Digital System Design using Arduino

K Prasanna Kumar

Contents

***Abstract*—This module explains how do design a digital logic circuit using Arduino.**

1 Basic Gates

1.1 Fundamental gates

A	B	$\overline{A}$	AB	A + B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

```
int A, B, C, D, E;

void setup() {
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
}

void loop() {

 A = digitalRead(3);
 B = digitalRead(4);

    C= !A;
    D = A&&B;
    E = A||B;

  digitalWrite(5, C);
  digitalWrite(6, D);
  digitalWrite(7, E);
}
```

1.2 Universal Gates

A	B	$\overline{AB}$	$\overline{A+B}$
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

```
int A, B, C, D, E;

void setup() {
  pinMode(3, INPUT);
```

The author is a project associate with the National Resource Centre, IIT, Hyderabad 502285 India e-mail: kk.prassu924@gmail.com.

```
  pinMode(4, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop() {
 A = digitalRead(3);
 B = digitalRead(4);
   C = !(A&&B);
   D = !(A||B);
 digitalWrite(5, C);
 digitalWrite(6, D);
}
```

1.3 *Exclusive OR and NOR gates*

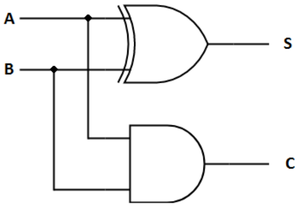
A	B	$A\overline{B}+\overline{A}B$	$\overline{A\overline{B}+\overline{A}B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

```
int A, B, C, D;

void setup() {
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop() {
 A = digitalRead(3);
 B = digitalRead(4);
    C = (A&&!B)||(!A&&B);
// XOR i.e. Exclusive OR operation on A,B
    D = !(C);
// XNOR i.e. Exclusive NOR operation on A,B
 digitalWrite(5, C);
 digitalWrite(6, D);
}
```

2 Combination Logic Circuits

2.1 *Adder*

Problem 1. Half adder



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

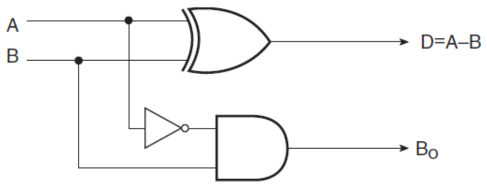
```
int A, B, S, C;

void setup() {
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop() {
 A = digitalRead(3);
 B = digitalRead(4);
    S = (A&&!B)||(!A&&B);
    C = A&&B;
 digitalWrite(5, S);
 digitalWrite(6, C);
}
```

Problem 2. Design full adder with basic gates

Problem 3. Design full adder with half adder

2.2 *Subtractor*

Problem 4. Half subtractor



A	B	NOT A	Difference	Borrow
0	0	1	0	0
0	1	1	0	1
1	0	0	0	0
1	1	0	1	0

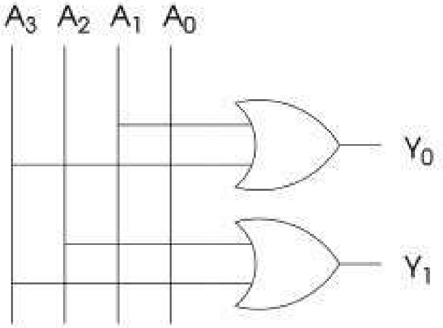
```
int A, B, D, Bo;

void setup() {
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop() {
 A = digitalRead(3);
 B = digitalRead(4);
    D = (A&&!B)||(!A&&B);
    Bo = !A&&B;
 digitalWrite(5, D);
 digitalWrite(6, Bo);
}
```

Problem 5. Design full subtractor with basic gates

Problem 6. Design full subtractor with half adder

2.3 Encoder

Problem 7. 4:2 Encoder



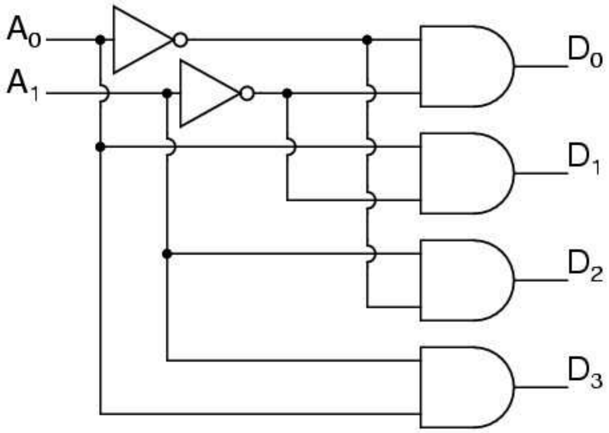
A0	A1	A2	A3	Y1	Y0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

```
int A,B,C,D,X,Y;
// A = A0, B= A1, C =A2, D = A3, X
   =Y0; Y = Y1;

void setup() {
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop() {
  A = digitalRead(3);
  B = digitalRead(4);
  C = digitalRead(5);
  D = digitalRead(6);
X = B||D;
Y = C||D;
 digitalWrite(5, X);
 digitalWrite(6, Y);
}
```

Problem 8. Design an 8 bit and 16 bit Encoder

2.4 Decoder

Problem 9. 2:4 Decoder



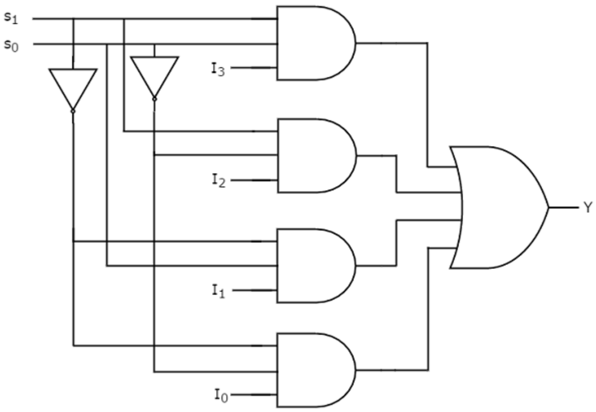
A1	A0	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

```
int A,B,C,D,a,b;
// A = D0, B= D1, C =D2, D = D3, a
   =A0; b = A1;

void setup() {
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop() {
  a = digitalRead(3);
  b = digitalRead(4);

  A = !a&&!b;
  B = a&&!b;
  C = !a&&b;
  D = a&&b;

 digitalWrite(5, A);
 digitalWrite(6, B);
 digitalWrite(7, C);
 digitalWrite(8, D);
}
```

Problem 10. Design an 8 bit and 16 bit Decoder

2.5 Multiplexer

Problem 11. 4:1 Multiplexer



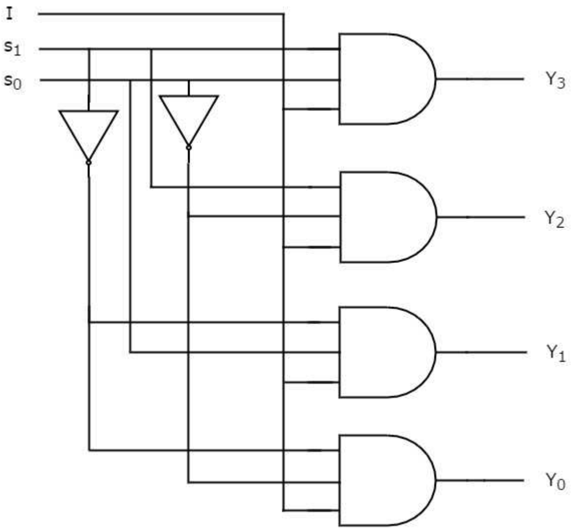
S0	S1	Output Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

```
int A,B,C,D,a,b;
// A = D0, B= D1, C =D2, D = D3, a
   =A0; b = A1;
int I0,I1, I2, I3, Y;
void setup() {
  pinMode(3, INPUT);
```

```
  pinMode(4, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop() {
  a = digitalRead(3);
  b = digitalRead(4);

  A = !a&&!b&&I0;
  B = a&&!b&&I1;
  C = !a&&b&&I2;
  D = a&&b&&I3;

  Y = A||B||C||D;

 digitalWrite(5, Y);
}
```

2.6 Demultiplexer

Problem 12. 1:4 Demultiplexer



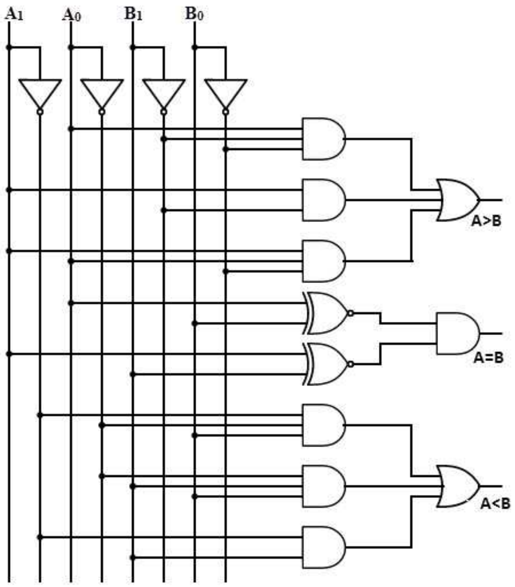
S0	S1	D0	D1	D2	D3
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

```
int S0, S1;
int Y0,Y1, Y2, Y3, I;
void setup() {
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop() {
  S0 = digitalRead(3);
  S1 = digitalRead(4);

 Y0 = !S0&&!S1&&I;
 Y1 =  S0&&!S1&&I;
 Y2 = !S0&&!S1&&I;
 Y3 =  S0&&S1&&I;

 digitalWrite(5, Y0);
 digitalWrite(6, Y1);
 digitalWrite(7, Y2);
 digitalWrite(8, Y3);
}
```

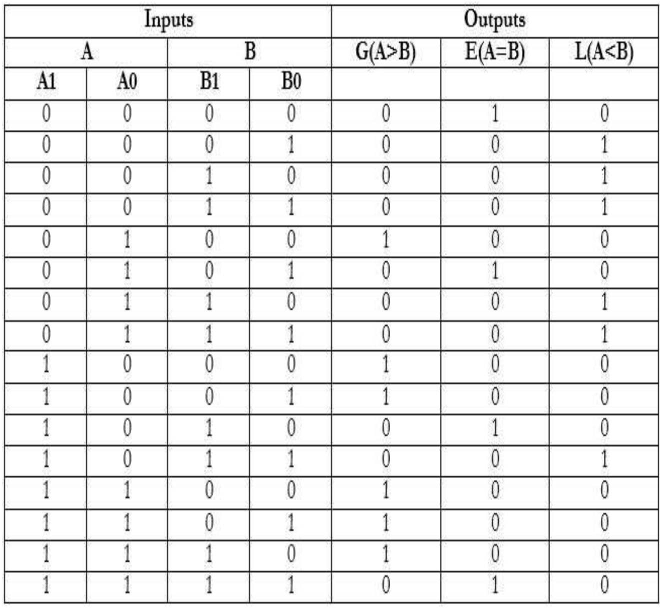
Problem 13. Design Mux and De-Mux with 3,4 and 5 input select lines

2.7 Comparator

Problem 14. Design 2-bit comparator using Arduino



using the following truth table to verify the results

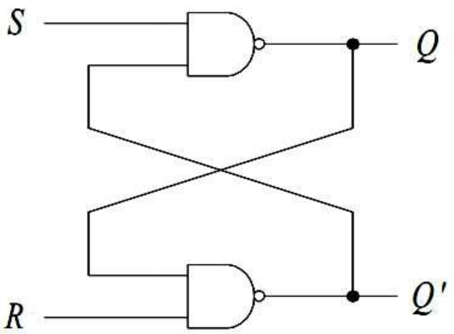


Inputs				Outputs		
A		B		G(A>B)	E(A=B)	L(A<B)
A1	A0	B1	B0			
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

3 Sequential Logic Circuits

3.1 SR Latch

Problem 15. SR Latch using NAND



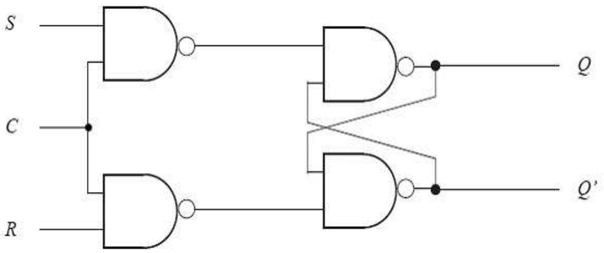
S	R	Q_n	$\overline{Q_n}$	Condition
0	0	-	-	Not used
0	1	1	0	-
1	0	0	1	-
1	1	-	-	Memory

```
int S, R, Q=0,NQ;
void setup()
{
  pinMode(3,INPUT);
  pinMode(4,INPUT);
  pinMode(8,OUTPUT);
  pinMode(7,OUTPUT);
}
void loop()
{
  S = digitalRead(3);
  R = digitalRead(4);

  NQ=!(R&&Q);
  Q=!(S&&NQ);

   digitalWrite(7,Q);
   digitalWrite(8,NQ);
}
```

Problem 16. SR Latch using NOR

3.2 SR Flip Flop

Problem 17. SR Flip Flop using NAND SR Latch



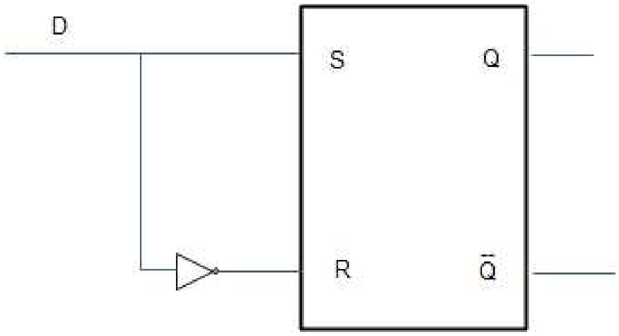
CLK	S	R	Q_n	$\overline{Q_n}$	Condition
1	0	0	-	-	Memory
1	0	1	0	1	-
1	1	0	1	0	-
1	1	1	-	-	Not Used
0	x	x	-	-	Memory

```
int S, R, Q=0,NQ;
int s, r;
int CK = 1;
void setup()
{
  pinMode(3,INPUT);
  pinMode(4,INPUT);
  pinMode(8,OUTPUT);
  pinMode(7,OUTPUT);
}
void loop()
{
  S = digitalRead(3);
  R = digitalRead(4);

  s = !(CK&&S);
  r = !(CK&&R);
  Q=!(s&&NQ);
  NQ=!(r&&Q);

   digitalWrite(7,Q);
   digitalWrite(8,NQ);
}
```

Problem 18. SR Flip Flop using NOR SR Latch

3.3 D Flip Flop

Problem 19. D Flip Flop using NAND SR Latch



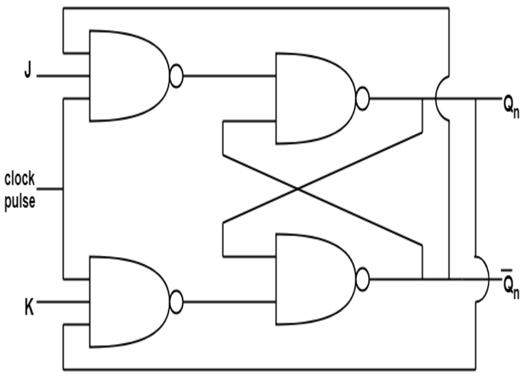
CLK	D	Q_{n+1}
0	x	Q_n
1	0	D
1	1	D

```
int S, R, Q=0,NQ;
int s, r;
int CK = 1;
int D;
void setup()
{
  pinMode(3,INPUT);
  pinMode(8,OUTPUT);
  pinMode(7,OUTPUT);
}
void loop()
{
  D = digitalRead(3);

  S = D;
  R = !D;
  s = !(CK&&S);
  r = !(CK&&R);
  Q=!(s&&NQ);
  NQ=!(r&&Q);

   digitalWrite(7,Q);
   digitalWrite(8,NQ);
}
```

Problem 20. D Flip Flop using NOR SR Latch

3.4 JK Flip Flop

Problem 21. JK Flip Flop using NAND SR Latch



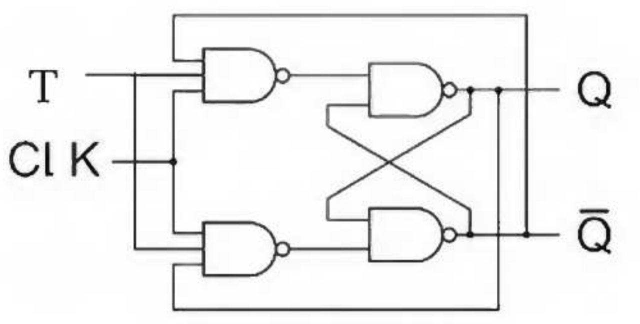
CLK	J	K	Q_{n+1}	Condition
1	0	0	Q_n	Memory
1	0	1	0	-
1	1	0	1	-
1	1	1	$\overline{Q_n}$	Toggle
0	x	x	Q_n	Memory

```
int J, K , Q=0,NQ;
int s, r;
int CK = 1;
void setup()
{
  pinMode(3 ,INPUT);
  pinMode(4 ,INPUT);
  pinMode(8 ,OUTPUT);
  pinMode(7 ,OUTPUT);
}
void loop()
{
  J = digitalRead(3);
  K = digitalRead(4);

  s = !(CK&&J&&NQ);
  r = !(CK&&K&&Q);
  Q=!(s&&NQ);
  NQ=!(r&&Q);

 // As toggle is at J =1, K =1
// To observe the toggle delay is
   given
  if (J == HIGH, K == HIGH) {

  digitalWrite(7 ,Q);
  delay(1000);
  digitalWrite(8 ,NQ);
  }
  else {

  digitalWrite(7 ,Q);
  digitalWrite(8 ,NQ);
  }

}
```

Problem 22. JR Flip Flop using NOR SR Latch

3.5 T Flip Flop

Problem 23. T Flip Flop using NAND SR Latch



CLK	T	Q_{n+1}	Condition
0	x	Q_n	Memory
1	0	Q_n	Memory
1	1	$\overline{Q_n}$	Toggle

```
int T, Q=0,NQ;
int s, r;
int CK = 1;
void setup()
{
  pinMode(3 ,INPUT);
  pinMode(8 ,OUTPUT);
  pinMode(7 ,OUTPUT);
}
void loop()
{
  T = digitalRead(3);

if (T == HIGH) {
  s = !(CK&&T&&NQ);
  r = !(CK&&T&&Q);
  Q=!(s&&NQ);
  NQ=!(r&&Q);

   digitalWrite(7,Q);
   delay(1000);
   digitalWrite(8,NQ);
   }
   else {
    s = !(CK&&T&&NQ);
   r = !(CK&&T&&Q);
  Q=!(s&&NQ);
  NQ=!(r&&Q);

   digitalWrite(7,Q);
   digitalWrite(8,NQ);
   }
}
```

Problem 24. T Flip Flop using NOR SR Latch

3.6 Counters

Problem 25. Design 4-Bit up Counter and give the output to 7447 IC, observe the result on SSD

Problem 26. Design 4-Bit Down Counter and give the output to 7447 IC, observe the result on SSD