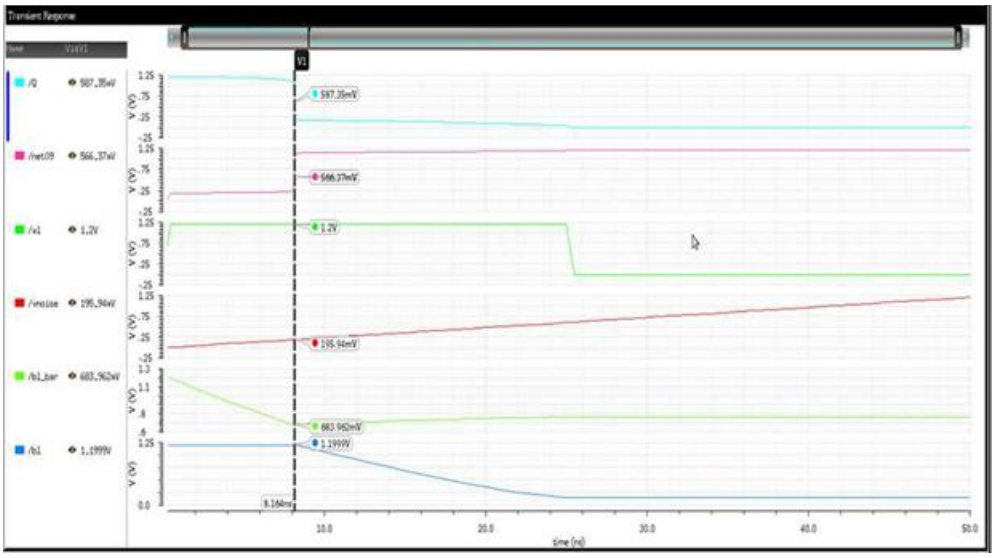


Fig. 12.3: Transient Analysis of 6T SRAM

Sample window of LVS:

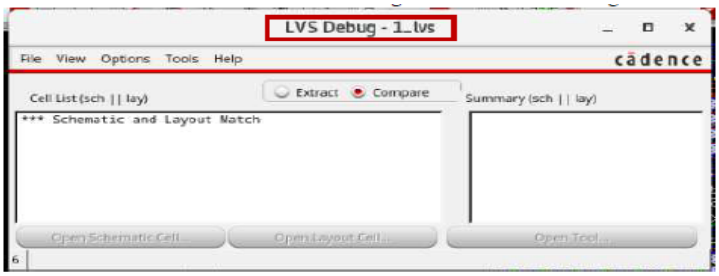


Fig. 12.4: LVS result of 6T SRAM

VIRTUAL EXPERIMENT – Schematic Design of Pass Transistor Logic & Multiplexer

Objectives:

- To design positive level pass transistor logic.
- To design a 2-input multiplexer using pass transistor logic for following logical expression: In1CLK' + In2CLK
- To design negative level pass transistor logic.

Software / Tool: Cadence/ Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)

Block Diagram:

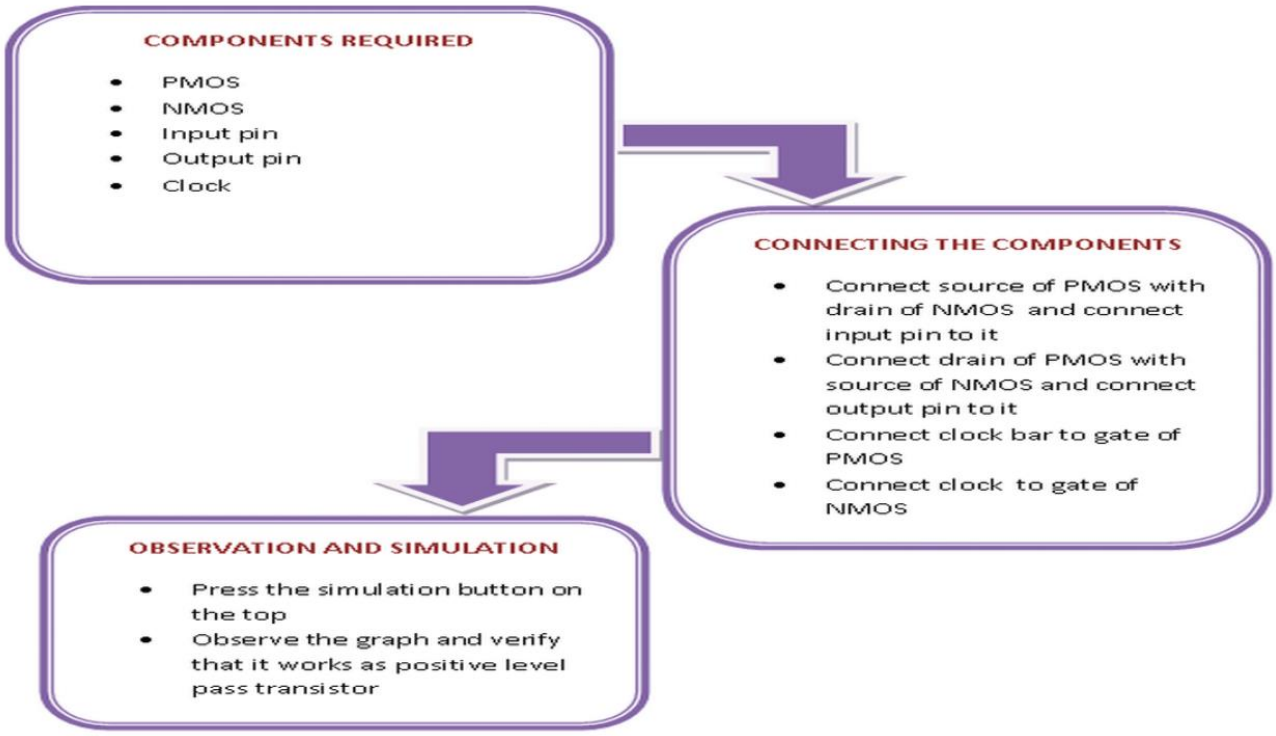


Fig. 13.1: Positive Level Pass Transistor Logic

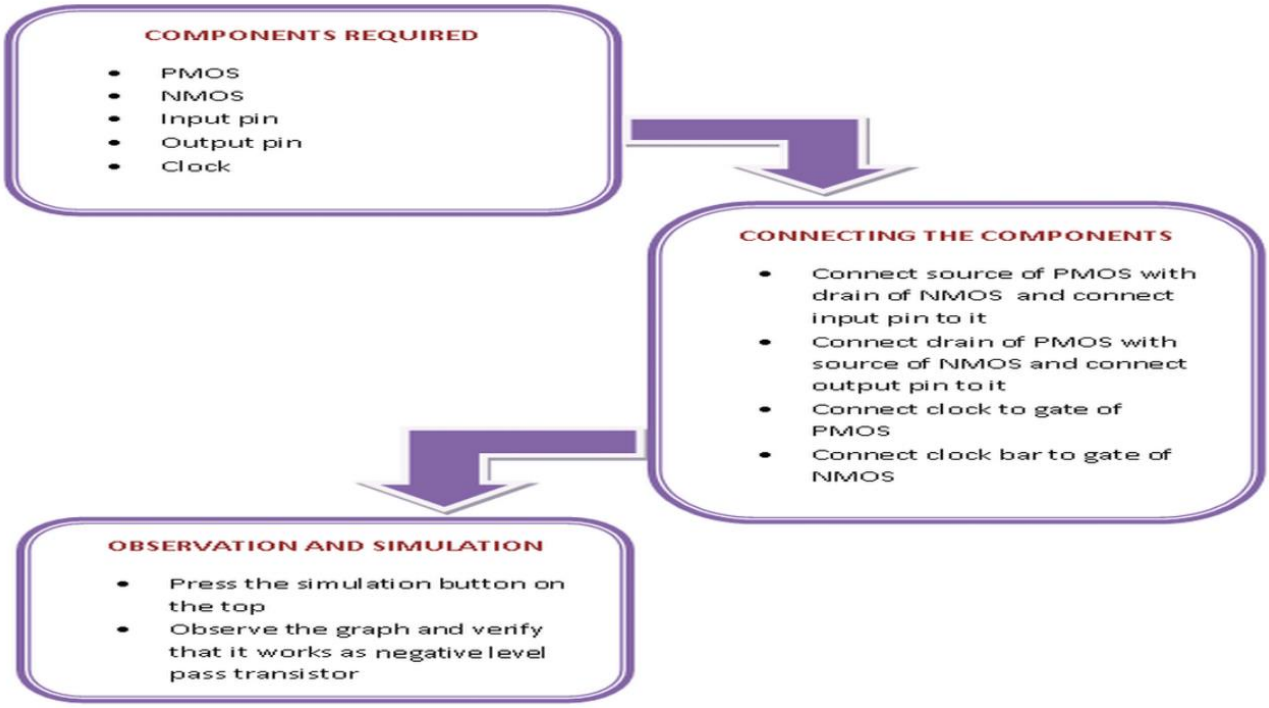


Fig. 13.2: Negative Level Pass Transistor Logic

Fig. 13.3: Multiplexer using Pass Transistor Logic

Link: Virtual Labs

KLS VISHWANATHRAO DESHPANDE INSTITUTE OF TECHNOLOGY,

Haliyal – 581 329, Uttara Kannada

Accredited by NAAC with "A" Grade

Department of Electronics & Communication Engineering

LABORATORY MANUAL

Course Title	:	**IoT (Internet of Things) Lab**
Course Code	:	**BEC657C**
Year / Semester	:	**3rd Year / 6th Semester**
Academic Year	:	**Even Semester of 2025 - 2026**
Course In-Charge	:	**Prof. Raghavendra N.**

College Vision and Mission Statements

Vision
To nurture talent & enrich society through excellence in technical education, research & innovation.
Mission
1. To augment innovative pedagogy & kindle quest for interdisciplinary learning & to enhance conceptual understanding. 2. To build competence, professional ethics & develop entrepreneurial thinking. 3. To strengthen industry institute partnership & explore global collaborations. 4. To inculcate culture of socially responsible citizenship. 5. To focus on holistic & sustainable development.

Department Vision, Mission,PEOs and PSOs Statements

Vision
To bring out talented, skilled, and sustainable Electronics and Communication Engineering Graduates through strong domain expertise to serve the Society with greater Professional Ethics.
Mission
1. To create and impart an active learning ambience to accomplish a high degree of Professional competencies 2. To inculcate innovative research and developmental thinking in effective Teaching and Learning processes for solving Societal challenges 3. To deliver the needs and requirements of the latest state of art of the Industry through quality multidisciplinary internship and training programs

PEOs	
PEO 1:	To be successful in professional career in electronics, communication and allied industries by acquiring the knowledge in the fundamentals of Electronics and Communication Engineering principles and professional skills.
PEO 2:	To be in a position to analyze real life problems and design socially accepted and economically feasible solutions in the respective fields.
PEO 3:	To exhibit good communication skills in their professional career, lead a team with good leadership traits and good interpersonal relationship with the members related to other engineering streams.
PEO 4:	To involve themselves in lifelong learning and professional development by pursuing higher education and participation in research and development activities.
PEO 5:	To demonstrate professional and ethical responsibilities towards their profession, society and the environment.
PSOs	
PSO 1:	An ability to use appropriate modern techniques for analysis, design and development of VLSI and Embedded Systems.
PSO 2:	Understand the architectural specifications of a communication system and determine their performance.

Course Title	**:**	IoT (Internet of Things) Lab
Course Code	**:**	BEC657C
Year / Semester	**:**	3rd Year / 6th Semester
Academic Year	**:**	Even Semester of 2025 – 2026

Sl. No.	Content	Page No.
1	To interface LED/Buzzer with Arduino/Raspberry Pi and write a program to 'turn ON' LED for 1 sec after every 2 seconds.	10
	To interface Push button/Digital sensor (IR/LDR) with Arduino/Raspberry Pi and write a program to 'turn ON' LED when push button is pressed or at sensor detection.	12
2	To interface DHT11 sensor with Arduino/Raspberry Pi and write a program to print temperature and humidity readings.	14
	To interface OLED with Arduino/Raspberry Pi and write a program to print temperature and humidity readings on it.	17
3	To interface motor using relay with Arduino/Raspberry Pi and write a program to 'turn ON' motor when push button is pressed.	21
4	Write an Arduino/Raspberry Pi program to interface the Soil Moisture Sensor.	23
	Write an Arduino/Raspberry Pi program to interface the LDR/Photo Sensor.	25
5	Write a program to interface an Ultrasonic Sensor with Arduino /Raspberry Pi.	27
6	Write a program on Arduino/Raspberry Pi to upload temperature and humidity data to thingspeak cloud.	29
7	Write a program on Arduino/Raspberry Pi to retrieve temperature and humidity data from thingspeak cloud.	33
8	Write a program to interface LED using Telegram App.	36
9	Write a program on Arduino/Raspberry Pi to publish temperature data to MQTT broker.	42
10	Write a program to create UDP server on Arduino/Raspberry Pi and respond with humidity data to UDP client when requested.	46
11	Write a program to create TCP server on Arduino/Raspberry Pi and respond with humidity data to TCP client when requested.	49
12	Write a program on Arduino/Raspberry Pi to subscribe to MQTT broker for temperature data and print it.	52
13	Virtual Lab	56

CLOs	**:**	• To impart necessary and practical knowledge of components of Internet of Things • To develop skills required to build real-life IoT based projects.
COs	**:**	**CO1:** Explain internet of Things and its hardware and software components **CO2:** Interface I/O devices, sensors & communication modules **CO3:** Remotely monitor data and control devices **CO4:** Develop real life IoT based projects

CO / PO Mapping	**:**	**COs**	**CO1**	**CO2**	**CO2**	**CO4**
		POs / PSOs	**1, 2, 3, 5/1,2**	**1, 2, 3, 5/1,2**	**1, 2, 3, 5/1,2**	**1, 2, 3, 5,6/1,2**

Course In-Charge	**:**	Prof. Raghavendra N.

CO-PO Mapping Matrix

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
1	1	2	1		1								1	1
2	1	2	1		1								1	1
3	1	2	1		1								1	1
4	1	2	1		1	1							1	1

Evaluation Details

Students Assessment through CIE (50%) + SEE (50%)

Parameter	Particulars	Marks	Total	Scale Down Marks
CIA	Performance	05	120 (10*12 Expt)	30
	Viva-Voce	02		
	Journal	03		
LAB IA-1	Write-up + Conduction+ Result	60	100	20
	viva	40		
Total Marks				50

Note:

1. CIA for each lab session will be for 10 Marks
2. Total CIA marks scale down to 30 Marks
3. One lab Internals to be conducted for 100 Marks
4. Lab Internal marks to be scale down to 20 Marks

Introduction to Internet of Things

IoT (Internet of Things) is a revolutionary concept that refers to the network of interconnected physical objects, devices, and sensors that communicate and exchange data over the internet. These objects, often called "things," are embedded with various technologies such as sensors, actuators, and communication interfaces, enabling them to collect, transmit, and receive data. Here's an explanation of IoT:

Key Components and Characteristics of IoT:

1. **Things/Devices:** These are the physical objects equipped with sensors, actuators, and communication modules. These devices can be anything from smart thermostats, wearable fitness trackers, industrial machines, to environmental sensors, and more. They gather data from their surroundings or perform actions based on commands.
2. **Sensors and Actuators:** Sensors collect data from the environment, measuring variables such as temperature, humidity, pressure, motion, and more. Actuators are responsible for carrying out actions based on commands received from the network or other devices, such as turning on/off lights or adjusting the thermostat.
3. **Connectivity:** IoT devices rely on various communication technologies to connect to the internet and other devices. This can include Wi-Fi, cellular networks, Bluetooth, Zigbee, LoRa, or Ethernet, depending on the device's range, power consumption, and bandwidth requirements.
4. **Data Transmission:** Data collected by IoT devices is transmitted to central processing units, such as cloud servers or edge computing platforms. This data can be transmitted in real-time or stored locally for later transmission, depending on the use case.
5. **Data Processing:** The collected data is processed to extract valuable insights, trends, and patterns. This can involve analytics, artificial intelligence, and machine learning to make informed decisions.
6. **Cloud/Edge Computing:** IoT data can be processed either in the cloud (centralized) or at the edge (closer to the device). Edge computing reduces latency and allows for real-time decision-making, while cloud computing offers scalability and accessibility from anywhere.
7. **Applications:** IoT applications are software programs or interfaces that enable users to interact with IoT data and control devices. These can take the form of web-based dashboards, mobile apps, or automation systems.

Key Principles of IoT:

1. **Interconnectivity:** IoT thrives on the idea that virtually any device can be connected to the internet, allowing for data sharing and remote control. This interconnectivity facilitates automation and real-time monitoring.
2. **Sensing and Data Collection:** IoT devices are equipped with sensors that collect data from the physical world. This data can be environmental data, usage statistics, or other relevant information.
3. **Data Analysis:** The collected data is analyzed to derive valuable insights, enabling businesses and individuals to make informed decisions, optimize processes, and improve efficiency.

4. **Remote Control and Automation:** IoT allows for remote monitoring and control of devices. This can range from adjusting home thermostat settings from a smartphone to remotely managing industrial equipment.

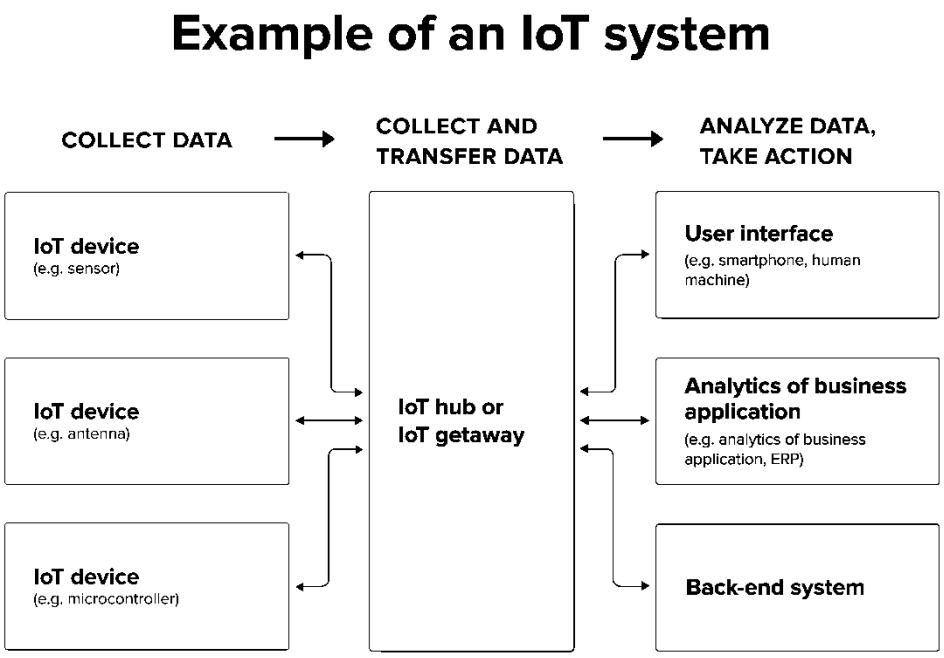
Applications of IoT:

IoT has a wide range of applications across various industries, including:

- **Smart Homes:** Home automation, security systems, and energy management.
- **Smart Cities:** Traffic management, waste management, and environmental monitoring.
- **Industrial IoT (IIoT):** Predictive maintenance, asset tracking, and process optimization.
- **Healthcare:** Remote patient monitoring and medical device connectivity.
- **Agriculture:** Precision farming and livestock monitoring.
- **Logistics and Supply Chain:** Inventory management and asset tracking.

How does Internet of Thing (IoT) Work?

The Internet of Things (IoT) works by connecting physical devices, sensors, and objects to the internet, enabling them to collect, transmit, and receive data. The core principles of how IoT works involve data sensing, connectivity, data processing, and user interaction:



1. **Sensing Data:** IoT devices are equipped with various sensors that capture data from the physical world. These sensors can measure parameters like temperature, humidity, pressure, light, motion, and more. Some devices also include cameras and microphones for visual and audio data.
2. **Data Transmission:** IoT devices use communication technologies to transmit the data they collect to other devices or central processing units. The choice of communication method depends on factors such as range, power consumption, and bandwidth requirements. Common communication technologies include Wi-Fi, cellular networks (3G, 4G, 5G), Bluetooth, Zigbee, LoRa, and more.

3. **Data Processing:** The data collected by IoT devices is sent to central processing units, such as cloud servers or edge computing platforms. Here, the data is processed and analyzed. This processing can include data filtering, aggregation, storage, and, in many cases, the application of machine learning or artificial intelligence algorithms to extract meaningful insights from the data.
4. **Storage:** Processed data is often stored in databases or data lakes, making it accessible for future analysis and historical reference.
5. **Decision-Making:** Based on the processed data, IoT systems can make informed decisions. These decisions can range from simple actions like turning on a light when a motion sensor is triggered to complex decisions in industrial IoT (IIoT), such as predicting equipment maintenance needs.
6. **User Interaction:** IoT applications and interfaces enable users to interact with the system. Users can monitor and control devices, view data through web-based dashboards or mobile apps, and receive alerts or notifications. Some systems also allow voice control or integration with other software applications.
7. **Automation:** IoT can enable automation based on pre-defined rules or data-driven triggers. For example, a smart thermostat can automatically adjust the temperature based on occupancy and ambient conditions.
8. **Security and Privacy:** Security measures are crucial in IoT to protect data, devices, and the network from unauthorized access and potential vulnerabilities. This includes encryption, authentication, and access control.

IoT Ecosystem: IoT systems are part of a larger ecosystem that includes:

- **Devices/Things:** These are the IoT devices and sensors that collect data and perform actions.
- **Connectivity:** Communication networks and protocols that allow devices to transmit data over the internet.
- **Cloud/Edge:** Centralized cloud platforms or edge computing environments for data processing and storage.
- **Applications:** Software interfaces that enable users and other systems to interact with IoT data and devices

IoT Architecture

IoT (Internet of Things) architecture defines the structure and components of an IoT system, outlining how devices, data, and processes interact. It provides a blueprint for building a functional and secure IoT ecosystem. Here's an explanation of the typical architecture of IoT:

1. Devices/Things: These are the physical objects or devices embedded with sensors, actuators, and communication modules.

IoT devices can include a wide range of objects, from environmental sensors to industrial machinery and wearable devices.

They collect data from their surroundings or perform actions based on commands received.

2. Sensors and Actuators: Sensors are responsible for collecting data from the physical world. They measure variables such as temperature, humidity, pressure, motion, light, and more.

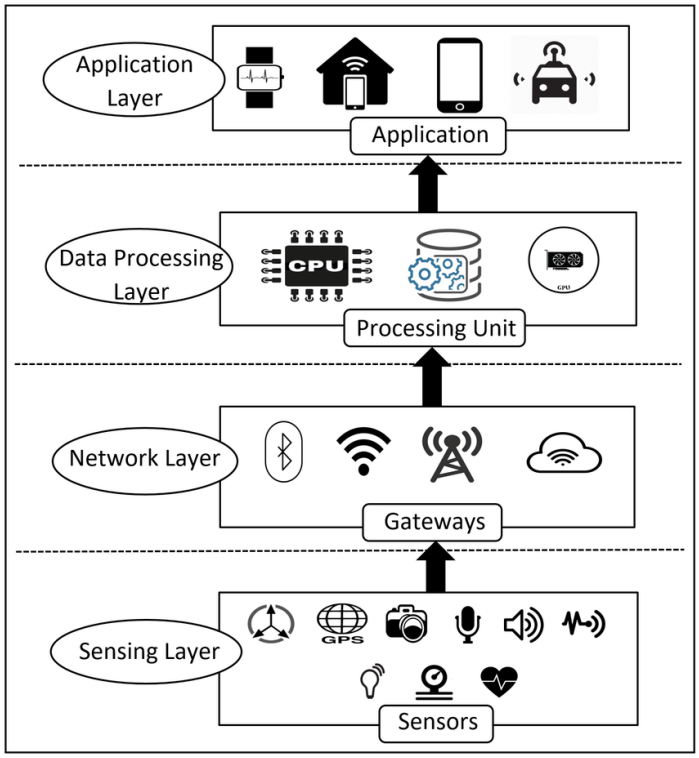
Actuators carry out actions based on commands received from the network or other devices. Examples include turning on lights or adjusting thermostat settings.

3. Connectivity: IoT devices rely on various communication technologies to connect to the internet and other devices. The choice of connectivity depends on the specific use case.

Common IoT communication technologies include Wi-Fi, cellular networks (3G, 4G, 5G), Bluetooth, Zigbee, LoRa, and Ethernet.

4. Data Transmission: Data collected by IoT devices is transmitted to central processing units, such as cloud servers or edge computing platforms.

Data can be transmitted in real-time or stored locally for later transmission, depending on the use



case.

5. Data Processing: Processed data is essential for extracting valuable insights from the collected information. Data processing can involve filtering, aggregation, and applying machine learning algorithms.

Data processing can occur in the cloud (centralized) or at the edge (closer to the device), depending on the architecture and requirements.

6. Cloud/Edge Computing: The data is sent to cloud platforms or edge computing environments for further processing, storage, and analysis.

Cloud computing offers scalability and accessibility from anywhere, while edge computing reduces latency and allows for real-time decision-making.

7. Applications: IoT applications are software programs or interfaces that enable users to interact with IoT data and control devices.

These applications can take the form of web-based dashboards, mobile apps, or automation systems.

8. User Interaction: End-users interact with the IoT system through web interfaces, mobile apps, voice commands, or other interfaces.

Users can monitor data, control devices, and receive alerts or notifications.

9. Security and Privacy: Security measures are crucial to protect against data breaches, unauthorized access, and device tampering. IoT security includes encryption, authentication, access control, and regular updates to address vulnerabilities.

10. External Integration: IoT systems may integrate with other systems, such as enterprise resource planning (ERP) systems, customer relationship management (CRM) systems, or third-party APIs for extended functionality and data sharing.

IoT Protocols

IoT (Internet of Things) protocols are communication standards that enable devices and systems in the IoT ecosystem to exchange data reliably and efficiently. These protocols play a crucial role in ensuring that devices from different manufacturers can communicate and work together seamlessly. Here are some common IoT protocols:

MQTT (Message Queuing Telemetry Transport):

- MQTT is a lightweight and efficient publish-subscribe messaging protocol.
- It is well-suited for IoT due to its low overhead, making it ideal for low-power and low-bandwidth devices.
- Devices can publish messages to topics, and other devices can subscribe to these topics to receive data.

HTTP (Hypertext Transfer Protocol) and HTTPS (HTTP Secure):

- HTTP is a standard protocol for transmitting data over the web.
- IoT devices can communicate with web servers and cloud services using HTTP or its secure version, HTTPS.
- It's commonly used for interactions between IoT devices and cloud platforms or web applications.

CoAP (Constrained Application Protocol):

- CoAP is a lightweight, UDP-based protocol designed for constrained IoT devices.
- It is suitable for low-power, low-data-rate applications, and it is often used in resource-constrained environments.

AMQP (Advanced Message Queuing Protocol):

- AMQP is a robust and reliable messaging protocol that ensures guaranteed message delivery.
- It is suitable for IoT applications that require reliable communication, such as industrial and enterprise IoT use cases.

DDS (Data Distribution Service):

- DDS is a protocol for real-time, data-centric communication between IoT devices and systems.
- It is commonly used in industrial and mission-critical applications where low latency and high reliability are essential.

Bluetooth and Bluetooth Low Energy (BLE):

- Bluetooth is a wireless communication protocol often used in short-range IoT applications.
- BLE, a low-power variant of Bluetooth, is well-suited for battery-operated IoT devices and smartphone connectivity.

LoRaWAN (Long Range Wide Area Network):

- LoRaWAN is designed for long-range, low-power communication.
- It is suitable for IoT applications like remote environmental monitoring and asset tracking.

Zigbee:

- Zigbee is a low-power, low-data-rate wireless communication protocol.
- It is commonly used in home automation and industrial IoT applications.

Modbus:

- Modbus is a communication protocol widely used in industrial automation.
- It is suitable for connecting industrial devices and sensors to control systems and SCADA (Supervisory Control and Data Acquisition) systems.

DDS (Device Device Server):

- DDS is a lightweight protocol that facilitates peer-to-peer communication between IoT devices.
- It is suitable for applications where low latency and efficient data exchange are critical.

IoT Software

Embedded systems have less storage and processing power, their language needs are different. The most commonly used operating systems for such embedded systems are Linux or UNIX-like OSs like Ubuntu Core or Android.

IoT software encompasses a wide range of software and programming languages from general-purpose languages like C++ and Java to embedded-specific choices like Google ‘s Go language or Parasail.

Fig. a: IoT Hardware | IoT Software

Few IoT Software’s are

- **C & C++**: The C programming language has its roots in embedded systems—it even got its start for programming telephone switches. It‘s pretty ubiquitous, that is, it can be used almost everywhere and many programmers already know it. C++ is the object-oriented version of C, which is a language popular for both the Linux OS and Arduino embedded IoT software systems. These languages were basically written for the hardware systems which makes them so easy to use.

- **Java**: While C and C++ are hardware specific, the code in JAVA is more portable. It is more like a write once and read anywhere language, where you install libraries, invests time in writing codes once and you are good to go.
- **Python:** There has been a recent surge in the number of python users and has now become one of the ―go-to‖ languages in Web development. Its use is slowly spreading to the embedded control and IoT world—specially the Raspberry Pi processor. Python is an interpreted language, which is, easy to read, quick to learn and quick to write. Also, it‘s a powerhouse for serving data-heavy applications.

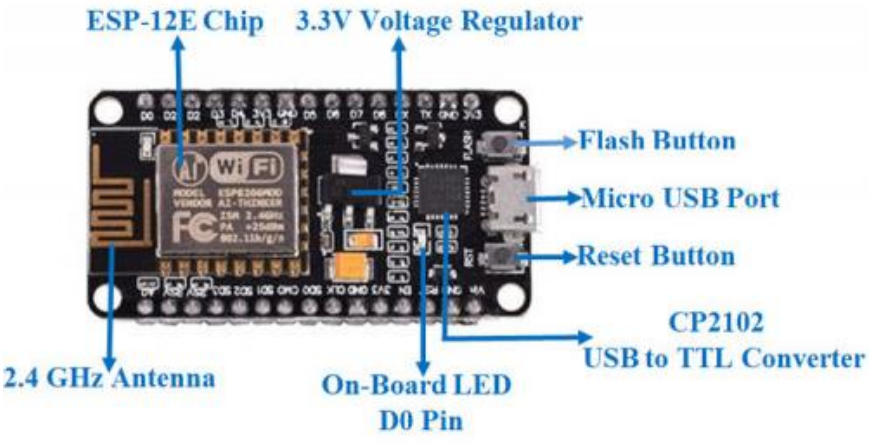
Introduction to Node MCU ESP8266 module

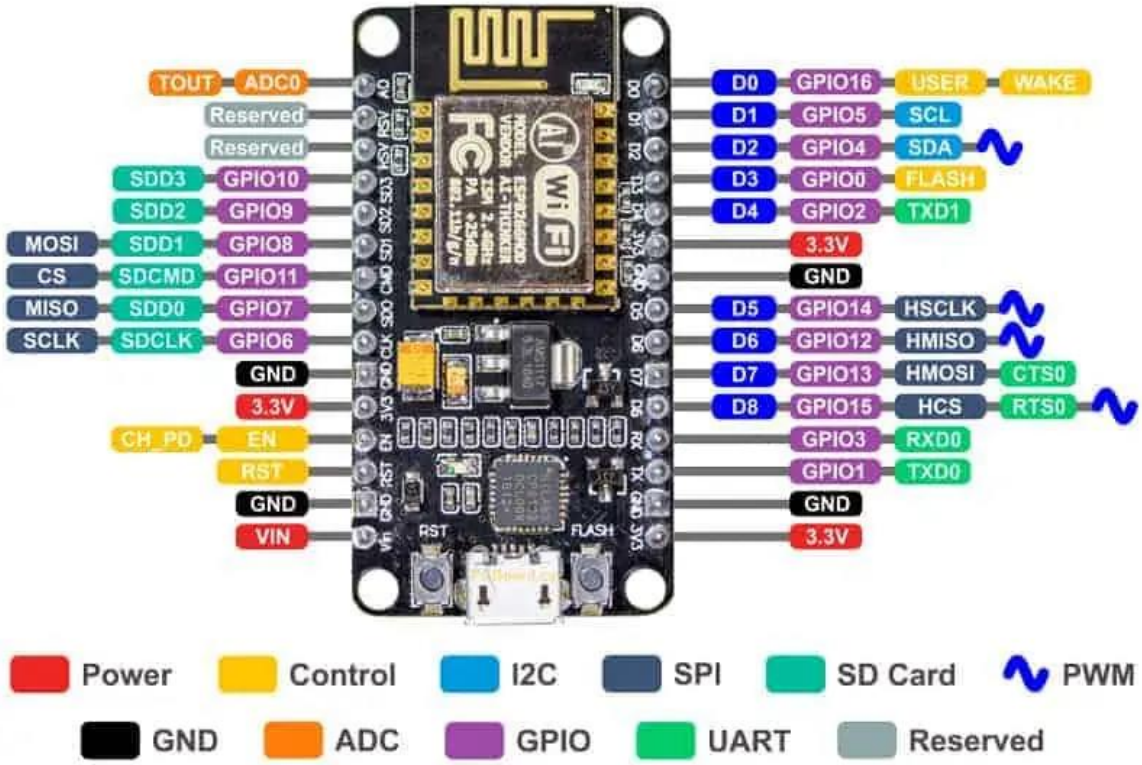
The NodeMCU (*N*ode *M*icro-Controller *U*nit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.

NodeMCU ESP8266 Specifications & Features and Pinout

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

PIN	CODE
A0	A0
GPIO 16	D0
GPIO 5	D1
GPIO 4	D2
GPIO 0	D3
GPIO 2	D4
GPIO14	D5
GPIO 12	D6
GPIO 13	D7
GPIO 15	D8
GPIO 9	SD2
GPIO10	SD3
GPIO3	Rx
GPIO1	Tx





Steps to install Node MCU

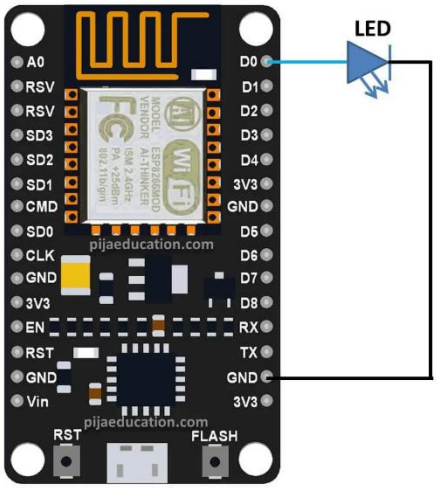
1. Download and install Arduino IDE
2. Open the IDE and follow this path. **File -> preferences -> Additional board manager URL.**
3. Now paste the URL in the dialog box **:** **http://arduino.esp8266.com/stable/package_esp8266com_index.json**
4. Then, click the "**OK**" button.
5. Now follow this path**. Tools -> Board -> Boards Manager**
6. Search for **ESP8266** and install the "**ESP8266 by ESP8266 Community**"
7. After this, restart your Arduino IDE.
8. Then, go to **Tools** > **Board** and check that you have ESP8266 boards available.
9. First, make sure you have an ESP8266 selected in **Tools** > **Board**. If you're using the ESP8266-12E NodeMCU Kit as shown in previous pictures, select the **NodeMCU 1.0 (ESP-12E Module)** option.

Experiment No: 1.a
To interface LED/Buzzer with Arduino and write a program to 'turn ON' LED for 1 sec after every 2 seconds.

Apparatus Required:

Sl No	Components	Quantity
1	Node MCU ESP 8266	1
2	LED or Buzzer	1
3	USB cable for NODE MCU	1
4	Connecting wires	1

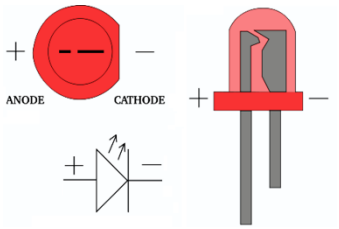
Circuit Diagram:



Pin Connection	
NODE MCU	**LED**
GND	Cathode
D0	Anode

Theory: LED, which stands for Light Emitting Diode, is a semiconductor device that emits light when an electric current passes through it. LEDs are widely used for various applications due to their energy efficiency, long lifespan, and versatility. Here are some key points about LEDs.

Basic Operation: LEDs work on the principle of electroluminescence. When electrons and holes (positive counterparts of electrons) recombine within the semiconductor material, they release energy in the form of photons, which produces light.



Item	Min	Max	Unit
Forward Current	20	30	mA
Forward Voltage	1.8	2.2	V

Program

```
#define led  DO //inbuilt LED Pin
void setup() {
  pinMode(led, OUTPUT);
}
void loop() {
 digitalWrite(led, HIGH);
 delay(1000);
 digitalWrite(led, LOW);
 delay(2000);
}
```

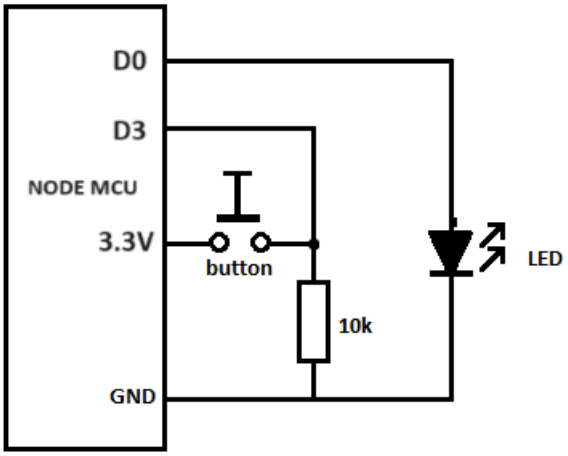
Result:

Experiment No: 1.b
To interface Push button with Arduino and write a program to 'turn ON' LED when push button is pressed or at sensor detection.

Component Required:

Sl No	Components	Quantity
1	NODE MCU ESP8266	1
2	LED or Buzzer	1
3	USB cable for NODE MCU	1
4	Connecting wires	--
5	Push Button	1
6	Bread Board	1
7	Resistance (Any value above 1KΩ)	1

Circuit Diagram



Program

```
#define buttonpin D3
#define led  D0

int buttonstate=0;

void setup() {
pinMode(buttonpin, INPUT);
pinMode(led, OUTPUT);
Serial.begin(9600);
}

void loop() {
buttonstate=digitalRead(buttonpin);
Serial.print("buttonstate status is:");
Serial.println(buttonstate);
if(buttonstate == HIGH)
 digitalWrite(led, HIGH);
else
```

```
 digitalWrite(led, LOW);
delay(500);
}
```

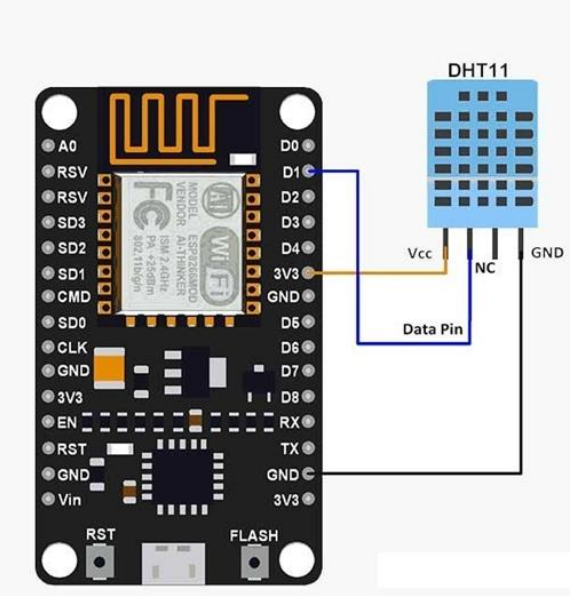
Result:

Experiment No.: 2.a
To interface DHT11 sensor with Arduino and write a program to print temperature and humidity readings

Component Required:

Sl No	Components	Quantity
1	NODE MCU ESP8266	1
2	DHT11 Temperature sensor	1
3	USB cable for NODE MCU	1
4	Connecting wires	--

Circuit Diagram



NODE MCU	DHT11
GND	GND
3V3	VCC
D2	DATA

Theory :

The **DHT11 sensor** is a **low-cost** digital temperature and humidity sensor. It operates at a **voltage of 3.3V to 5V** and can measure temperatures ranging from **0°C to 50°C** with an accuracy of ±2°C. Additionally, it can measure relative humidity ranging from **20% to 90%** with an accuracy of ±5% .

The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form. For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature.

DHT11 Specifications

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the Liquid Crystal library makes it easy to talk to character LCD displays.

There are thousands of libraries available for download directly through the Arduino IDE, and you can find all of them listed at the Arduino Library Reference.

Steps to Add DHT11 Sensor Library

1) Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**. The Library Manager should open.
2) Search DHT then find the DHT Sensor library by Adafruit
3) Click install button to install library
4) If ask click on install all button to install library dependencies

Program

```
#include <DHT.h>


#define DHTPIN D2
#define DHTTYPE DHT11
DHT dht(DHTPIN,  DHTTYPE);

float humidity;
float temp;

void setup() {
  Serial.begin(9600);
  dht.begin();

}

void loop() {
  humidity= dht.readHumidity();
  Serial.print("Humidity = ");
  Serial.print(humidity, 2);
```

```
  Serial.println (" %");

  temp=dht.readTemperature();
  Serial.print("Temperature = ");
  Serial.println(temp,2);
  delay(500);

}
```

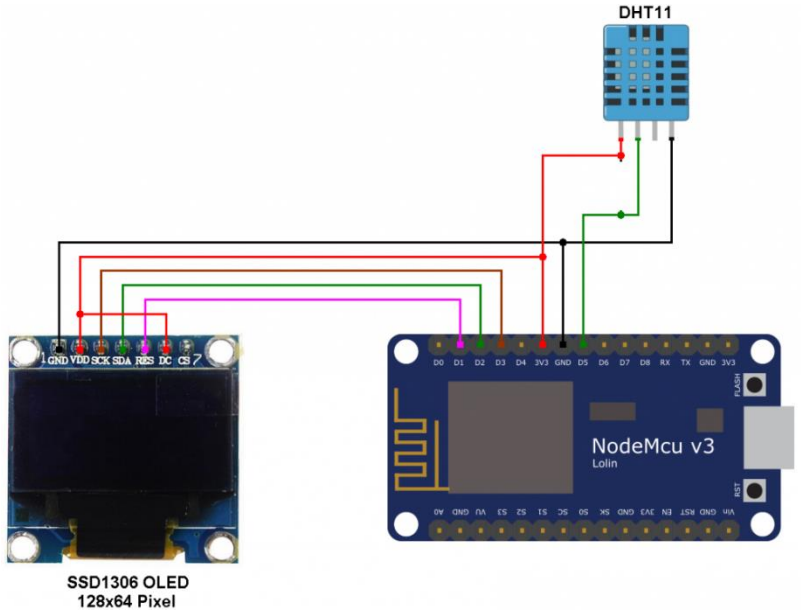
Result:

Experiment No.: 2.b
To interface OLED with Arduino and write a program to print temperature and humidity readings on it.

Component Required:

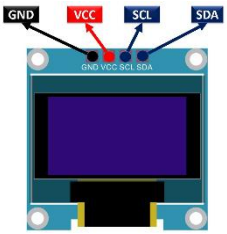
Sl No	Components	Quantity
1	NODE MCU ESP8266	1
2	DHT11 Temperature sensor	1
3	USB cable for NODE MCU	1
4	Connecting wires	--
5	OLED Display 128×64, SSD1306 I2C	1

Circuit Diagram



NODE MCU ESP 8266	DHT 11/22	OLED Pins
3.3V	VCC	VCC
GND	GND	GND
D5	DATA	--
D2	--	SDA
D3	--	SCK

Theory:



The SSD1306 OLED I2C 128X64 OLED Display module is a small monochrome organic light-emitting diode (OLED) display that is controlled through an I2C interface. It has a display resolution of 128×64 pixels, and the SSD1306 is the controller chip that manages the display. It's commonly used for display purposes in various electronics projects and is compact low power, and easily readable in low light conditions.

Specification of OLED	
Size	0.96 inch
Terminals	4
Pixels	128×64
Communication	I2C only
VCC	3.3V-5V
Operating Temperature	-40℃ to +80℃

To control the OLED display you need the adafruit_SSD1306.h and the adafruit_GFX.h libraries. Follow the next instructions to install those libraries.

1. Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**. The Library Manager should open.
2. Type "**SSD1306**" in the search box and install the SSD1306 library from Adafruit.
3. Click install button to install library
4. If ask click on install all button to install library dependencies
5. After installing the SSD1306 library from Adafruit, type "**GFX**" in the search box and install the library.
6. If ask click on install all button to install library dependencies
7. After installing the libraries, restart your Arduino IDE
8. Find the **Adafruit_SSD1306.h** file in the Arduino Library folder. Generally, it is located at **Documents\Arduino\libraries** on windows systems. There you will find the Adafruit_SSD1306.h file inside the Adafruit_SSD1306 folder.

Program

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>                    // Include DHT library code
#define OLED_RESET    5     // define SSD1306 OLED reset at ESP8266 GPIO5 (NodeMCU D1)
Adafruit_SSD1306 display(OLED_RESET);
#define DHTPIN  D5
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
char temperature[] = "00.0 C";
char humidity[]    = "00.0 %";
```

```
void setup(void)
{
  Serial.begin(9600);
  delay(1000);
  // set I2C pins [SDA = GPIO4 (D2), SCL = GPIO0 (D3)], default clock is 100kHz
  Wire.begin(4, 0);
  // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
   // initialize with the I2C addr 0x3D (for the 128x64)
  // init done
  display.begin(SSD1306_SWITCHCAPVCC, 0x3D);
  dht.begin();

  display.clearDisplay();
  display.drawFastHLine(0, 32, SSD1306_LCDWIDTH, WHITE);
  display.setTextSize(1);
  display.setTextColor(WHITE, BLACK);
  display.setCursor(10, 5);
  display.print("DHT11 TEMPERATURE:");
  display.setCursor(19, 37);
  display.print("DHT11 HUMIDITY:");
  display.display();
}

void loop()
{
  // Read humidity
  byte RH = dht.readHumidity();
  //Read temperature in degree Celsius
  byte Temp = dht.readTemperature();
  temperature[0] = Temp / 10 + '0';
  temperature[1] = Temp % 10 + '0';
  humidity[0]    = RH / 10   + '0';
  humidity[1]    = RH % 10   + '0';
  // print data on serial monitor

  Serial.printf("Temperature = %02u°C\r\n", Temp);
  Serial.printf("Humidity    = %02u %%\r\n\r\n", RH);
  // print data on the SSD1306 display

  display.setCursor(46, 20);
  display.print(temperature);
  display.setCursor(46, 52);
  display.print(humidity);
  display.drawRect(71, 20, 3, 3, WHITE);     // Put degree symbol ( ° )
  display.display();
  delay(1000);

}
```

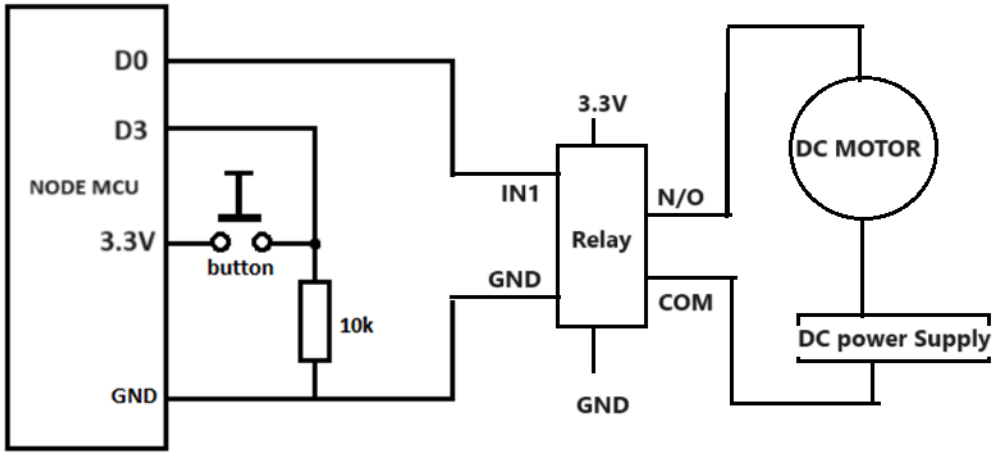
Result:

Experiment No: 3
To interface motor using relay with Arduino and write a program to 'turn ON' motor when push button is pressed.

Component Required:

Sl No	Components	Quantity
1	NOD MCU ESP 8266	1
2	Relay 2 Channel	1
3	USB cable for NODE MCU	1
4	Connecting wires	--
5	Push Button	1
6	Power Supply	1
7	DC motor	1

Circuit Diagram



Note : Set Regulated DC power supply based on Capacity of motor voltage

Theory

Relay: A relay module is an electrical switch that is operated by an electromagnet. **2-Channel 5V Relay Module** is a relay interface board, it can be controlled directly by a wide range of microcontrollers such as Arduino, AVR, PIC, ARM and so on. It uses a low-level triggered control signal (3.3-5VDC) to control the relay. Triggering the relay operates the normally open or normally closed contacts. It is an automatic switch to control a high-current circuit with a low-current signal.

Dual-Channel Relay Module Specifications

Supply voltage – 3.75V to 6V

Trigger current – 5mA

Current when relay is active - ~70mA (single), ~140mA (both)

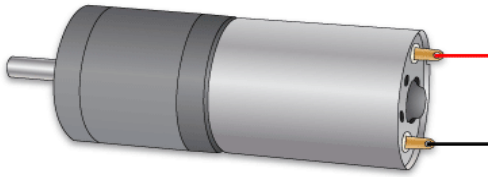
Relay maximum contact voltage – 250VAC, 30VDC

Relay maximum current – 10A

DC Motor: A DC motor is a machine which converts DC electrical energy into mechanical energy. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

DC Motor Specification

Operating Voltage(V): 12
Rated Speed (RPM): 200
Rated Torque(kg-cm): 1.5
Load Current (A): 0.3
No Load Current (A): 0.06



Program

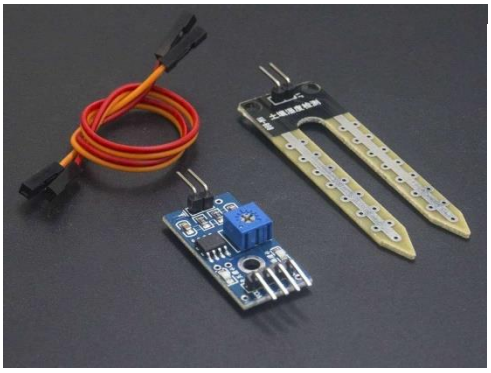
```
#define buttonpin D3
#define relay  D0

int buttonstate=0;
void setup() {
pinMode(buttonpin, INPUT);
pinMode(relay, OUTPUT);
Serial.begin(9600);
}

void loop() {
buttonstate=digitalRead(buttonpin);
Serial.print("buttonstate status is:");
Serial.println(buttonstate);
if(buttonstate == HIGH)
digitalWrite(relay, HIGH);
else
digitalWrite(relay, LOW);
delay(500);
}
```

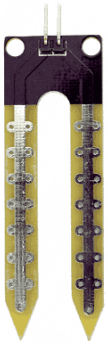
Result:

Experiment No: 4a
Write an Arduino/Raspberry Pi program to interface the Soil Moisture Sensor.

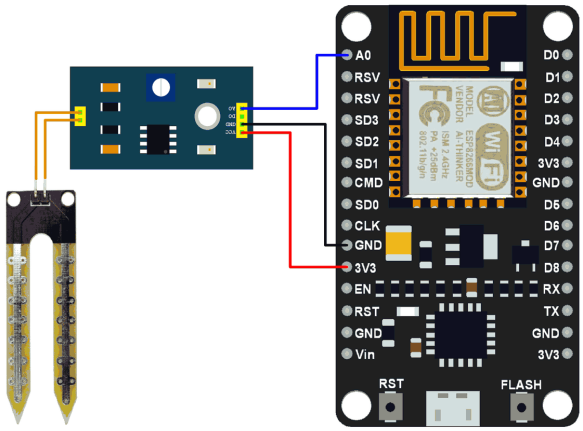
Component Required:

Sl No	Components	Quantity
1	Node MCU ESP8266	1
2	Soil Moisture Sensor	1
3	USB cable for NODE MCU	1
4	Connecting wires	--





Circuit Diagram



Pin Connection details		
Arduino Pins	**Bluetooth Module**	**LED**
3.3 V	VCC	
GND	GND	Cathode
Rx	Tx	
Tx	RX	
D4	--	Anode

Theory

Soil moisture is basically the content of water present in the soil. This can be measured using a soil moisture sensor which consists of two conducting probes that act as a probe. It can measure the moisture content in the soil based on the change in resistance between the two conducting plates.

The resistance between the two conducting plates varies in an inverse manner with the amount of moisture present in the soil.

Program

```
const int sensor_pin = A0;  /* Connect Soil moisture analog sensor pin to A0 of NodeMCU */

void setup() {
  Serial.begin(9600); /* Define baud rate for serial communication */
}

void loop() {
  float moisture_percentage;

  moisture_percentage = ( 100.00 - ( (analogRead(sensor_pin)/1023.00) * 100.00 ) );

  Serial.print("Soil Moisture(in Percentage) = ");
  Serial.print(moisture_percentage);
  Serial.println("%");

  delay(1000);
}
```

Result:

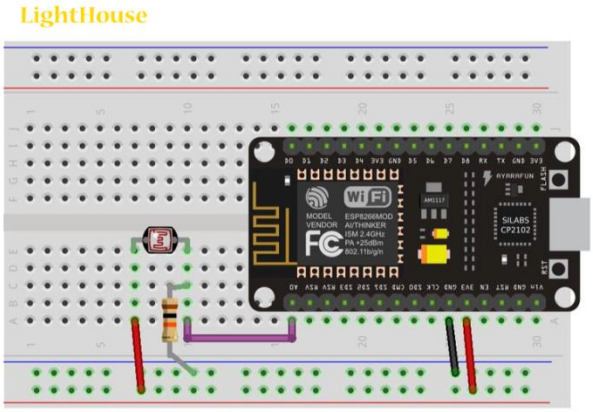
Experiment No: 4b

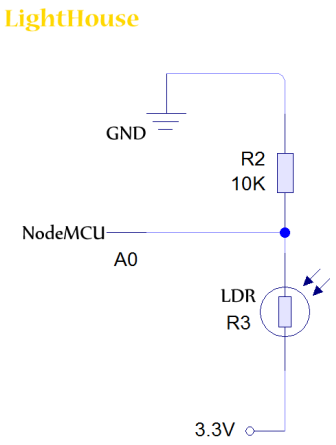
Write an Arduino/Raspberry Pi program to interface the LDR/Photo Sensor.

Component Required:

Sl No	Components	Quantity
1	NODE MCU ESP 8266	1
2	LDR	1
3	USB cable A to B	1
4	Connecting wires	1
5	Breadboard	1
6	10k ohm resistor	1

Circuit Diagram





Theory

A Light Dependent Resistor (LDR) or a photo resistor is a device whose resistivity is a function of the incident electromagnetic radiation. Hence, they are light sensitive devices. They are also called as photo conductors, photo conductive cells or simply photocells. They are made up of semiconductor materials having high resistance. The LDR output is actually analog in nature, so it gets connected to the A0 pin of the NodeMCU

Program

```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);

  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
```

```
  // print out the value you read:
  Serial.println(voltage);
}
```

Result:

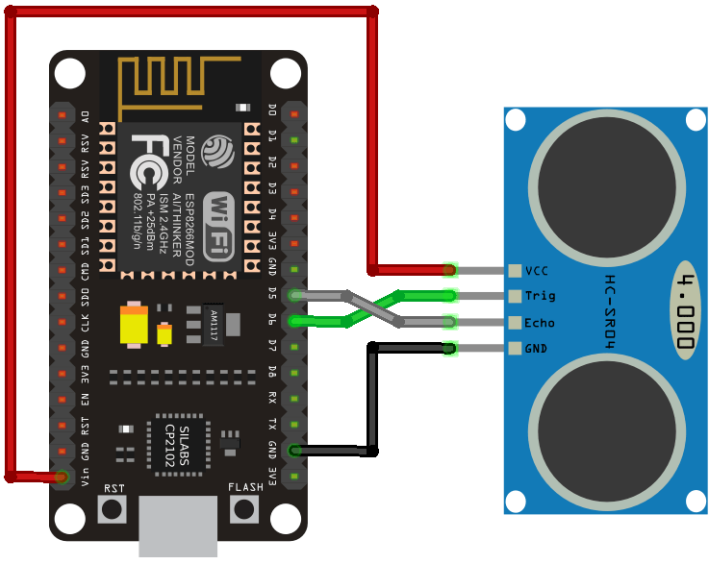
Experiment No: 5

Write a program to interface an Ultrasonic Sensor with Arduino /Raspberry Pi.

Component Required:

Sl No	Components	Quantity
1	NODE MCU ESP 8266	1
2	Ultrasonic Sensor	1
3	USB cable A to B	1
4	Connecting wires	1
5	Breadboard	1

Circuit Diagram



Ultrasonic Sensor	ESP8266
VCC	VIN
Trig	GPIO 12 (D6)
Echo	GPIO 14 (D5)
GND	GND

Theory

The ultrasonic sensor uses sonar to determine the distance to an object. This sensor reads from 2cm to 400cm (0.8inch to 157inch) with an accuracy of 0.3cm (0.1inches), which is good for most hobbyist projects. In addition, this particular module comes with ultrasonic transmitter and receiver modules.

Working of sensors:

1. The ultrasound transmitter (trig pin) emits a high-frequency sound (40 kHz).
2. The sound travels through the air. If it finds an object, it bounces back to the module.
3. The ultrasound receiver (echo pin) receives the reflected sound (echo).

Program

```
const int trigPin = 12;
const int echoPin = 14;

//define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}

void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculate the distance
  distanceCm = duration * SOUND_VELOCITY/2;
  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;
  // Prints the distance on the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);
   delay(1000);
}
```

Result:

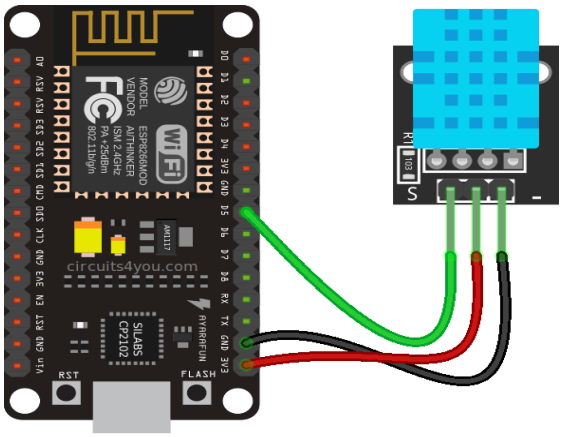
Experiment No: 6

Write a program on Arduino to upload temperature and humidity data to thingspeak cloud.

Component Required:

Sl No	Components	Quantity
1	ESP8266 12E Node MCU Kit	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--
5	Breadboard	1

Circuit diagram



NODE MCU	DHT11
GND	GND
3V3	VCC
D5	DATA

Theory

ThingSpeak is an open-source Internet of Things (IoT) application and API that allows users to collect and store sensor data in the cloud and perform analytics on that data. It allows users to create "channels" to collect data from multiple sensors, and also has built-in support for visualizing and analyzing the data. **ThingSpeak** can be used for a variety of applications, such as monitoring environmental conditions, tracking the location of assets, and controlling devices remotely. It is available for free and also has paid subscription plans for additional features and support. The device that sends the data must be configured with the correct channel information, such as the channel ID and write API key.

Steps to Connect

Step 1: ThingSpeak Setup for Temperature and Humidity Monitoring

a. For creating your channel on Thingspeak, you first need to Sign up on Thingspeak.
b. In case if you already have an account on Thingspeak, just sign in using your id and password.
c. For creating your account go to www.thinspeak.com.
d. After this, verify your E-mail id and click on continue.

Step 2: Create a Channel for Your Data

a. Once you Sign in after your account verification, create a new channel by clicking "New Channel" button.
b. After clicking on "New Channel", enter the Name and Description of the data you want to upload on this channel.
c. Enter the name of your data 'Temperature' in Field1 and 'Humidity' in Field2.
d. Click on the save channel button to save your details.

Step 3: API Key

a. To send data to Thingspeak, we need a unique API key, which we will use later in our code to upload our sensor data to Thingspeak Website.
b. Click on "API Keys" button to get your unique API key for uploading your sensor data.
c. Now copy your "Write API Key". We will use this API key in our code.

Program

```
#include <DHT.h>
#include <ESP8266WiFi.h>


const char* ssid = "choukimathwifi";
const char* password ="12345678";
String apiKey = "5OI7FCEK635ZJ336"; //Write API Key from website
const char* server = "api.thingspeak.com";


#define DHTPIN D1
#define DHTTYPE DHT11
DHT dht(DHTPIN,  DHTTYPE);
float humidity;
float temp;
```

```
WiFiClient client; //need to communication with the ThingSpeak server

void setup() {
  Serial.begin(9600);
  dht.begin();
  WiFi.begin(ssid, password);
  Serial.print("Wifi connecting to ...SSID:");
  Serial.println(ssid);
  Serial.print("connecting");

  while(WiFi.status()!=WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }

 Serial.println("Wifi connected successfully");
 Serial.print("nodemcu Ip Adress");
 Serial.println(WiFi.localIP());

}

void loop() {
  temp=dht.readTemperature();
  humidity= dht.readHumidity();


 if(client.connect(server, 80)){

    String postStr = apiKey;
    postStr += "&field1=";
    postStr += String(temp);
    postStr += "&field2=";
    postStr += String(humidity);
    postStr += "\r\n\r\n";

    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(postStr.length());
    client.print("\n\n");
    client.print(postStr);
    delay(1000); //necessary for posting to ThingSpeak

    Serial.print("Temperature = ");
    Serial.println(temp,2);
    Serial.print("Humidity = ");
    Serial.print(humidity, 2);
    Serial.println (" %");
    Serial.println("Data sent to ThingSpeak");
  }
```

```
  client.stop();
  Serial.println("Waiting for 15sec...");
  delay(15000); //ThingSpeak needs min. 15sec delay between each data post
}
```

Result :

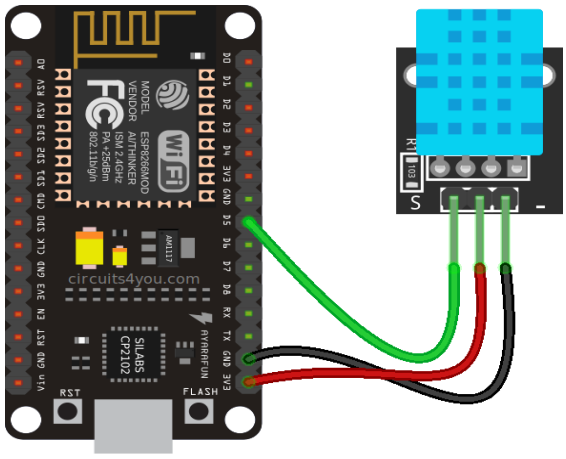
Experiment No: 7

Write a program on Arduino to retrieve temperature and humidity data from Thingspeak cloud.

Component Required:

Sl No	Components	Quantity
1	Node MCU	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--
5	Breadboard	1

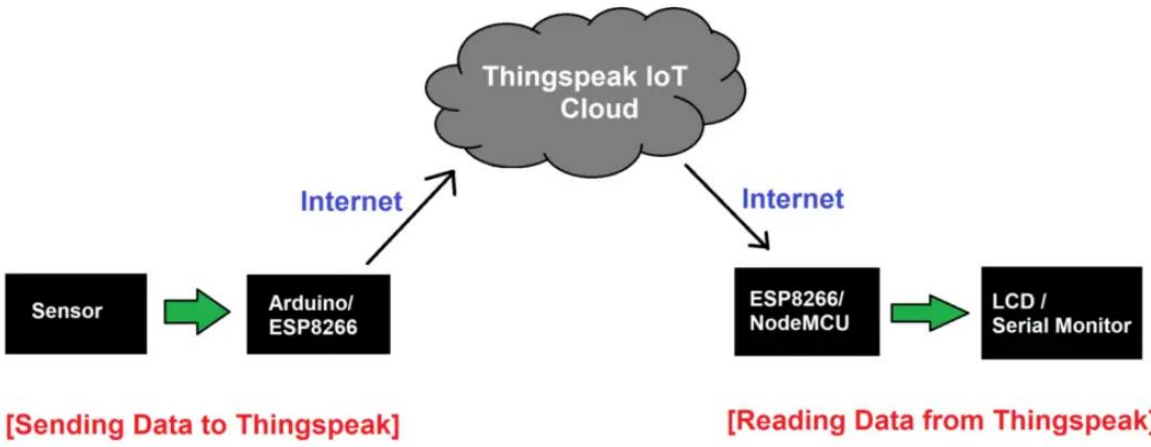
Circuit diagram



NODE MCU	DHT11
GND	GND
3V3	VCC
D5	DATA

Theory

To read values from Thingspeak we need to upload some data in real time, to do this, first upload temperature and humidity data to Thingspeak using previous experiment using NodeMCU 8266,



Installing the ThingSpeak Library

1. To send or receive sensor readings to ThingSpeak, we'll use the ThingSpeak Arduino library. Go to **Sketch** > **Include Library** > **Manage Libraries…** and search for

"**ThingSpeak**" in the Library Manager. Install the ThingSpeak library by MathWorks.

OR

2. Upload the Library from drive Go to **Sketch > Include Library > Add ZIP Library.… > open t**o install Library

Channel Settings that you need to do read data

Go to your Thingspeak account and do the following setting to receive temperature and humidity data.

3. Go to channel setting put 'tick' mark for both filed 1 and filed 2 and scroll down to bottom and save it.
4. You need your channel ID to read the fields on your channel you wish to read so that copy your channel id and paste in the code
5. You need your **Read API key** from your channel and copy **Read API key**.
6. use this **Read API key** in our code.
7. **Write following program and upload in the Node MCU82666**
8. After successful upload Open the serial monitor; you will be able to see the values read from your channel.

Program

```
#include <ThingSpeak.h>
#include <ESP8266WiFi.h>

const char* ssid = "choukimathwifi";
const char* password ="12345678";

//---------Channel Details---------//
// enter your Channel ID
unsigned long counterChannelNumber = 2307603;
// enter your Read API Key
const char * myCounterReadAPIKey = "GVA4D3FK3A4VG36C";
const int FieldNumber1 = 1;  // The field you wish to read
const int FieldNumber2 = 2;  // The field you wish to read


WiFiClient  client; //need to communication with the ThingSpeak server

void setup()
{
  Serial.begin(9600);
   WiFi.begin(ssid, password);

  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);


    while(WiFi.status()!=WL_CONNECTED){
    delay(500);
```

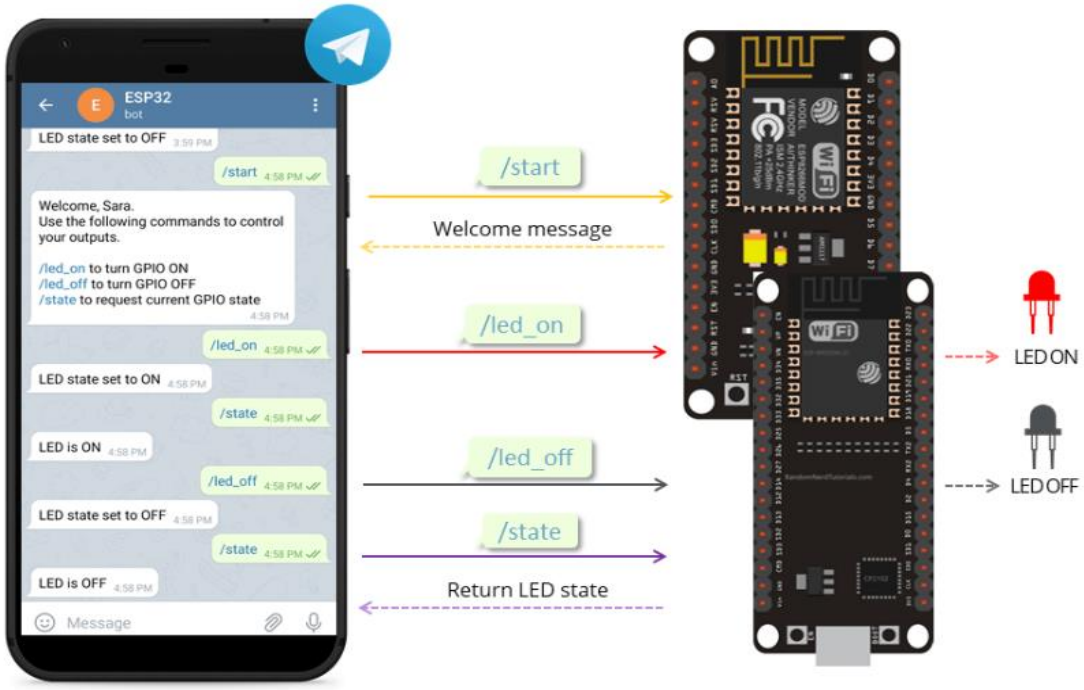
```
    Serial.println("Wifi connecting.....");
  }
  Serial.println("Wifi connected successfully ");
}

void loop(){
  //---------------- Channel 1 ----------------//
  long temp = ThingSpeak.readLongField(counterChannelNumber, FieldNumber1,
myCounterReadAPIKey);
  int statusCode = 0;
  statusCode = ThingSpeak.getLastReadStatus();
  if (statusCode == 200)
  {
    Serial.print("Temperature: ");
    Serial.println(temp);
  }
  else
  {
    Serial.println("Unable to read channel / No internet connection");
  }
  delay(100);

  //---------------- Channel 2 ----------------//
  long humidity = ThingSpeak.readLongField(counterChannelNumber, FieldNumber2,
myCounterReadAPIKey);
  statusCode = ThingSpeak.getLastReadStatus();
  if (statusCode == 200)
  {
    Serial.print("Humidity: ");
    Serial.println(humidity);
  }
  else
  {
    Serial.println("Unable to read channel / No internet connection");
  }
  delay(100);
  //-------------- End of Channel 2 ------------//
}
```

Result:

Experiment No: 8
Write a program to interface LED using Telegram App

Component Required:

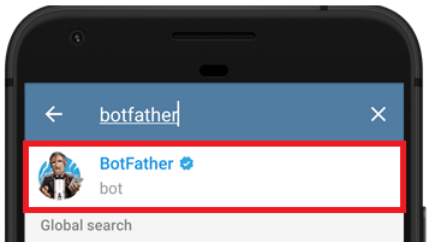
Sl No	Components	Quantity
1	Node MCU ESP 8266	1
2	LED	1
3	USB cable	1
4	Connecting wires	--
6	Breadboard	1



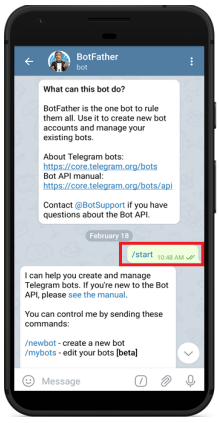
Theory
Telegram Messenger is a cloud-based instant messaging and voice over IP service. You can easily install it in your smartphone (Android and iPhone) or computer (PC, Mac and Linux). It is free and without any ads. Telegram allows you to create bots that you can interact with. The ESP32/ESP8266 will interact with the Telegram bot to receive and handle the messages, and send responses. In this experiment you'll learn how to use Telegram to send messages to your bot to control the ESP outputs from anywhere (you just need Telegram and access to the internet).

Creating a Telegram Bot

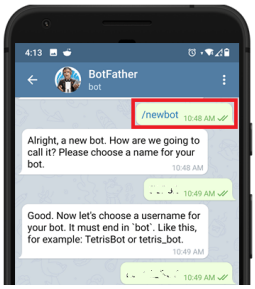
1. Go to Google Play or App Store, download and install Telegram.
2. Open Telegram App, search for "botfather" and click the BotFather as shown below. Or open this link t.me/botfather in your smartphone.



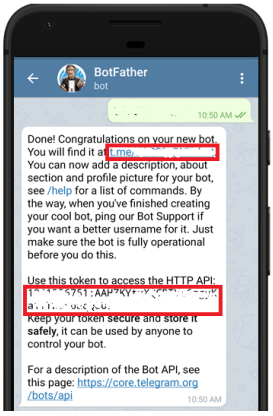
3. The following window should open and you'll be prompted to click the start button



4. Type /newbot and follow the instructions to create your bot. Give it a name and username.



5. If your bot is successfully created, you'll receive a message with a link to access the bot and the bot token. Save the bot token because you'll need it so that the ESP32/ESP8266 can interact with the bot.

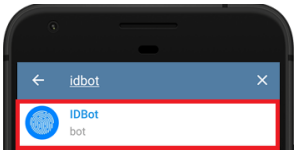


Get Your Telegram User ID:

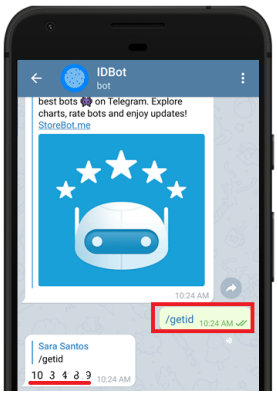
Anyone that knows your bot username can interact with it. To make sure that we ignore messages that are not from our Telegram account (or any authorized users), you can get your Telegram User ID.

Then, when your telegram bot receives a message, the ESP can check whether the sender ID corresponds to your User ID and handle the message or ignore it.

In your Telegram account, search for "IDBot" or open this link t.me/myidbot in your smartphone.



Start a conversation with that bot and type /getid. You will get a reply back with your user ID. Save that user ID, because you'll need it later in this tutorial.



Universal Telegram Bot Library

To interact with the Telegram bot, we'll use the Universal Telegram Bot Library created by Brian Lough that provides an easy interface for the Telegram Bot API.
Follow the next steps to install the latest release of the library.

1. Click here to download the Universal Arduino Telegram Bot library.
2. Go to **Sketch** > **Include Library** > **Add.ZIP Library..**.
3. Add the library you've just downloaded.

And that's it. The library is installed.

ArduinoJson Library

To install the ArduinoJson library. Follow the next steps to install the library.

1. Go to **Skech** > **Include Library** > **Manage Libraries**.
2. Search for "ArduinoJson".
3. Install the library.

Program

```
#ifdef ESP32
#include <WiFi.h>
#else
#include <ESP8266WiFi.h>
#endif
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>   // Universal Telegram Bot Library written by Brian Lough: https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot
#include <ArduinoJson.h>

// Replace with your network credentials
```

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

// Initialize Telegram BOT
#define BOTtoken
"XXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"  // your
Bot Token (Get from Botfather)

// Use @myidbot to find out the chat ID of an individual or a group
// Also note that you need to click "start" on a bot before it can
// message you
#define CHAT_ID "XXXXXXXXXX"

#ifdef ESP8266
  X509List cert(TELEGRAM_CERTIFICATE_ROOT);
#endif

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);

// Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

const int ledPin = 2;
bool ledState = LOW;

// Handle what happens when you receive new messages
void handleNewMessages(int numNewMessages) {
  Serial.println("handleNewMessages");
  Serial.println(String(numNewMessages));

  for (int i=0; i<numNewMessages; i++) {
    // Chat id of the requester
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != CHAT_ID){
      bot.sendMessage(chat_id, "Unauthorized user", "");
      continue;
    }

    // Print the received message
    String text = bot.messages[i].text;
    Serial.println(text);

    String from_name = bot.messages[i].from_name;

    if (text == "/start") {
```

```
      String welcome = "Welcome, " + from_name + ".\n";
      welcome += "Use the following commands to control your outputs.\n\n";
      welcome += "/led_on to turn GPIO ON \n";
      welcome += "/led_off to turn GPIO OFF \n";
      welcome += "/state to request current GPIO state \n";
      bot.sendMessage(chat_id, welcome, "");
    }

    if (text == "/led_on") {
      bot.sendMessage(chat_id, "LED state set to ON", "");
      ledState = HIGH;
      digitalWrite(ledPin, ledState);
    }

    if (text == "/led_off") {
      bot.sendMessage(chat_id, "LED state set to OFF", "");
      ledState = LOW;
      digitalWrite(ledPin, ledState);
    }

    if (text == "/state") {
      if (digitalRead(ledPin)){
        bot.sendMessage(chat_id, "LED is ON", "");
      }
      else{
        bot.sendMessage(chat_id, "LED is OFF", "");
      }
    }
  }
}

void setup() {
  Serial.begin(115200);

  #ifdef ESP8266
    configTime(0, 0, "pool.ntp.org");      // get UTC time via NTP
    client.setTrustAnchors(&cert); // Add root certificate for api.telegram.org
  #endif

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState);

  // Connect to Wi-Fi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  #ifdef ESP32
    client.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Add root certificate for
```

```
api.telegram.org
  #endif
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }
  // Print ESP32 Local IP Address
  Serial.println(WiFi.localIP());
}

void loop() {
  if (millis() > lastTimeBotRan + botRequestDelay)  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

    while(numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }
}
```

Result:

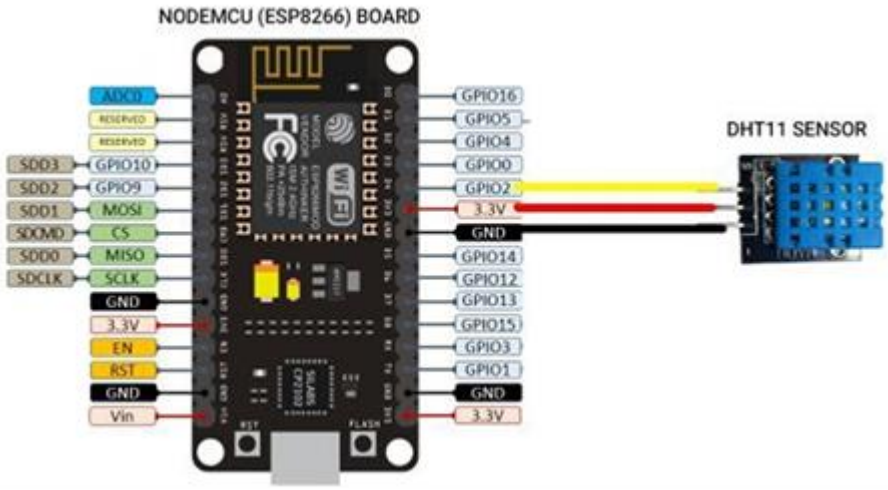
Experiment No: 9

Write a program on Arduino to publish temperature data to MQTT broker.

Component Required:

Sl No	Components	Quantity
1	Node MCU ESP 8266	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--
6	Breadboard	1

Circuit Diagram

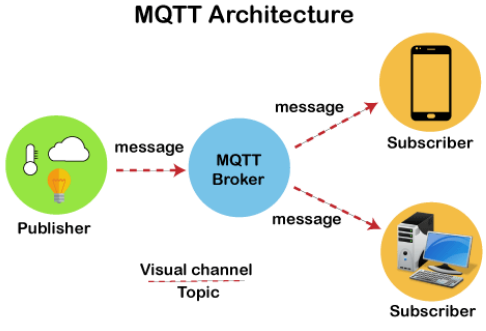


PIN Configuration : NODMCU 3.3V to DHT VCC Pin
NODMCU GND to DHT GND pin
NODMCU GPIO2 to DHT DATA pin

Theory

MQTT stands for **M**essage **Q**ueuing **T**elemetry **T**ransport. MQTT is a simple messaging protocol, designed for constrained devices with low bandwidth. So, it's the perfect solution to exchange data between multiple IoT devices.

Devices ***publish*** messages on a specific topic. All devices that are ***subscribed*** to that topic receive the message.



In a publish and subscribe system, a device can publish a message on a topic, or it can be subscribed to a particular topic to receive messages

The **MQTT broker** is responsible for **receiving** all messages, **filtering** the messages, **deciding** who is interested in them, and then **publishing** the message to all subscribed clients.

The MQTT broker is the central point of communication, and it is in charge of dispatching all

messages between the senders and the rightful receivers. A client is any device that connects to the broker and can publish or subscribe to topics to access the information. A topic contains the routing information for the broker. Each client that wants to send messages publishes them to a certain topic, and each client that wants to receive messages subscribes to a certain topic. The broker delivers all messages with the matching topic to the appropriate clients.

ThingSpeak™ has an MQTT broker at the URL mqtt3.thingspeak.com and port 1883. The ThingSpeak broker supports both MQTT publish and MQTT subscribe

In this Experiment, we will create a setup that allows a NODE MCUESP8266 board to send data to another MCU ESP 8266, using MQTT (Message Queuing Telemetry Transport). The sender device, simply publishes a message to a broker service, which then can be subscribed to by a receiver device.

The data we will send consists of readings from a DHT11 sensor, including temperature and humidity data, from a NODE MCU ESP8266 to another NODE MCU. This experiment utilizes the broker test.mosquitto.org, an open-source service that is free for anyone to use.

Getting Started to MQTT

Go to Tools **>** **Manage libraries..**, and search for **ArduinoMqttClient** and install.

Program

```
#include <ESP8266WiFi.h>
#include <ArduinoMqttClient.h>
#include <DHT.h>
#include <DHT_U.h>

const char* ssid = "choukimathwifi";

const char* password ="12345678";

const long interval = 8000;
unsigned long previousMillis = 0;
int count = 0;

const char topicT[]  = "Temperature  Value is ";
const char topicH[]  = "Humidity Value is";

WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

const char broker[] = "test.mosquitto.org";
int        port     = 1883;
```

```
#define DHTPIN D4
#define DHTTYPE DHT11
DHT dht(DHTPIN,  DHTTYPE);
float humidity;
float temp;

void setup() {

  Serial.begin(9600);
  dht.begin();
  Serial.println();
  Serial.print("Wifi connecting to ....");
  Serial.print(ssid);
  WiFi.begin(ssid, password);


  while(WiFi.status()!=WL_CONNECTED){
    Serial.println("connecting.....");
    delay(500);
  }
  Serial.println("Wifi connected successfully");

  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);

  if (!mqttClient.connect(broker, port)) {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());

    while (1);
  }
  Serial.println("You're connected to the MQTT broker!");

}

void loop() {
   mqttClient.poll();


   unsigned long previousMillis = 0;
   unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // save the last time a message was sent
    previousMillis = currentMillis;

   humidity= dht.readHumidity();
   Serial.print("Humidity = ");
   Serial.print(humidity, 2);
   Serial.println (" %");
```

```
  temp=dht.readTemperature();
  Serial.print("Temperature = ");
  Serial.println(temp,2);
  delay(500);

   mqttClient.beginMessage(topicT);
   mqttClient.print(temp);
   mqttClient.endMessage();

   mqttClient.beginMessage(topicH);
   mqttClient.print(humidity);
   mqttClient.endMessage();

   Serial.println();
 }
 delay(1000);
}
```

Result:

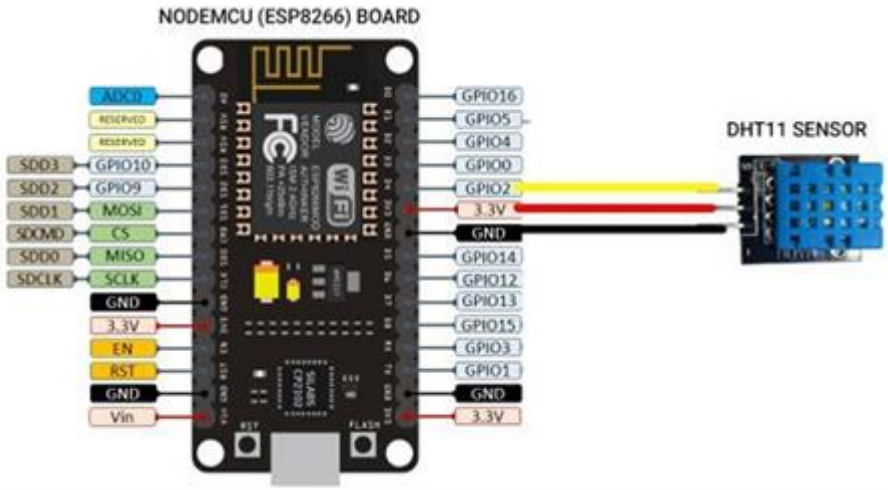
Experiment No: 10

Write a program to create UDP server on Arduino and respond with humidity data to UDP client when requested.

Component Required:

Sl No	Components	Quantity
1	Node MCU ESP 8266	1
2	DHT11	1
3	USB cable	1
4	Connecting wires	--
6	Breadboard	1

Circuit Diagram



PIN Configuration : NODMCU 3.3V to DHT VCC Pin
NODMCU GND to DHT GND pin
NODMCU D4 (GPIO2) to DHT DATA pin

Theory

User Datagram Protocol (UDP) is a network communication protocol that operates at the transport layer of the Internet Protocol (IP) suite. It is a connectionless and lightweight protocol designed for fast and efficient data transmission, but it does not provide the same level of reliability and error-checking as Transmission Control Protocol (TCP).

UDP is a lightweight, connectionless, and fast protocol that prioritizes low-latency data transmission over reliability. It is suitable for applications where occasional packet loss or out-of-order delivery can be tolerated, and real-time communication is essential. However, for applications that require guaranteed delivery and error recovery, TCP is a better choice.

Program

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <DHT.h>

#define DHTPIN D4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

float temp;
char packet[1024];
const char* ssid = "choukimathwifi";
const char* password ="12345678";
const char* udpServerIP = "192.168.0.245";  // Replace <IP> with the actual IP address of your computer
unsigned int port = 9000;

WiFiUDP Udp; // Create a UDP object

void setup() {
  Serial.begin(9600);
  dht.begin();
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi...");
  }
 Serial.println("Wifi connected successfully");
}

void loop() {
  Udp.beginPacket(udpServerIP, port);

  // Read temperature from DHT sensor and convert it to a string
  temp = dht.readTemperature();
  //confirm value in serila monitor
  Serial.print("Temperature in degree Cel. :");
  Serial.println(temp);


  String tempStr = String(temp);

  // Copy the temperature string into the packet buffer
  tempStr.toCharArray(packet, 1024);

  // Send the packet over UDP
  Udp.write(packet);

  // End the UDP packet and send it
  Udp.endPacket();
  delay(500);
```

```
}
```

Python code for UDP Server

```
import socket

UDP_IP = "192.168.0.245"
UDP_PORT = 9000

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((UDP_IP, UDP_PORT))

while True:
   received_data, addr = sock.recvfrom(1024)
   print(received_data)
```

Procedure

To run Python code for UDP Server

1) Install python software
2) Open python IDLE
3) In python IDLE go to **File** → **New File** (it opens new script windows) → **Type the code**
4) Click on **Run** button and **Save** the program from script window
5) See the output from IDLE shell (command prompt)

Result:

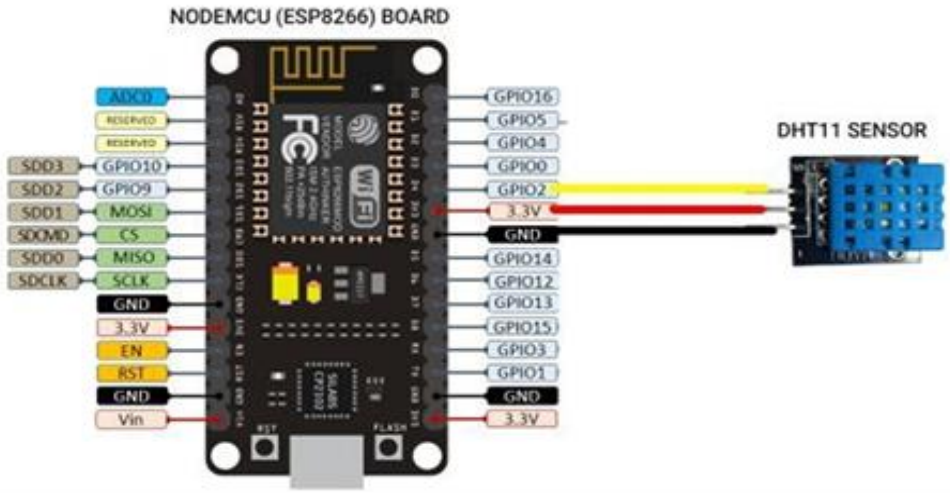
Experiment No: 11

Write a program to create a TCP server on Arduino /Raspberry Pi and respond with humidity data to the TCP client when requested.

Component Required:

Sl No	Components	Quantity
1	Node MCU ESP 8266	1
2	DHT11	1
3	USB cable	1
4	Connecting wires	--
6	Breadboard	1

Circuit Diagram



PIN Configuration : NODMCU 3.3V to DHT VCC Pin
NODMCU GND to DHT GND pin
NODMCU D4 (GPIO2) to DHT DATA pin

Theory

The Transmission Control Protocol (TCP) is a widely used protocol in the Internet Protocol (IP) suite. It is a connection-oriented protocol that provides reliable, ordered, and error-checked delivery of data between applications running on hosts1. In IoT, TCP is used to establish connections between clients and servers, allowing devices to interact with each other and resolve common problems.

A TCP server is a program that listens for incoming connections from clients and responds to their requests. When a client connects to the server, it sends a request for data. The server then sends back the requested data to the client. In IoT, TCP servers are used to provide access to data from sensors and other devices
In the context of the Internet of Things (IoT), a TCP server refers to a network service running on a device or gateway that listens for incoming TCP connections from other IoT devices or clients

In the context of IoT, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are two protocols of the Transport Layer that are used to transmit data between devices over a network.

TCP is a connection-oriented protocol that provides reliable delivery services by keeping track of the segments being transmitted or received by assigning numbers to every single one of them. It also implements an error control mechanism for reliable data transfer and takes into account the level of congestion in the network .

On the other hand, UDP is a connectionless protocol that is used for simple request-response communication when the size of data is less and hence there is lesser concern about flow and error control. It is a suitable protocol for multicasting as UDP supports packet switching. Normally used for real-time applications which cannot tolerate uneven delays between sections of a received message.

In summary, TCP is more reliable but slower than UDP, while UDP is faster but less reliable than TCP

Finding IP address of your computer

a. Search command Prompt from your PC
b. Type ***ipconfig*** in command prompt
c. Search ***IPv4 Adress …….: 192.168.137.1*** like this in prompt
d. Copy and paste the same in the program at ***tcpServerIP***

Program

```
#include <ESP8266WiFi.h>
#include <DHT.h>

#define DHTPIN D4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

float temp;
const char* ssid = "choukimathwifi";
const char* password = "12345678";
const char* tcpServerIP = "192.168.1.6";  // Replace <IP> with the actual IP address of your computer
unsigned int port = 80; //most commonly used protocol port in TCP

WiFiClient client;  // Create a TCP client object

void setup() {
  Serial.begin(9600);
  dht.begin();
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi connected successfully");
}

void loop() {
  client.connect(tcpServerIP, port);
    // Read temperature from DHT sensor and convert it to a string
  temp = dht.readTemperature();
    // Confirm value in serial monitor
  Serial.print("Temperature in degree Celsius: ");
  Serial.println(temp);
```

```
  // Send the temperature over TCP as a string
 client.print(temp);
 delay(500);

}
```

Python code for UDP Server

```
import socket
mysock =socket.socket(socket.AF_INET,  socket.SOCK_STREAM)
TCP_IP = '192.168.1.6' # Get local machine IP from LAN or WIFI setting of PC
TCP_PORT=80; #most commonly used protocol in TCP

mysock.bind((TCP_IP, TCP_PORT))

mysock.listen(5)  # Now wait for client connection.
while True:
   received_data, addr=mysock.accept() # Establish connection with client.

   data = received_data.recv(1024).decode()
   #we are extracting 1024 bytes of data at a time.
   #(One can use one's convenient number here).
   print(data)
```

Procedure

To run Python code for UDP Server

1) Install python software (Ignore if already installed)
2) Open python IDLE
3) In python IDLE go to **File** → **New File** (it opens new script windows) → **Type the code**
4) Click on **Run** button and **Save** the program from script window
5) See the output from IDLE shell (command prompt)

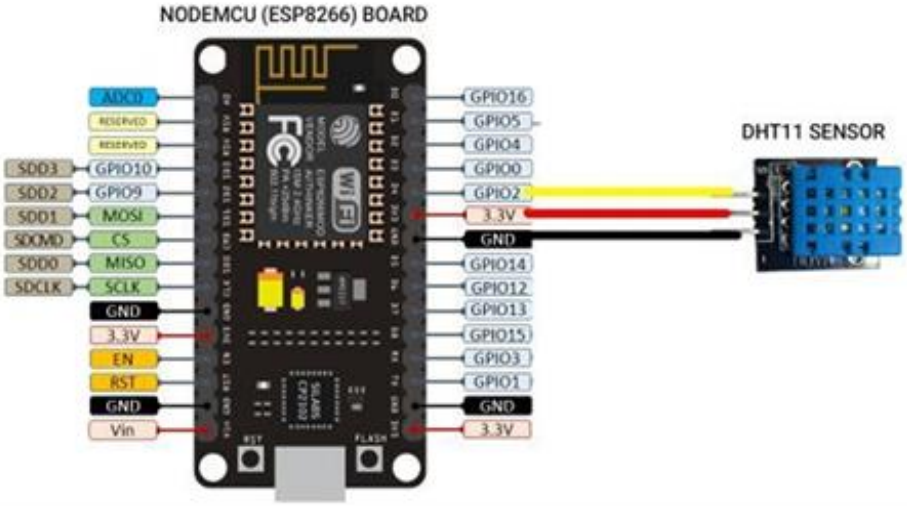
Result:

Experiment No: 12
Write a program on Arduino to subscribe to MQTT broker for temperature data and print it.

Component Required:

Sl No	Components	Quantity
1	Node MCU ESP 8266	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--
6	Breadboard	1

Circuit Diagram

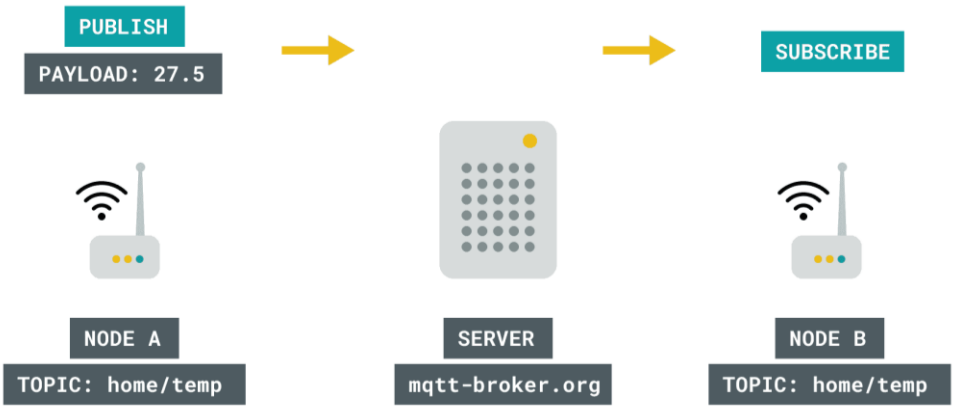


PIN Configuration : NODMCU 3.3V to DHT VCC Pin
NODMCU GND to DHT GND pin
NODMCU D5 to DHT DATA pin

Theory

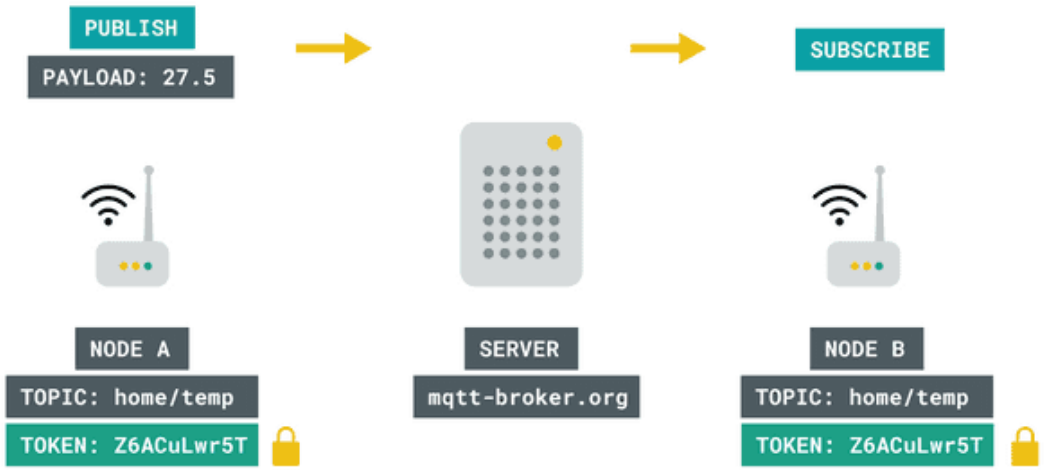
The MQTT protocol was first introduced in 1999, as a light-weight **publish** and **subscribe** system. It is particularly useful for devices with low-bandwidth, where we can send commands, sensor values or messages over the Internet with little effort.

A basic explanation on how it works is that a node, for example and Arduino with a Wi-Fi module, sends a payload to a broker. A broker is a kind of "middle-point" server, that essentially stores payloads sent to it, in something called **topics**. A topic, is a definition of what type of data it contains, it could for example be "humidity" or " temperature". Another node



can then subscribe to this information, from the broker, data has been moved from Node A to Node B over the Internet.

One way to protect the data is for example, by using a **token**, something that is quite common



when working with various IoT services. For instance, if we are publishing something to a broker, anyone that has the URL, e.g. **randombroker.org/randomtopic** can subscribe to it. But if we add a unique token on both sides, they wouldn't be able to. These tokens could for example be **Z6ACuLwr5T**, which is not exactly something easy to guess.

Program

```
#include <ESP8266WiFi.h>
#include <ArduinoMqttClient.h>

const char* ssid = "choukimathwifi";

const char* password ="12345678";


const char broker[] = "test.mosquitto.org";
int        port     = 1883;
const char topicT[]  = "Temperature  Value is ";
const char topicH[]  = "Humidity Value is";
```

```
WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);



void setup() {

  Serial.begin(9600);

  Serial.print("Wifi connecting to ....");
  Serial.print(ssid);
  WiFi.begin(ssid, password);


  while(WiFi.status()!=WL_CONNECTED){
    Serial.println("connecting.....");
    delay(500);
  }
  Serial.println("Wifi connected successfully");

  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);

  if (!mqttClient.connect(broker, port)) {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());

    while (1);
  }
  Serial.println("You're connected to the MQTT broker!");
  Serial.println();

  // set the message receive callback
  mqttClient.onMessage(onMqttMessage);

  Serial.print("Subscribing to topic: ");
  Serial.println(topicH);
  Serial.println(topicT);
  Serial.println();

  // subscribe to a topic
  mqttClient.subscribe(topicT);
  mqttClient.subscribe(topicH);


  // topics can be unsubscribed using:
  // mqttClient.unsubscribe(topic);

  Serial.print("Topic Tempertaure: ");
  Serial.println(topicT);
  Serial.print("Topic Humidity: ");
```

```
  Serial.println(topicH);
  Serial.println();

}

void loop() {
   mqttClient.poll();
  delay(1000);
}
void onMqttMessage(int messageSize) {
  // we received a message, print out the topic and contents
  Serial.println("Received a message with topic '");
  Serial.print(mqttClient.messageTopic());
  Serial.print("', length ");
  Serial.print(messageSize);
  Serial.println(" bytes:");

  // use the Stream interface to print the contents
  while (mqttClient.available()) {
    Serial.print((char)mqttClient.read());
  }
  Serial.println();
  Serial.println();
  delay(1500);
}
```

Result:

Experiment No: 13
Virtual Lab: Internet of Things (IoT) Virtual Lab

Virtual Lab link: https://iot-amrt.vlabs.ac.in/

Introduction:

By simulating real-world scenarios and enabling cloud connectivity, the lab fosters creativity, problem-solving, and technical fluency. Whether students are designing sensor-based systems or developing cloud-integrated applications, the virtual lab offers a dynamic platform to learn, test, and innovate They are of the following types:

Objectives:

Conceptual Mastery: Understand the architecture and components of IoT systems, including sensors, microcontrollers, and communication protocols.

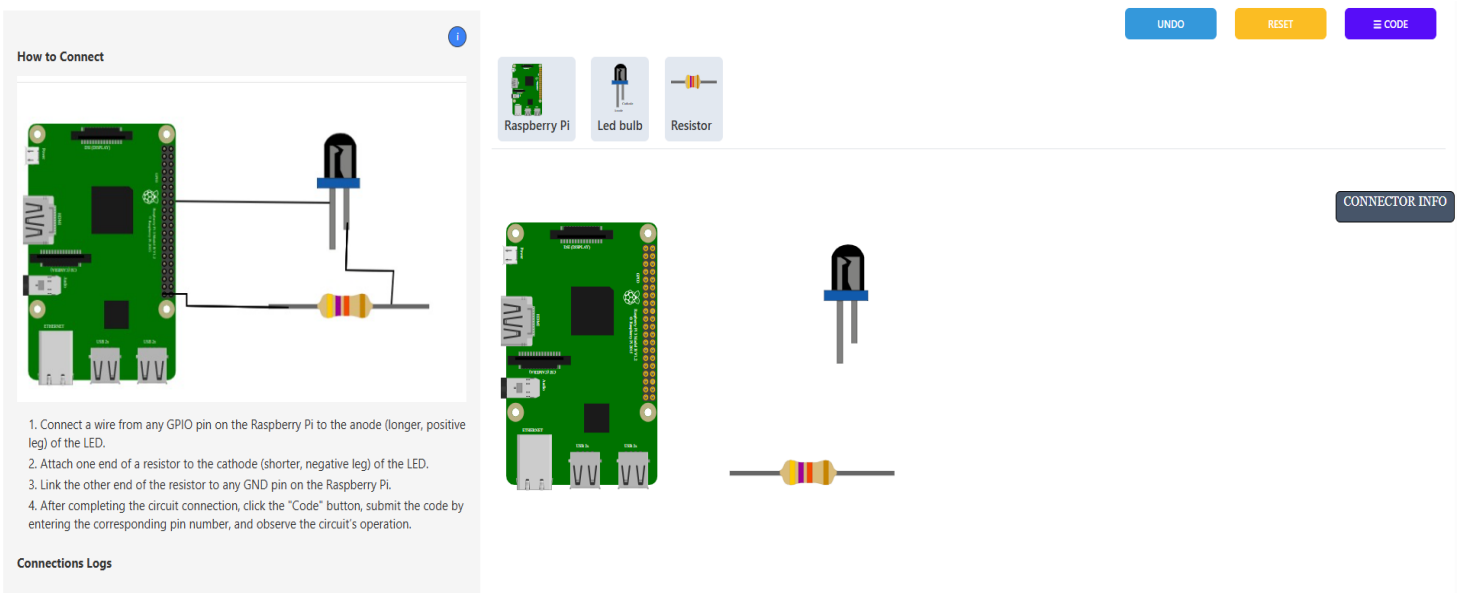
Hands-on Skills: Develop embedded programs and build IoT prototypes using platforms like Arduino and Raspberry Pi.

Sensor Integration: Identify and utilize sensor technologies to capture real-world data across industrial and social domains.

Application Development: Apply IoT concepts to create meaningful, scalable solutions for real-world challenges using virtualized hardware environments.

List of Experiments:

1. Introduction to Raspberry PI
2. Raspberry Pi LED Blink Program
3. Raspberry Pi Push Button LED Blink Program
4. DHT11 sensor with Raspberry Pi Program
5. Raspberry Pi Ultrasonic Sensor Interface Program
6. Raspberry Pi Motor Control with Relay
7. Raspberry Pi Bluetooth Data Transfer to Smartphone.
8. Raspberry Pi Pico-LED Blink Program Via Bluetooth and Smartphone



Result: