Scilab Textbook Companion for Digital Integrated Circuits: A Design Perspective by J. Rabaey[1]

Created by
Sandeep Kumar C
Btech
Electronics Engineering
Sreenidhi Institute Of Science And Technology
College Teacher
None
Cross-Checked by
None

August 29, 2018

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: Digital Integrated Circuits: A Design Perspective

Author: J. Rabaey

Publisher: Prentice Hall, India

Edition: 2

Year: 2002

ISBN: 81-203-1244-9

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes

Chapter 2

The Devices

Scilab code Exa 2.1 `Built in Voltage of PN Junction`

```
//Example 2.1, Page Number 20
//Built-in Voltage of PN Junction
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Na=10^15        //Acceptor ions doping densities in
   atoms per cm cube (atoms/cm^3)
Nd=10^16       //Donor ions doping densities in atoms
    per cm cube (atoms/cm^3)
T=300           //Temperature in Kelvin(K)
phi_t=26          //Thermal Voltage in milli Volts(mV
   ) at 300 kelvin
ni=1.5*10^10      //Intrinsic carrier concentration
   of pure semiconductor in inverse cm cube (1/cm^3)

//Outputs
phi_o= phi_t*log((Na*Nd)/(ni*ni))       // Built-in
   Potential of P-N junction at zero bias in milli
   volts(mV)
```

```
//Results
mprintf("Built-in potential of P-N junction at 300k
   phi_o: %.2f mV",phi_o);

//Outputs
//Built-in potential of P-N junction at 300k phi_o:
   637.46 mV
```

Scilab code Exa 2.2 `Analysis of Diode Network`

```
//Example 2.2, Page Number 27
//Analysis of Diode Network
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vs=3                    //Source voltage in volts(V)
Rs=10*(10^3)            //Series resistance in ohms(
      )
Is=0.5 *(10^-16)        //Saturation current of the
   diode in amperes(A)
phi_t=26*(10^-3)                //Thermal voltage in
   volts(V)

//Using ideal diode equation and solving the non
   linear equation using either numerical or
   iterative techniques
Id=0.224 *(10^-3)       //Diode current in amperes(A)

//Outputs
Vd=Vs-Rs*Id             //Voltage across the diode in
    volts(V)

```

```
//Results
mprintf("Voltage across diode Vd: %.2f V",Vd);

//Output
// Voltage across diode Vd: 0.76 V
```

Scilab code Exa 2.3 Analysis of Diode Network

```
//Example 2.3, Page Number 29
//Junction Capacitance
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
//Consider a silicon junction with following
   parameters:
Cjo=0.5*((10^-15)/(10^-12))  //Capacitance under
   zero-bias in farads per metre square(F/m^2)
Ad=12*(10^-6)    //Junction area in metre square(m
   ^2)
phi_o=0.64    //Built-in potential at zero bias in
   volts(V)
Vd=-5      //Reverse bias voltage in volts(V)

//Outputs
Cj=Cjo/(sqrt(1-(Vd/phi_o)))      //Junction
   capacitance in farads per metre square(F/m^2)
//The above equation is valid under conditions that
   the P-N junction is a abrupt junction.

//Results
mprintf("Junction Capacitance Cj: %.6f F/m^2",Cj);

//Output
```

```
//Junction Capacitance is: 0.000168 F/m^2 (or) 0.168
    fF/ m ^2
```

Scilab code Exa 2.4 Average Junction Capacitance

```
//Example 2.4, Page Number 30
//Average Junction Capacitance
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
V_high=0       //Diode high level voltage in volts(V)
V_low=-5       //Diode low level voltage in volts(V)
m=0.5          //Grading co-efficient is equal to
   (1/2) for abrupt junction and (1/3) for linear
   graded junction
phi_o=0.64        //Built-in potential for zero bias
   in volts(V)
Cjo=0.5*((10^-15)/(10^-12))  //Capacitance under
   zero-bias in farads per metre square(F/m^2)

//Outputs
Keq=-(phi_o^m)*[((phi_o-V_high)^(1-m))-((phi_o-V_low
   )^(1-m))]/((V_high-V_low)*(1-m))...//Device
   parameter

Ceq=Keq*Cjo     //Average junction capacitance in
   farads per metre square(F/m^2)

//Results
mprintf("Keq: %.4f",Keq);
mprintf("\nAverage junction capacitance at the diode
    Ceq: %.6f F/m^2",Ceq);

```

```
//Outputs
// Keq: 0.5040
//Average junction capacitance at the diode Ceq
   :0.000252 F/m^2 (or) 0.2520 fF/ m ^2
```

Scilab code Exa 2.5 Mean Transit Times

```
//Example 2.5, Page Number 31
//Mean Transit Times
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
//Mean transit times for the diode of problem 2.1
Wn=5*(10^-6)         //Width of N-depletion layer
   region in micro meters(m)
W2=0.15*(10^-6)    //Small change in width of N-
   depletion layer region in meters(m)
Dp1=10*(10^-4)         //Diffusion co-efficient in
   metres square per second(m^2/sec)
Dp2=25*(10^-4)          //Diffusion co-efficient in
   metres square per second(m^2/sec)
Wp=0.7*(10^-6)        //Width of P-depletion layer
   region in meters(m)
W1=0.03*(10^-6)        //Small change in width of P-
   depletion layer region in meters(m)

//Outputs
tou_tp=(((Wn-W2)^2)/(2*Dp1))      //Mean transit time
    of P-region in seconds(sec)
tou_tn=(((Wp-W1)^2)/(2*Dp2))      //Mean transit time
    of N-region in seconds(sec)

//Results
```

```
mprintf("Transit time for P-region tou_tp: %.11f secs",tou_tp)
mprintf("\nTransit time for N-region tou_tn: %.11f secs",tou_tn)

//Outputs
//Transit time for P-region tou_tp: 0.00000001176 secs (or) 11.76nsec
//Transit time for N-region tou_tn: 0.00000000009 secs (or) 0.09nsec
```

Scilab code Exa 2.6 Diffusion Capacitance

```
//Example 2.6, Page Number 32
//Diffusion Capacitance
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Is=0.5*(10^-16)      //Saturation current of diode in amperes(A)
tou_t=1*(10^-9)         //Mean time in seconds(secs)
 t =26*(10^-3)                //Thermal voltage in volts(V)
Vd=0.75              //Forward bias diode voltage in volts(V)

//Outputs
Id=Is*(exp((Vd/ t ))-1) //Diode equation to obtain diode current(Id)in amperes(A)
Cd=(tou_t*Id)/ t              //Diffusion capacitance in farads(f)

//Results
```

```
mprintf("Diffusion Capacitance Cd: %.13f farads",Cd)

//Output
//Diffusion Capacitance Cd: 0.0000000000065 farads (or) 6.5 picofarads
```

Scilab code Exa 2.7 `Diode Transient Response`

```
//Example 2.7, Page Number 35
//Diode Transient Response
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
R_src=50*(10^3)      //Source resistance in ohms(  )
I1=1*(10^-3)          //Initial current in amperes(A)
I2=-0.1*(10^-3)          //Final current in amperes(A)
Is=2*(10^-16)      // Saturation current of the diode in amperes(A)
Cjo=0.2*(10^-12)      //Capacitance of junction at zero bias in farads(F)
tou_t=5*(10^-9)         // Mean time in seconds(secs)
phi_o=0.65          //Built-in potential at zero bias in volts(V)
phi_t=26*(10^-3)                //Thermal voltage in volts(V)
m=0.5         //Grading co-efficient is equal to (1/2) for abrupt junction and (1/3) for linear graded junction
Keq=0.51               //Device parameter

//Outputs
Id=Is*(exp((Vd/phi_t))-1) //Diode equation to obtain
```

```
diode current(Id)in amperes(A)
Vd_on=phi_t*log(Id/Is)     //Steady state voltage of diode when diode is on in volts(V)
Vd_off=I2*R_src          //Steady state voltage of diode when diode is off in volts(V)
t1=tou_t*log((I1-I2)/-I2)  //Length of various intervals in turn off transients in seconds(secs)
Cj=Keq*Cjo    //Average junction capacitance in farads per metre square(F/m^2)
tou_90_off=2.2*tou_t          //Time to reach 90% of final value(90% transition point[t2-t1]]) in seconds(secs)
T_off=t1+tou_90_off         //Total turn off time in seconds(secs)
t3=R_src*Cj*log((I1-I2)/I1)  //Length of various intervals in turn on transients in seconds(secs)
tou_90_on=2.2*tou_t          //Time to reach 90% of final value(90% transition point[t4-t3]]) in seconds(secs)
T_on=t3+tou_90_on            ////Total turn on time in seconds(secs)

//Results
mprintf("Steady state voltage when diode is on(Vd_on) is is: %.2f volts",Vd_on)
mprintf("\nSteady state voltage when diode is off is(Vd_on) is: %.1f volts",Vd_off)
mprintf("\nTransient time t1 is: %.13f secs",t1)
mprintf("\nAverage junction capacitance(Cj): %.15f farads",Cj)
mprintf("\nTransition point at 90 percent of final value(t2-t1): %.10f secs",tou_90_off)
mprintf("\nTotal turn off time(T_off): %.13f secs",T_off)
mprintf("\nTransient time t3: %.13f secs",t3)
mprintf("\nTransition point at 90 percent of final value(t4-t3): %.10f secs",tou_90_on)
mprintf("\nTotal turn off time(T_on): %.13f secs",
    T_on)

//Outputs
// Steady state voltage when diode is on(Vd_on) is
    is: 0.75 volts
//Steady state voltage when diode is off is(Vd_on)
    is: −5.0 volts
//Transient time t1 is: 0.0000000119895 secs (or)
    11.9 nsecs
//Average junction capacitance(Cj):
    0.000000000000102 farads (or) 0.102 picofarads
//Transition point at 90 percent of final value(t2−
    t1): 0.0000000110 secs (or) 11.0 nsecs
//Total turn off time(T_off): 0.0000000229895 secs (
    or) 22.9895 nsecs
//Transient time t3: 0.0000000004861 secs (or)
    0.4861 nsecs
//Transition point at 90 percent of final value(t4−
    t3): 0.0000000110 secs (or) 11.0 nsecs
//Total turn off time(T_on): 0.0000000114861 secs (
    or) 11.4861 nsecs
```

Scilab code Exa 2.8 Threshold Voltage of an NMOS Transistor

```
//Example 2.8, Page Number 43
//Threshold Voltage of an NMOS Transistor
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vto=0.75              //Threshold voltage for zero
    Substrate bias voltage(V_sb=0) in volts(V)
tou=0.54                    //Body−effect coefficient(
    Gamma)
```

```
V_sb=5            //Substrate bias voltage in volts(V)
Vb=0.6      //Voltage drop across depletion region at inversion(Vb=-2* f ) in volts(V)

//Outputs
Vt=Vto+(tou*(sqrt((Vb)+V_sb)-sqrt(Vb)))  //Threshold voltage at high level in volts(V)

//Results
mprintf("Threshold voltage Vt: %.12f volts",Vt)
//The obtained threshold voltage is more than twice the threshold under zero bias conditions.

//Output
// Threshold voltage Vt: 1.609591031759 volts
```

Scilab code Exa 2.9 `MOS Transistor Capacitances`

```
//Example 2.9, Page Number 50
//MOS Transistor Capacitances
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
t_ox=20*(10^-9)      //Thickness of oxide layer in metres(m)
epsilon_ox=3.5*(10^-11)     //Electrical permitivity of oxide layer in farad per metre(F/m)
L=1.2*(10^-6)        //Length of the channel in metres(m)
W=1.8*(10^-6)        //Width of the channel in metres(m)
Ls=3.6*(10^-6)       //Lenght of side wall of source
```

```
    junction in metres(m)
Ld=3.6*(10^-6)      //Lenght of diffusion wall of
    source junction in metres(m)
xd=0.15*(10^-6)      //Lateral diffusion in metres(m
    )
Cjo=3*(10^-4)  //Capacitance under zero-bias in
    farads per metre square(F/m^2)
C_jswo=8*(10^-10)   //Side wall capacitance under
    zero-bias in farads per metre (F/m)

//Outputs
Cox=(epsilon_ox/t_ox)      //Gate capacitance per
    unit area in farads per metre square(F/m^2)
Cg=W*L*Cox             //Total gate capacitance in
    farads(F)
C_gso=W*xd*Cox          //Overlap capacitance in
    farads(F)
C_channel=Cg-(2*C_gso)    //Channel capacitance which
     splits between source,drain,bulk terminals in
    farads(F)
Cd_bottom=Cjo*Ld*W          //Diffusion capacitance
    of bottom wall in farad(F)
Cd_side=C_jswo*(2*Ld+W)  //Diffusion capacitance of
    side wall in fard(F)


//Results
mprintf("\nGate capacitance per unit area Cox: %.5f
    farad/metre^2",Cox)
mprintf("\nTotal gate capacitance Cg: %.20f farad",
    Cg)
mprintf("\nOverlap capacitance C_gso: %.20f farad",
    C_gso)
mprintf("\nChannel capacitance C_channel: %.20f
    farad",C_channel)
mprintf("\nDiffusion capacitance of bottom wall
    Cd_bottom: %.20f farad",Cd_bottom)
mprintf("\nDiffusion capacitance of side wall
   Cd_side: %.20f farad",Cd_side)

//The diffusion capacitance seems to dominate gate
   capacitance.Advanced processes are reduce the
   diffusion capacitance by using materials such as
   SiO2 to isolate devices.This approach is  called
   Trench Isolation

//Outputs
//Gate capacitance per unit area Gox: 0.00175 farad/
   metre^2 (or) 1.75 fF/ m ^2
//Total gate capacitance Cg: 0.00000000000000378000
   farad (or) 3.78000 fF
//Overlap capacitance C_gso: 0.00000000000000047250
   farad (or) 0.47250 fF
//Channel capacitance C_channel:
   0.00000000000000283500 farad (or) 2.83500 fF
//Diffusion capacitance of bottom wall Cd_bottom:
   0.00000000000000194400 farad (or) 1.94400 fF
//Diffusion capacitance of side wall Cd_side:
   0.00000000000000720000 farad (or) 7.20000 fF
```

Scilab code Exa 2.11 Subthreshold Slope

```
//Example 2.11, Page Number 56
//Subthreshold Slope
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
slope=121          //Inverse Slope obtained from plot
    between Id vs Vgs( ln  (Id)/  Vgs  ) in milli
   volts per decade (mV/decade)
q=1.6*(10^-19)       //Charge in coulombs(C)
```

```
K=1.38064852*(10^23)        //Boltzman constant in (m2
    *kg*s^-2*K^-1)
T=300                       //Temperature in kelvin(K)

//Outputs
//Expression (  ln  (Id)/  Vgs  )^-1=(K*T/q)*log(10)
    *(1+   ),where (K*T/q)*log(10) evaluates to 60mv/
    decade
alpha=(slope/60)-1     //Sloving for alpha-factor
    using above expression

//Results
mprintf("alpha-factor is: %.2f",alpha);

//Output
//alpha-factor is: 1.02
```

Scilab code Exa 2.15 Dynamic Behaviour of an npn Transistor

```
//Example 2.15, Page Number 77
//Dynamic Behaviour of an npn Transistor
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
beta_f=100                  //Forward current gain
Vbe_on=0.75           //Forward bias Voltage across
    base and emitter in volts(V)
tou_f=10*(10^-12)       //Mean forward transit time
    in seconds(secs)
C_beo=20*(10^-15)      //Zero bias base emitter
    parasitic capacitance in farads(F)
C_bco=22*(10^-15)      //Zero bias base collector
    parasitic capacitance in farads(F)
```

```
C_cso=20*(10^-15)      //Zero bias base collector
   substrate parasitic capacitance in farads(F)
m=0.33        //Grading co−efficient is equal to
   (1/2) for abrupt junction and (1/3) for linear
   graded junction
phi_o=0.75                //Built in potential in
   volts(V)
Vcc=2          //Supply in volts(V)
Vo=1           //Initial voltage in volts(V)
Rb=5*(10^3)    //Base resistance in ohms(  )
Vce=2          //Voltage across collector and emitter
   (Vce=Vcc) in volts(V)
Vbe_high=0.75   //Voltage across base emmiter in
   forward bias in volts(V)
Vbe_low=0   //Voltage across base emmiter in reverse
    bias in volts(V)
Vbc_high=2   //Voltage across base collector in
   forward bias in volts(V)
Vbc_low=1.25   //Voltage across base collector in
   forward bias in volts(V)


//Outputs
Ib_to=(Vo-Vbe_on)/Rb    //Base current at t=0 in
   amperes(A)
Ib_t=(0-Vbe_on)/Rb        //Base current after
   switching input(changes direction) in amperes(A)
t_base=(beta_f*tou_f)*log((Ib_to-Ib_t)/-Ib_t)    //
   Time to remove excess base charge in seconds(secs
   )
Keq_be=-(phi_o^m)*[((phi_o-Vbe_high)^(1-m))-((phi_o-
   Vbe_low)^(1-m))]/((Vbe_high-Vbe_low)*(1-m))
          //Device parameter
//Similarly for Base−Collector junction the above
   equation gives Keq_bc:
Keq_bc=0.68            //Device parameter
t_depl=2.2*Rb*((Keq_be*C_beo)+(Keq_bc*C_bco))
          //Depletion region time constant in
```

```
seconds(secs)

//Results
mprintf("\nBase current at t=0 is Ib_to : %.7f
    amperes",Ib_to);
mprintf("\nBase current after switching input is
    Ib_t: %.7f amperes",Ib_t);
mprintf("\nTime to remove excess base charge is
    t_base: %.13f seconds",t_base);
mprintf("\nKeq_be is Keq_be: %.2f ",Keq_be);
mprintf("\nKeq_bc is Keq_bc: %.2f",Keq_bc);
mprintf("\nTime to remove excess base charge is
    t_depl: %.13f seconds",t_depl);

//Outputs
//Base current at t=0 is Ib_to : 0.0000500 amperes (
    or) 50.0 A
//Base current after switching input is Ib_t:
    -0.0001500 amperes (or) 150.0  A
//Time to remove excess base charge is t_base:
    0.0000000002876 seconds (or) 287.6 psecs
//Keq_be is Keq_be: 1.49
//Keq_bc is Keq_bc: 0.68
//Time to remove excess base charge is t_depl:
    0.0000000004929 seconds (or) 492.9 psecs
```

Scilab code Exa 2.17 `MOS Transistor Process Variations`

```
//Example 2.17, Page Number 85
//MOS Transistor Process Variations
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
```

```
Vgs=5      // Gate source voltage in volts(V)
Vds=5      //Drain source voltage in volts(V)
W=1.8*(10^-6)    //Channel width in metres(m)
L=0.9*(10^-6)     //Effective channel length in metres(m)
Kn=19.6*(10^-6)     //Device parameter in A/V^2
Vt=0.75             //Threshold voltage in volts(V)

//Outputs
Id=(Kn/2)*(W/L)*((Vgs-Vt)^2)        //Nominal diode current in amperes(A)

//The above parameters are subject to process variation.So,considering the following conditions :1)10% variation in Kn   2) 25% variation in Vt   3) Distribution of W and L has value 0.15 m   4) Variation of 0.5V in supply

Id_max=(((Kn+(0.1*Kn))/2)*((W+(0.15*(10^-6)))/(L-(0.15*(10^-6))))*((Vgs+0.5)-(Vt-(0.25*Vt)))^2)      //Maximum nominal diode current in amperes(A)
Id_min=(((Kn-(0.1*Kn))/2)*((W-(0.15*(10^-6)))/(L+(0.15*(10^-6))))*((Vgs-0.5)-(Vt+(0.25*Vt)))^2)     //Minimum nominal diode current in amperes(A)

//Results
mprintf("\nNominal diode current is Id: %.7f amperes",Id);
mprintf("\nMaximum nominal diode current is Id_max: %.7f amperes",Id_max);
mprintf("\nMinimum nominal diode current is Id_min: %.7f amperes",Id_min);

//Outputs
//Nominal diode current is Id: 0.0003540 amperes (or) 354.0  A
//Maximum nominal diode current is Id_max: 0.0006832
```

```
    amperes (or) 683.2  A
//Minimum nominal diode current is Id_min: 0.0001759
    amperes (or) 175.9  A
```

Chapter 3

The Inverter

Scilab code Exa 3.4 VTC of a CMOS Inverter

```
//Example 3.4, Page Number 127
//VTC of a CMOS Inverter
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Kn=80*(10^-6)     //Transistor parameters in A/V^2
Kp=27*(10^-6)       //Transistor parameter in A/V^2
Vdd=5       //Supply voltage in volts(V)
W_to_L_PMOS=5.4/1.2    //(W/L) ratio of PMOS
   transistor
W_to_L_NMOS=1.8/1.2    //(W/L) ratio of NMOS
   transistor
//DC Parameters:
V_ol=0              //Output lowel level transition
   voltage in volts(V)
V_oh=5              //Output high level transition
   voltage in volts(V)
Vtn=0.74                    //Threshold voltage of NMOS in
    volts(V)
```

```
Vtp=-0.74                //Threshold voltage of PMOS
   in volts(V)

//Outputs
//To obtain Vih,the following equations have to be
   solved:
//Kn*[(V_ih-Vtn)*Vout-(((Vout)^2)/2)]=(Kp/2)*[Vdd-
   V_ih-|Vtp|]^2 which yeilds:
V_ih=2.92                          //Input high level
   transition voltage in volts(V)
Vout=0.42                           //Output voltage in
   volts(V)
NM_H=V_oh-V_ih                      //Noise margin for
   logic high in volts(V)
//Similary,for V_il
V_il=2.06                 //Input low level transition
    voltage in volts(V)
NM_L=V_il-V_ol                      //Noise margin for
   logic low in volts(V)
//Given that r=1.01 where r=(sqrt(Kp /Kn ))
r=1.01
Vm=((r*(Vdd-abs(Vtp))+Vtn)/(1+r))             //
   Inverter threshold in volts(V)

//Results
mprintf("\nInput high level transition voltage V_ih:
    %.2f volts",V_ih);
mprintf("\nOutput voltage Vout: %.2f volts",Vout);
mprintf("\nNoise margin for logic high NM_H: %.2f
   volts",NM_H);
mprintf("\nInput low level transition voltage V_il:
   %.2f volts",V_il);
mprintf("\nNoise margin for logic low NM_L: %.2f
   volts",NM_L);
mprintf("\nInverter threshold Vm: %.2f volts",Vm);

//Outputs
//Input high level transition voltage V_ih: 2.92
```

```
   volts
//Output voltage Vout: 0.42 volts
//Noise margin for logic high NM_H: 2.08 volts
//Input low level transition voltage V_il: 2.06
   volts
//Noise margin for logic low NM_L: 2.06 volts
//Inverter threshold Vm: 2.51 volts
```

Scilab code Exa 3.5 Keq for a 5V CMOS Inverter

```
//Example 3.5, Page Number 130
//Keq for a 5V CMOS Inverter
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
m=0.5        //Grading co-efficient is equal to (1/2)
    for abrupt junction and (1/3) for linear graded
   junction
phi_o=0.6                //Built in potential in
   volts(V)
Vcc=5          //Supply in volts(V)
V_high_NMOS_HL=-5   //High level transition voltage
   for NMOS during high to low transition in volts(V
   )
V_low_NMOS_HL=-2.5   //Low level transition voltage
   for NMOS during high to low transition in volts(V
   )
V_high_PMOS_HL=-2.5   //High level transition
   voltage for NMOS during high to low transition in
    volts(V)
V_low_PMOS_HL=0   //Low level transition voltage for
    NMOS during high to low transition in volts(V)

```

```
V_high_NMOS_LH=0    //High level transition voltage
    for NMOS during low to high  transition in volts(
    V)
V_low_NMOS_LH=-2.5    //Low level transition voltage
    for NMOS during low to high transition in volts(V
    )
V_high_PMOS_LH=-5    //High level transition voltage
    for NMOS during low to high transition in volts(V
    )
V_low_PMOS_LH=-2.5    //Low level transition voltage
    for NMOS during low to high transition in volts(V
    )

//Outputs
Keq_NMOS_HL=-(phi_o^m)*[((phi_o-V_high_NMOS_HL)^(1-m
    ))-((phi_o-V_low_NMOS_HL)^(1-m))]/((
    V_high_NMOS_HL-V_low_NMOS_HL)*(1-m))         //
    Device parameter for NMOS
Keq_PMOS_HL=-(phi_o^m)*[((phi_o-V_high_PMOS_HL)^(1-m
    ))-((phi_o-V_low_PMOS_HL)^(1-m))]/((
    V_high_PMOS_HL-V_low_PMOS_HL)*(1-m))         //
    Device parameter for PMOS

Keq_NMOS_LH=-(phi_o^m)*[((phi_o-V_high_NMOS_LH)^(1-m
    ))-((phi_o-V_low_NMOS_LH)^(1-m))]/((
    V_high_NMOS_LH-V_low_NMOS_LH)*(1-m))         //
    Device parameter for NMOS
Keq_PMOS_LH=-(phi_o^m)*[((phi_o-V_high_PMOS_LH)^(1-m
    ))-((phi_o-V_low_PMOS_LH)^(1-m))]/((
    V_high_PMOS_LH-V_low_PMOS_LH)*(1-m))         //
    Device parameter for PMOS


//Results
mprintf("\nKeq_be_NMOS_HL: %.20f",Keq_NMOS_HL);
mprintf("\nKeq_be_PMOS_HL: %.20f",Keq_PMOS_HL);
mprintf("\nKeq_be_NMOS_LH: %.20f",Keq_NMOS_LH);
mprintf("\nKeq_be_PMOS_LH: %.20f",Keq_PMOS_LH);
```

```

//Outputs
//Keq_be_NMOS_HL: 0.37536968662700032000
//Keq_be_PMOS_HL: 0.61105453575886848000
//Keq_be_NMOS_LH: 0.61105453575886848000
//Keq_be_PMOS_LH: 0.37536968662700032000
```

Scilab code Exa 3.6 Capacitance of a 1 2um CMOS Inverter

```
//Example 3.6, Page Number 131
//Capacitance of a 1.2 m CMOS Inverter
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
//Considering a minimum size symmetrical CMOS
   inverter with its parameters.
Vdd=5                    //Supply voltage in volts(V)
lamda=0.6*(10^-6)              //Channel modulation in
    metres(m)

//Outputs
drain_area_NMOS=4*4*(lamda^2)           //Drain area
    formed by metal-diffusion contact of NMOS in
   metre square(m^2)
rect_contact_gate_NMOS=3*(lamda^2)         //
   Rectangle area between contact and gate of NMOS
   in metre square(m^2)
total_area_NMOS=drain_area_NMOS+
   rect_contact_gate_NMOS    //Total area of layers
   of NMOS in metre square(m^2)
perimeter_drain_NMOS=15*lamda              //
   Perimeter of drain in NMOS in metres(m)
sidewall_area_NMOS=19*(lamda^2)            //sidewall
```

```
    area formed by metal-diffusion contact of NMOS
   in metre square(m^2)
perimeter_sidewall_NMOS=15*lamda                    //
   Perimeter of sidewall in NMOS in metres(m)
total_area_PMOS=45*(lamda^2)    //Total area of
   layers of PMOS in metre square(m^2)
perimeter_drain_PMOS=19*lamda                    //
   Perimeter of drain in PMOS in metres(m)
sidewall_area_PMOS=45*(lamda^2)                  //sidewall
    area formed by metal-diffusion contact of PMOS
   in metre square(m^2)
perimeter_sidewall_PMOS=19*lamda                    //
   Perimeter of sidewall in PMOS in metres(m)

//Results
mprintf("\nTotal area of layers of NMOS : %.14f (
   metres^2)",total_area_NMOS);
mprintf("\nPerimeter of drain in NMOS: %.7f metres",
   perimeter_drain_NMOS);
mprintf("\nsidewall area formed by metal-diffusion
   contact of NMOS sidewall_area_NMOS: %.14f metres"
   ,sidewall_area_NMOS);
mprintf("\nPerimeter of sidewall in NMOS
   perimeter_sidewall_NMOS: %.7f metres",
   perimeter_sidewall_NMOS);
mprintf("\nTotal area of layers of PMOS
   total_area_PMOS: %.14f (metres^2)",
   total_area_PMOS);
mprintf("\nPerimeter of drain in PMOS
   perimeter_drain_PMOS: %.7f metres",
   perimeter_drain_PMOS);
mprintf("\nsidewall area formed by metal-diffusion
   contact of PMOS sidewall_area_PMOS: %.14f (metres
   ^2)",sidewall_area_PMOS);
mprintf("\nPerimeter of sidewall in PMOS
   perimeter_sidewall_PMOS: %.7f metres",
   perimeter_sidewall_PMOS);

```

```

//Outputs
//Total area of layers of NMOS total_area_NMOS : 0.00000000000684 (metres^2) *(or) 6.84 m ^2
//Perimeter of drain in NMOS perimeter_drain_NMOS: 0.0000090 metres (or) 9.00 m
//sidewall area formed by metal-diffusion contact of NMOS: 0.00000000000684 metres (or) 6.84 m ^2
//Perimeter of sidewall in NMOS perimeter_sidewall_NMOS: 0.0000090 metres (or) 9.0 m
//Total area of layers of PMOS total_area_PMOS: 0.00000000001620 (metres^2) (or) 16.20 m ^2
//Perimeter of drain in PMOS perimeter_drain_PMOS: 0.0000114 metres (or) 11.40 m
//sidewall area formed by metal-diffusion contact of PMOS sidewall_area_PMOS: 0.00000000001620 (metres^2) (or) 16.20 m ^2
//Perimeter of sidewall in PMOS perimeter_sidewall_PMOS: 0.0000114 metres (or) 11.40 m
```

Scilab code Exa 3.7 `Propogation Delay of a 1 2um CMOS Inverter`

```
//Example 3.7, Page Number 135
//Propogation Delay of a 1.2 m  CMOS Inverter
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vgs=5     // Gate source voltage in volts(V)
Vds=5     //Drain source voltage in volts(V)
Vdd=5      //Supply voltage in volts(V)
Wp=5.4*(10^-6)    //PMOS Channel width in metres(m)
```

```
Wn=1.8*(10^-6)    //NMOS Channel width in metres(m)
L_eff=0.9*(10^-6)     //Effective channel length in
   metres(m)
Kn=19.6*(10^-6)     //in A/V^2
Kp=5.4*(10^-6)      //in A/V^2
lamda_p=0.19                  //Channel modulation of
   PMOS in voltage inverse(V^-1)
lamda_n=0.06                  //Channel modulation of
   NMOS in voltage inverse(V^-1)
Vt0=0.74              //Threshold voltage in volts(V)

//Outputs
Idp_Vout_zero=(Kp/2)*(Wp/L_eff)*((Vgs-Vt0)^2)*(1+(
   lamda_p*Vds))        //Current through PMOS in
   amperes(A) when output voltage is zero
Idp_Vout=Kp*(Wp/L_eff)*((Vgs-Vt0)*(Vdd/2)-((Vdd)^2)
   /8)        //Current through PMOS in amperes(A)
   when output voltage is not zero
Idn_Vout_zero=(Kn/2)*(Wn/L_eff)*((Vgs-Vt0)^2)*(1+(
   lamda_n*Vds))        //Current through NMOS in
   amperes(A) when output voltage is zero
Idn_Vout=Kn*(Wn/L_eff)*((Vgs-Vt0)*(Vdd/2)-((Vdd)^2)
   /8)        //Current through NMOS in amperes(A)
   when output voltage is zero

//Results
mprintf("\nCurrent through PMOS Idp(Vout=0): %.6f
   amperes",Idp_Vout_zero);
mprintf("\nCurrent through PMOS Idp(Vout=2.5): %.6f
   amperes",Idp_Vout);
mprintf("\nCurrent through NMOS Idn(Vout=0): %.6f
   amperes",Idn_Vout_zero);
mprintf("\nCurrent through NMOS Idn(Vout=2.5): %.6f
   amperes",Idn_Vout);

//Outputs
//Current through PMOS Idp(Vout=0): 0.000573 amperes
    (or) 0.573mA
```

```
//Current through PMOS Idp(Vout=2.5): 0.000244 amperes (or) 0.244mA
//Current through NMOS Idn(Vout=0): 0.000462 amperes (or) 0.46.2mA
//Current through NMOS Idn(Vout=2.5): 0.000295 amperes (or) 0.295mA
```

Scilab code Exa 3.11 `VTC of an RTL Inverter`

```
//Example 3.11, Page Number 151
//VTC of an RTL Inverter
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vcc=5      //Supply voltage in volts(V)
Rc=1*(10^3)        //Collector resistance in ohms(  )
Rb=10*(10^3)        //Base resistance in ohms(  )
Re=20        //Emitter resistance in ohms(  )
Rc1=75        //Series collector resistance in ohms(  )
Vce_sat=0.1          //Collector-Emmiter saturation voltage in volts(V)
Vbe_eos=0.8             //Base-Emmiter voltage in volts(V)
beta_f=100               //Maximum forward current gain
//The major discrepancy is the value to V_ol due to which dc problems occur and to have a reasonable V_ol value Vin_eos>Vcc is maintained to reduce the gain in transient region
Vcc1=0.38          //Supply voltgage in volts(V)

//Outputs
```

```
Ib=((Vcc-Vce_sat)/(Rc*beta_f))              //Base
   current in amperes(A)
Vin_eos=Vbe_eos+(Ib*Rb)                     //Input
   voltage at eos in volts(V)
//In the saturation mode,the collector current Ic:
Ic=Vcc1/Rc1                                 //Collector
   current in amperes(A)
Vce_extra=Ic*(Rc1+Re)                       //Extra voltage
    drop caused over collector-emitter terminals in
   volts(V)

//Results
mprintf("\nBase Current Ib: %.7f amperes",Ib);
mprintf("\nInput Voltage at eos Vin_eos: %.5f volts"
   ,Vin_eos);
mprintf("\nCollector Current Ic: %.4f amperes",Ic);
mprintf("\nExtra voltage drop caused over collector-
   emitter terminals: %.2f volts",Vce_extra);

//Outputs
//Base Current Ib: 0.0000490 amperes (or) 49.0 A
//Input Voltage at eos Vin_eos: 1.29000 volts
//Collector Current Ic: 0.0051 amperes (or) 5.1mA
//Extra voltage drop caused over collector-emitter
   terminals: 0.48 volts
```

Scilab code Exa 3.12 Effect of Fanout on VTC of the RTL Inverter

```
//Example 3.12, Page Number 153
//Effect of Fanout on VTC of the RTL Inverter
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
```

```
Vcc=5       //Supply voltage in volts(V)
Rc=1*(10^3)        //Collector resistance in ohms(   )
Rb=10*(10^3)        //Base resistance in ohms(   )
//Re=20        //Emitter resistance in ohms(   )
//Rc1=75        //Series collector resistance in ohms
   (   )
Vbe_sat=0.8          //Base−Emmiter saturation
   voltage in volts(V)
//Vcc1=0.38          //Supply voltgage in volts(V)
N=5       //Number of fanouts

//Outputs
Vout=(Vcc+(N*(Rc/Rb)*Vbe_sat))/(1+(N*(Rc/Rb)))
             //Output voltage in volts(V)
//For large values for N,Vout eventually approaches
   Vbe_sat=0.8V which means that the NM_H is reduced
    to zero(or is even negative)

//Results
mprintf("\nOutput Voltage Vout: %.2f volts",Vout);


//Outputs
//Output Voltage Vout: 3.60 volts
```

Scilab code Exa 3.14 `VTC of ECL Gate`

```
//Example 3.14, Page Number 163
//VTC of ECL Gate
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vcc=0       //Supply voltage in volts(V)
```

```
Vee=-5      //Supply voltage in volts(V)
Vref=-0.95 //Reference voltage in volts(V)
Iee=0.5*(10^-3)      //Coupled emmiter current in amperes(A)
Rc=1*(10^3)        //Collector resistance in ohms(  )
Rb=50*(10^3)        //Base resistance in ohms(  )
Is=10^-17            //Transport saturation current in amperes(A)
Ic=0.1*(10^-3)         //Collector current in amperes (A)
beta_f=100               //Maximun forward current gain
phi_t=26*(10^-3)         //Thermal voltage in volts(A)
Vswing=Iee*Rc          //Transition voltage swing in volts(V)

//Outputs
Vbe_on=phi_t*log(Ic/Is)      //Base-Emitter voltage in volts(V)
V_oh=Vcc-Vbe_on            //Output high level transition voltage in volts(V)
V_ol=Vcc-Vbe_on-(Vswing)     //Output low level transition voltage in volts(V)
V_ih=Vref+(phi_t*log(((Vswing)/(2*phi_t))-1))   // Input high level transition voltage in volts(V)
V_il=Vref-(phi_t*log(((Vswing)/(2*phi_t))-1))   // Input low level transition voltage in volts(V)
Vm=Vref                                        //Inverter threshold in volts(V)

//Results
mprintf("\nBase Emmiter voltage in volts Vbe_on : %.2f volts",Vbe_on);
mprintf("\nVbe_oh: %.2f volts",V_oh);
mprintf("\nVbe_ol: %.2f volts",V_ol);
mprintf("\nVbe_ih: %.2f volts",V_ih);
mprintf("\nVbe_il: %.2f volts",V_il);
mprintf("\nInverter threshold Vm: %.2f volts",Vm);
```

```

//Outputs
//Base Emmiter voltage in volts Vbe_on : 0.77 volts
//Vbe_oh: -0.77 volts
//Vbe_ol: -1.27 volts
//Vbe_ih: -0.89 volts
//Vbe_il: -1.00 volts
//Inverter threshold Vm: -0.94 volts
```

Scilab code Exa 3.15 `Switching the Differential Pair`

```
//Example 3.15, Page Number 169
//Switching the Differential Pair
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
rc=75       //Transient collector resistance in ohms ( )
rb=120       //Transient base resistance in ohms( )
re=20       //Transient emitter resistance in ohms( )
Rc=1*(10^3)  //Collector resistance in ohms( )
C_be1=20*(10^-15)     //Base-Emitter junction capacitance in farads(F)
C_bc1=22*(10^-15)     //Base-collector junction capacitance in farads(F)
C_cs1=47*(10^-15)     //Collector-Source junction capacitance in farads(F)
Iee=0.5*(10^-3)     //Coupled emmiter current in amperes(A)
tou_f=10*(10^-12)     //Ideal forward transit time in secs
Keq_be=3.35
```

```
Keq_bc=0.75
alpha=5                       //Empirical factor
//Empirical factor(   ) is 2 and 5 for the 50% and 90
   % points respectively

//Outputs
Cd1=(tou_f/Rc)          //Diffusion capacitance in
   farads(F)
Cin_j=(Keq_bc*C_bc1)+((Keq_be*C_be1)/2)         //
   Equivalent input capacitance
tdp=rb*((2.2*Cin_j)+(alpha*Cd1))              //
   Propogation delay for collector current to reach
   90% of their value in seconds(secs)

//Results
mprintf("\nDiffusion capacitance at base during
   single transition Cd1 : %.16f farads",Cd1);
mprintf("\nEquivalent input capacitance Cin_j : %.16
   f farads",Cin_j);
mprintf("\nPropogation delay for collector current
   to reach ninty percent point tdp : %.13f seconds"
   ,tdp);

//Outputs
//Diffusion capacitance at base during single
   transition Cd1 : 0.0000000000000100 farads (or)
   10.0fF
//Equivalent input capacitance Cin_j :
   0.0000000000000500 farads (or) 50.0fF
//Propogation delay for collector current to reach
   ninty percent point tdp : 0.0000000000192 seconds
    (or) 19.2 picosecs
```

Scilab code Exa 3.16 `Load Capacitances of an ECL Gate`

```
//Example 3.16, Page Number 170
//Load Capacitances of an ECL Gate
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Cd3=0.2*(10^-15)          //Diffusion capacitance for
    high base resistance(Rb)in farads(F)
C_bc3=0.91*(10^-15)      //Base-Emitter junction
   capacitance in farads(F)
C_bc1=0.84*(10^-15)      //Base-collector junction
   capacitance in farads(F)
C_cs3=0.51*(10^-15)      //Collector-Source junction
   capacitance in farads(F)

//Outputs
Cc=(C_cs3*47)+(C_bc1*22)+(C_bc3*22)+Cd3     //
   Collector capacitance in farads(F)

//Results
mprintf("\nCollector capacitance Cc : %.16f farads",
   Cc);

//Output
//Collector capacitance Cc : 0.0000000000000627
   farads (or) 62.7 fF
```

Scilab code Exa 3.17 ECL Transient Response

```
//Example 3.17, Page Number 173
//ECL Transient Response
//Scilab 6.0.1
//Windows 10
clc;
```

```

//Inputs
Vcc=0       //Supply voltage in volts(V)
Vee=-5      //Supply voltage in volts(V)
Rc=1*(10^3)        //Collector resistance in ohms(  )
Rb=50*(10^3)        //Base resistance in ohms(  )
tdp=20*(10^-12)          //Propogation delay in seconds(secs)
Iee=0.5*(10^-3)      //Coupled emmiter current in amperes(A)
Vswing=Iee*Rc         //Transition voltage swing in volts(V)
Cl=60*(10^-15)          //Load capacitance farads(F)
Cc=62.7*(10^-15)     //Collector capacitance in farads(F)
V_oh=-0.7           //Output high level transition voltage in volts(V)

//Outputs
t_discharge=(0.5*Cl*Rb*Vswing)/(V_oh-Vee-(Vswing/4))              //Discharge time
tp_hl=tdp+t_discharge                         //Delay for high to low transition in seconds(secs)
t_charge=0.69*Rc*Cc                     //Charge time in seconds(secs)
tp_lh=tdp+t_charge                            //Delay for low to high transition in seconds(secs)
tp=(tp_hl+tp_lh)/2                            // Propogation delay in seconds(secs)
//Results
mprintf("\nDischarge time t_discharge : %.13f seconds",t_discharge);
mprintf("\nDelay for high to low transition tp_hl: %.13f seconds",tp_hl);
mprintf("\nCharge time t_charge: %.13f seconds",t_charge);
mprintf("\nDelay for low to high transition tp_lh: %.13f seconds",tp_lh);
mprintf("\nPropogation delay tp: %.13f seconds",tp);

//Outputs
//Discharge time t_discharge : 0.0000000001796
   seconds (or) 179.6 picosecs
//Delay for high to low transition tp_hl:
   0.0000000001996 seconds (or) 199.6 picosecs
//Charge time t_charge: 0.0000000000433 seconds (or)
    43.3 picosecs
//Delay for low to high transition tp_lh:
   0.0000000000633 seconds (or) 63.3 picosecs
//Propogation delay tp: 0.0000000001315 seconds (or)
    131.5 picosecs
```

Scilab code Exa 3.18 Power Dissipation of ECL Inverter

```
//Example 3.18, Page Number 177
//Power Dissipation of ECL Inverter
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vcc=0      //Supply voltage in volts(V)
Vee=-5     //Supply voltage in volts(V)
Vref=-0.95 //Reference voltage in volts(V)
Iee=0.5*(10^-3)     //Coupled emitter current in
   amperes(A)
Rc=1*(10^3)       //Collector resistance in ohms(  )
Rb=50*(10^3)       //Base resistance in ohms(  )
Is=10^-17           //Transport saturation current
   in amperes(A)
Ic=0.1*(10^-3)        //Collector current in amperes
   (A)
beta_f=100              //Maximum forward current
```

```
   gain
phi_t=26*(10^-3)        //Thermal voltage in volts(A)
Vswing=Iee*Rc         //Transition voltage swing in
   volts(V)
Cl=60*(10^-15)          //Load capacitance in farads(
   F)
Cc=72.8*(10^-15)     //Collector capacitance in
   farads(F)
Ct=Cl+Cc              //Total switching capacitance in
   farads(F)
tp=127.5*(10^-12)            //Propogation delay in
   seconds(secs)

//Outputs
Vbe_on=phi_t*log(Ic/Is)      //Base-Emitter voltage
   in volts(V)
V_oh=Vcc-Vbe_on            //Output high level
   transition voltage in volts(V)
V_ol=Vcc-Vbe_on-(Vswing)     //Output low level
   transition voltage in volts(V)
f=(1/tp)                          //Maximum allowable
   switching frequency in hertz(Hz)
//Ignoring the power consumed in bias network
Pstat=(Vcc-Vee)*(Iee+(2*(((((V_oh+V_ol)/2)-Vee)/Rb)))
              //Static power consumption in watts(W)
Pdyn=Ct*(Vcc-Vee)*Vswing*f           //Dyanamic power
   consumption in watts(W)
Pt=Pstat+Pdyn                //Total power consumption
   in watts(W)

//Results
mprintf("\nStatic power consumption Pstat : %.4f
   watts",Pstat);
mprintf("\nDynamic power consumption Pdyn : %.4f
   watts",Pdyn);
mprintf("\nTotal power consumption Pt : %.4f watts",
   Pt);

```

```
//Outputs
//Static power consumption Pstat : 0.0033 watts (or)
    3.3 mW
//Dynamic power consumption Pdyn : 0.0026 watts (or)
    2.6 mW
//Total power consumption Pt : 0.0059 watts (or) 5.9
    mW
```

Chapter 4

Designing Combinational Logic Gates In CMOS

Scilab code Exa 4.2 `Computing Ron`

```
//Example 4.2, Page Number 194
//Computing Ron
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vdd=5           //Supply voltage in volts(V)
//Values from table 3.3, Pg no:136
Id_n1=0.46*(10^-3)          //Drain current in NMOS
   when (Vout=Vdd) in amperes(A)
Id_n2=0.29*(10^-3)          //Drain current in NMOS
   when (Vout=Vdd/2) in amperes(A)
Id_p1=0.57*(10^-3)          //Drain current in PMOS
   when (Vout=Vdd) in amperes(A)
Id_p2=0.24*(10^-3)          //Drain current in PMOS
   when (Vout=Vdd) in amperes(A)

//Outputs
```

```
Rn1=(1/2)*((Vdd/Id_n1)+(Vdd/(2*Id_n2)))          //
   Resistance when (W/Leff =2) in ohms(   )
Rn2=Rn1*2                   //Resistance when (W/Leff
   =1) in ohms(   )
Rp1=(1/2)*((Vdd/Id_p1)+(Vdd/(2*Id_p2)))          //
   Resistance when (W/Leff =6) in ohms(   )
Rp2=Rp1*6              //Resistance when (W/Leff =1) in
   ohms(   )


//Results
mprintf("\nResistance when (W/Leff =2) Rn1 : %.1f
   ohms",Rn1);
mprintf("\nResistance when (W/Leff =1) Rn2 : %.1f
   ohms",Rn2);
mprintf("\nResistance when (W/Leff =6) Rp1 : %.1f
   ohms",Rp1);
mprintf("\nResistance when (W/Leff =1) Rp1 : %.1f
   ohms",Rp2);

//Outputs
//Resistance when (W/Leff =2) Rn1 : 9745.1 ohms (or)
    9.7451 k
//Resistance when (W/Leff =1) Rn2 : 19490.2 ohms(or)
    19.4902 k
//Resistance when (W/Leff =6) Rp1 : 9594.2 ohms (or)
    9.5942 k
//Resistance when (W/Leff =1) Rp1 : 57565.7 ohms (or
   ) 57.5657 k
```

Scilab code Exa 4.3 A Four Input Complimentary CMOS NAND Gate

```
//Example 4.3, Page Number 200
//A Four-Input Complimentary CMOS NAND Gate
//Scilab 6.0.1
```

```
//Windows 10
clc;

//Inputs
//Parameter Values for 1.2 m CMOS process
lamda_NMOS=0.06              //Channel modulation in volt inverse(1/V)
lamda_PMOS=0.19               //Channel modulation in volt inverse(1/V)

//Outputs
Area_PMOS=108*(lamda_PMOS^2)                         //Area of PMOS in micro metre square( m ^2)
Perimeter_PMOS=24*lamda_PMOS                         //Perimeter of PMOS in micro metre( m )
Area_NMOS=37*(lamda_NMOS^2)                         //Area of NMOS in micro metre square( m ^2)
Perimeter_NMOS=27*lamda_NMOS                         //Perimeter of NMOS in micro metre( m )

//Results
mprintf("\nArea of PMOS: %.2f  m ^2",Area_PMOS);
mprintf("\nPerimeter_PMOS: %.2f  m ",Perimeter_PMOS);
mprintf("\nArea of NMOS: %.2f  m ^2",Area_NMOS);
mprintf("\nPerimeter_NMOS: %.2f  m ",Perimeter_NMOS);


//Outputs
//Area of PMOS: 3.90  m ^2
//Perimeter_PMOS: 4.56  m
//Area of NMOS: 0.13  m ^2
//Perimeter_NMOS: 1.62  m
```

Scilab code Exa 4.7 Resistance of a Transmission Gate

```
//Example 4.7, Page Number 214
//Resistance of a Transmission Gate
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
//Parameter Values for 1.2 m CMOS process
Vdd=5                    //Supply voltage in volts(V)
Kn=19.6*(10^-6)                    //in ampere per volt
   square(A/V^2)
Kp=5.4*(10^-6)                    //in ampere per volt
   square(A/V^2)
Vtn=0.743                    //Threshold voltage of
   NMOS in volts(V)
Vtp1=-0.739                    //Threshold voltage of
   PMOS in volts(V)
Vtp=abs(Vtp1)

//Outputs
Geq=((Kn*((Vdd-Vtn)^2))+(Kp*((Vdd-Vtp)^2)))/(2*Vdd)
       //Equivalent conductance in mho
Req=1/Geq       //Equivalent resistance in ohms(  )

//Results
mprintf("\nEquivalent conductance Geq : %.7f mho",
   Geq);
mprintf("\nEquivalent resistance Req : %.4f ohms",
   Req);

//Output
//Equivalent conductance Geq : 0.0000453 mho (or)
   45.3 pico mho
//Equivalent resistance Req : 22063.5990 ohms (or)
   22.063 K

```

```
//Wrong Answer.
//Answer would be correct if formula is modified as
   (Replacing (2*Vdd) by (Vdd)) :
//Geq=((Kn*((Vdd-Vtn)^2))+(Kp*((Vdd-Vtp)^2)))/(Vdd)

//Output when equation is modified:
//Equivalent conductance Geq : 0.0000906 mho (or)
   90.6 pico mho
//Equivalent resistance Req : 11031.7995 ohms(or)
   11.031 K
```

Scilab code Exa 4.8 Pass Transistor Chain

```
//Example 4.8, Page Number 216
//Pass-Transistor Chain
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
//Parameter Values for 1.2 m CMOS process
Req=10*(10^3)          //Equivalent resistance in ohms
   (  )
C=10*(10^-15)          //Capacitance in farads(F)
tp_buf=0.5*(10^-9)             //Propogation delay of
   each buffer in seconds(secs)

//Outputs
m_opt=1.7*(sqrt(tp_buf/(C*Req)))       //Optimal
   number of switches between buffers

//Results
mprintf("\nOptimal number of switches between
   buffers m_opt : %.2f",m_opt);

```

```
//Output
//Optimal number of switches between buffers m_opt :
    3.80
```

Scilab code Exa 4.12 `Charge Redistribution`

```
//Example 4.12, Page Number 229
//Charge Redistribution
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vdd=-5                          //Supply voltage in volts
   (V)
//When top three transistors of the PDN are enabled
   and the internal capacitance hold diffusion and
   gate capacitance of neighbouring transistors.
C_internal=8.78*(10^-15)            //Internal
   capacitane in farads(F)
Cl=25*(10^-15)              //Load capacitane in farads
   (F)

//Outputs
delta_V=-Vdd*(C_internal/(C_internal+Cl))
                    //Total output voltage drop in volts
   (V)

//Results
mprintf("\nTotal output voltage drop delta_V: %.2f
   volts",delta_V);

//Outputs
//Total output voltage drop delta_V: 1.30 volts
```

Scilab code Exa 4.18 Optimizing Switching Activity at the Logic Level

```
//Example 4.18, Page Number 251
//Optimizing Switching Activity at the Logic Level
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
P_A=0.5               //Transition probability of input
     signal '(A=1)'
P_B=0.2               //Transition probability of input
     signal '(B=1)'
P_C=0.1               //Transition probability of input
     signal '(C=1)'

//Outputs
//As both the circuits implememt identical logic
    functionality , the output node 'Z' is equal in
    both the cases.So, difference in activity occurs
    at intermediate nodes:
Activity_first=(1-(P_A*P_B))*(P_A*P_B)         //
    Switching activity of first circuit
Activity_second=(1-(P_B*P_C))*(P_B*P_C)         //
    Switching activity of second circuit

//Results
mprintf("\nSwitching activity of first circuit : %.2
    f",Activity_first);
mprintf("\nSwitching activity of second circuit : %
    .4f",Activity_second)

//From the results we get to learn that it is
    benificial to postpone the introduction of
```

```
   signals with higher transition rate(i.e signals
   with signal probability close to )

//Outputs
//Switching activity of first circuit : 0.09
//Switching activity of second circuit : 0.0196
```

Chapter 5

Very High Performance Digital Circuits

Scilab code Exa 5.3 `Differential ECL Gate`

```
//Example 5.3, Page Number 277
//Differential ECL Gate
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vcc=5                         //Supply voltage in volts(
   V)
Iee=400*(10^-6)                    //Emitter current
   supply in amperes(A)
Rc=625                             //Collector resistance
    in ohms(  )

//Outputs
Voltage_swing=Iee*Rc                               //Output
   voltage swing in volts(V)
P_consm=Iee*Vcc                                    //Static
   power consumption in watts(W)
```

```

//Results
mprintf("\nOutput logic swing: %.2f volts",
   Voltage_swing);
mprintf("\nStatic power consumption: %.4f watts",
   P_consm);

//Outputs
//Output logic swing: 0.25 volts
//Static power consumption: 0.0020 watts (or) 2mW
```

Scilab code Exa 5.4 CML Gate Characteristics

```
//Example 5.4, Page Number 279
//CML Gate Characteristics
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vcc=0                          //Supply voltage in volts(
   V)
Vee=-1.7                            //Supply voltage in
    volts(V)
phi_t=26*(10^-3)        //Thermal voltage in volts(A)
Vref=-0.05                          //Reference voltage in
    volts(V)
Iee=0.4*(10^-3)                       //Emitter  current
   supply in amperes(A)
Rc=625                                //Collector resistance
     in ohms(  )
V_ih=-0.085                           //Input high level
    transition voltage using unity gain definition in
     volts(V)
V_il=-0.165                           //Input low level
```

```
   transition voltage using unity gain definition in
    volts(V)
Vbe_on=0                  //Base-Emitter voltage in
   volts(V)
//Vbe_on is zeo as Ic=Is (Vbe_on= t *log(Ic/Is)) i.e
    (log(1)=0)
C_fanout=60*(10^-15)          //Fanout capacitance
   in farads(F)
C_cs1=47*(10^-15)      //Collector-Source junction
   capacitance in farads(F)
C_bc1=22*(10^-15)      //Base-collector junction
   capacitance in farads(F)
Cc=((0.67*C_cs1)+(2*1.01*C_bc1))       //Collector
   capacitance in farads(F)
//Cbc1 is accounted twice to incoprate miller effect


//Outputs
Vswing=Iee*Rc                      //Output voltage
   swing in volts(V)
V_oh=Vcc-Vbe_on           //Output high level
   transition voltage in volts(V)
V_ol=Vcc-Vbe_on-(Vswing)    //Output low level
   transition voltage in volts(V)
NM_H=V_oh-V_ih                //Noise margin of high
   level transition in volts(V)
NM_L=V_il-V_ol                //Noise margin of low
   level transition in volts(V)
Cl=C_fanout+Cc                //Output capacitance in
    farads(F)
tp=0.69*Rc*Cl                  //Propogation delay in
    seconds(secs)
Pstat=(abs(Vee)*Iee)            //Static output power
    in watts(W)
Pdyn=Cl*(Vcc-Vee)*(Vswing/tp)    //Dynamic output
   power in watts(W)

//Results
```

```
mprintf("\nOutput voltage swing : %.2f volts",Vswing
   );
mprintf("\nVbe_oh: %.2f volts",V_oh);
mprintf("\nVbe_ol: %.2f volts",V_ol);
mprintf("\nVbe_ih: %.2f volts",V_ih);
mprintf("\nVbe_il: %.2f volts",V_il);
mprintf("\nNoise margin of high level transition: %
   .2f volts",NM_H);
mprintf("\nNoise margin of low level transition: %.2
   f volts",NM_L);
mprintf("\nOutput capacitance: %.16f farads",Cl);
mprintf("\nPropogation delay: %.13f seconds",tp);
mprintf("\nStatic output power: %.4f watts",Pstat);
mprintf("\nDynamic output power: %.4f watts",Pdyn);

//Outputs
//Output voltage swing : 0.25 volts
//Vbe_oh: 0.00 volts
//Vbe_ol: -0.25 volts
//Vbe_ih: -0.09 volts
//Vbe_il: -0.17 volts
//Noise margin of high level transition: 0.09 volts
//Noise margin of low level transition: 0.08 volts
//Output capacitance: 0.0000000000001359 farads (or)
    135.9 fF
//Propogation delay: 0.0000000000586 seconds (or)
   58.6 picosecs
//Static output power: 0.0007 watts (or) 0.7 mW
//Dynamic output power: 0.0010 watts (or) 1.0 mW
```

Scilab code Exa 5.5 VTC of an NTL Gate

```
//Example 5.5, Page Number 285
//VTC of an NTL Gate
//Scilab 6.0.1
```

```
//Windows 10
clc;

//Inputs
Vcc=1.9         //Supply voltage in volts(V)
Rc=2        //Collector resistance in ohms(   )
Re=0.5*(10^3)        //Emitter resistance in ohms(   )
Vbe_on=0.75            //Base−Emitter forward bias
   voltage in volts(V)

//Outputs
//A large value of ratio (−Rc/Re) enchances the
   voltage gain and noise margin,but increases gate
   delay.So,(Rc=2*Re) is a good compromise.Under
   these conditions V_ol evaluates as
V_ol=3*Vbe_on-Vcc           //Output low level
   transition voltage in volts(V)
V_oh=Vcc-Vbe_on             ////Output high level
   transition voltage in volts(V)
Signal_swing=2*Vcc-4*Vbe_on                //Signal
   swing in volts(V)

//Results
mprintf("\nVbe_oh: %.2f volts",V_oh);
mprintf("\nVbe_ol: %.2f volts",V_ol);
mprintf("\nSignal swing : %.2f volts",Signal_swing);

//Outputs
//Vbe_oh: 1.15 volts
//Vbe_ol: 0.35 volts
//Signal swing : 0.80 volts
```

Scilab code Exa 5.10 GaAs MESFET Current Voltage Characteristics

```
//Example 5.10, Page Number 305
```

```
//GaAs MESFET Current−Voltage Characteristics
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vgs=0.5                          //Gate−Source voltage in
    volts(V)
Vds=2                            //Drain−Source voltage in
   volts(V)
beta_enhancement=250*(10^-6)                     //
   =250∗(10ˆ−6) for Enhancement MESFET in Area per
   Voltage aquare(A/V∗∗2)
lamda=0.2                                //Channel
   modulation constant in volt inverse(1/V)
alpha=6.5                                 //Transistor
   parameter in inverese volt(V) and alpha=6.5 for
   enhancement MOSFET.
Vp=0.23                              //Pinch−Off voltage
   in volts(V)
W=4*(10^-6)                                  //Depletion
   region width
L=1*(10^-6)                                  //Depletion
   region length
delta_W=0.15*(10^-6)                              //
   Change in Depletion region width
delta_L=0.4*(10^-6)                               //Change
    in Depletion region length
Weff=W-delta_W                                     //
   Effective depletion region width
Leff=L-delta_L                                     //
   Effective depletion region length

//Outputs
Id=(Weff/Leff)*beta_enhancement*((Vgs-Vp)^2)*(1+(
   lamda*Vds))*tanh(alpha*Vds)    //Drain current in
   amperes(A)

```

```
//Results
mprintf("\nDrain current Id: %.7f amperes",Id);

//Output
//Drain current Id: 0.0001637 amperes (or) 163.7  A
```

Scilab code Exa 5.13 MESFET Source Follower

```
//Example 5.13, Page Number 310
//MESFET Source-Follower
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vd_on=0.7                    //Forward bias diode
   voltage in volts(V)
V_oh=-1.3                   //Output high level
   transition voltage in volts(V)
V_ol=-1.7                   //Output low level
   transition voltage in volts(V)
Vgs=0.5                     //Gate-Source voltage in
    volts(V)
Vds=0.8                     //Drain-Source voltage
   in volts(V)
beta_enhancement=250*(10^-6)               //
   =250*(10^-6) for Enhancement MESFET in Area per
   Voltage aquare(A/V**2)
lamda=0.2                             //Channel
   modulation constant in volt inverse(1/V)
alpha=6.5                              //
Vout=-1.5                          //Output voltage at
   midpoint of voltage swing in volts(V)
Vp=0.23                           //Pinch-Off voltage
   in volts(V)
```

```
Weff_to_Leff=10                              //Width to
   length ratio of depletion region

//Outputs
Isf=(Weff_to_Leff)*beta_enhancement*((Vgs-Vp)^2)
   *(1+(lamda*Vds))*tanh(alpha*Vds)    //Drain Source
    current in amperes(A)

//Results
mprintf("\nDrain current Isf: %.5f amperes",Isf);
//For this value of current,the voltage drop over
   the source follower is virtually constant over
   complete range of voltage interest.

//Output
//Drain current Isf: 0.00021 amperes (or) 0.21 mA
```

Chapter 6

Designing Sequential Logic Circuits

Scilab code Exa 6.3 `ECL SR Flip Flop`

```
//Example 6.3, Page Number 346
//ECL SR Flip-Flop
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vbe_on=0.7                    //Forward Base-Emitter
   voltage on volts(V)
Vbe_sat=0.8                   //Saturated Base-
   Emitter voltage on volts(V)
Vce_sat=0.1                   //Saturated Collector-
   Emitter voltage on volts(V)
beta_f=100                        //Forward current
   gain
Vcc=0                           //Supply voltage in
   volts(V)
Vee=-3.5                       //Emmiter voltage in
   volts(V)
```

```
//Collector current is set to 0.5mA
Ic1=0.5*(10^-3)                    //Collector current in amperes(A)

//Outputs
Vb1=Vcc-Vbe_on                 //Base voltage of Q1 in volts(V)
Ve1=Vb1-Vbe_on                 //Emitter voltage of Q1 in volts(V)
Re=(Ve1-Vee)/Ic1               //Emitter resistance in ohms(  )
//Solving the below equations gives relation between Re and Rc:
//Vc1=Vcc-(Ic1*Rc)
//Vq_compliment=Vc1-Vbe_on
//Vq_compliment=Vcc-(Rc/Re)*(Vcc-(2*Vbe_on)-Vee)-Vbe_on
Rc=Re/4.2                      //Collector resistance in ohms(  )
//The absolute values of Rc,Re and Rb are determined by desired current levels and the required transient response time

//Results
mprintf("\nEmitter resistance Re: %.1f ohms(  )",Re);
mprintf("\nCollector resistance Re: %.1f ohms(  )",Rc);

//Outputs
//Emitter resistance Re: 4200.0 ohms(  ) (or) 4.2 k
//Collector resistance Re: 1000.0 ohms(  ) (or) 1 k
```

Scilab code Exa 6.5 Emitter Coupled Schmitt Trigger

```
//Example 6.5, Page Number 366
//Emitter-Coupled Schmitt Trigger
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vbe_on=0.7                    //Forward Base-Emitter voltage on volts(V)
Vbe_sat=0.8                   //Saturated Base-Emitter voltage on volts(V)
Vce_sat=0.1                   //Saturated Collector-Emitter voltage on volts(V)
beta_f=100                        //Forward current gain
Vcc=5             //Supply voltage in volts(V)
R1=4*(10^3)           //Collecor resitance of transistor Q1 in ohms( )
R2=2.5*(10^3)           //Collecor resitance of transistor Q2 in ohms( )
Re=1*(10^3)           //Emitter resitance in ohms( )


//Outputs
V_oh=Vcc                        //Output high level transition voltage in volts(V)
R1_parallel_R2= (R1*R2)/(R1+R2)
Ve_M_plus=Vcc*Re/(Re+(R1_parallel_R2))       //Emitter voltage during low to high transition in volts(V)
V_ol=Ve_M_plus+Vce_sat                       //Output low level transition voltage in volts(V)
V_M_plus=Ve_M_plus+Vbe_on                 //Switching threshold during low to high transition in volts(V)
Ve_M_minus=((Vcc*Re)/(R1+Re))      //Emitter voltage during high to low transition in volts(V)
```

```
V_M_minus=Ve_M_minus+Vbe_on                //
   Switching threshold during high to low transition
    in volts(V)


//Results
mprintf("\nOutput high level transition voltage V_oh
   : %.4f volts",V_oh);
mprintf("\nOutput low level transition voltage V_ol:
    %.4f volts",V_ol);
mprintf("\nSwitching threshold during low to high
   transition V_M_plus: %.4f volts",V_M_plus);
mprintf("\nSwitching threshold during high to low
   transition V_M_minus: %.2f volts",V_M_minus);

//Outputs
//Output high level transition voltage V_oh: 5.0000
   volts
//Output low level transition voltage V_ol: 2.0697
   volts
//Switching threshold during low to high transition
   V_M_plus: 2.6697 volts
//Switching threshold during high to low transition
   V_M_minus: 1.70 volts
```

Scilab code Exa 6.7 Current Starved Inverter

```
//Example 6.7, Page Number 372
//Current-Starved Inverter
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vgs=5                        //Gate-Source voltage in
```

```
   volts(V)
beta_depletion=19.6*(10^-6)                //
   =19.6*(10^-6) for Depletion FET in Area per
   Voltage square(A/V**2)
lamda=0.2                              //Channel
   modulation constant in volt inverse(1/V)
Vp=0.74                             //Pinch-Off voltage
   in volts(V)
Weff1=1.8                                    //Effective
    depletion region width
Leff=0.9                                     //Effective
   depletion region length
Weff2=9                                      //Effective
   depletion region width

//Outputs
//Maximum current of minimum-size inverter is given
   by below relation:
Isat=0.5*(Weff1/Leff)*beta_depletion*((Vgs-Vp)^2)
   //Saturation current in amperes(A)
//Maximum current of minimum-size control transistor
    M3 is given by below relation:
Vcontr=sqrt((Isat/(0.5*(Weff2/Leff)*beta_depletion))
   )-Vp      //Control voltage of transistor M3 in
   volts(V)

//Results
mprintf("\nSaturation current: %.7f amperes",Isat);
mprintf("\ncontrol voltage: %.2f volts",Vcontr);

//Outputs
//Saturation current: 0.0003556929600 amperes (or)
   355.7 A
//control voltage: 1.17 volts
```

Chapter 7

Designing Arithmetic Building Blocks

Scilab code Exa 7.2 Static Adder Design

```
//Example 7.2, Page Number 391
//Static Adder Design
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
tab_c0=1.63*(10^-9)                //Delay to generate
    carry output for inputs in seconds(secs)
tci_c0=0.32*(10^-9)                //Delay to generate
    carry output for input carry in seconds(secs)
tci_s=1*(10^-9)                //Delay to generate sum
    output for input carry in seconds(secs)
N=32                                //Number of bit
   adder

//Outputs
tadd=tab_c0+(N-2)*tci_c0+tci_s               //Adder
   delay in seconds(secs)
```

```

//Results
mprintf("\nAdder delay tadd: %.11f seconds",tadd);

//Output
//Adder delay tadd : 0.00000001223 seconds (or)
   12.23 nsecs
```

Scilab code Exa 7.4 `Transistor Sizing in the Manchester Carry Chain`

```
//Example 7.4, Page Number 394
//Transistor Sizing in the Manchester Carry Chain
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
//The worst case delay of the carry chain adder can
   be modelled by linear RC network.
R=20*(10^3)                            //Linearized on-
   resistance of RC network in ohms(  )
C=20*(10^-15)                          //Linearized
   diffusion capacitance of RC network in ohms(  )

//Outputs
//Propogation delay of RC network:tp=(0.69*( Ci )*(
     *Rj))
//Analysing equivalent network to determine
   propogation delay of carry chain
tp=0.69*C*(6*R+5*R+4*R+3*R+2*R+1*R)        //Worst
   case propogation delay of adder circuit in
   seconds(secs)

//Results
mprintf("\nWorst case propogation delay of adder
   circuit tp: %.10f seconds",tp);

//Output
//Worst case propogation delay of adder circuit tp:
   0.0000000058 seconds (or) 5.8 nsecs
```

Chapter 8

Coping With Interconnect

Scilab code Exa 8.2 Capacitance of Metal Wire

```
//Example 8.2, Page Number 444
//Capacitance of Metal Wire
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
L=10*(10^-2)                       //Length of wire in
   metres(m)
W=4*(10^-6)                         //Width of wire in
   metres(m)
C_layer=0.031*((10^-15)/(10^-12))
                       //Interconnect layer
   capacitance per unit area between metal1 and
   substrate in farads per metre square(F/m^2)
C_fring=0.044*((10^-15)/(10^-6))             //
   Fringing capacitance between metal1 and substrate
    in farads per metre(F/m)

//Outputs
C_area=L*W*C_layer            //Total area
```

```
    capacitance in farads(F)
C_fringing=L*2*C_fring              //Total fringing
    capacitance in farads(F)
//Factor '2' in C_fringing is to consider two sides
    of the wire.
C_total=C_area+C_fringing           //Total
    capacitance in farads(F)

//Results
mprintf("\nTotal area capacitance C_area: %.13f
    farads",C_area);
mprintf("\nTotal fringing capacitance C_fringing: %
    .13f farads",C_fringing);
mprintf("\nTotal capacitance C_total: %.13f farads",
    C_total);

//Outputs
//Total area capacitance C_area: 0.0000000000124
    farads (or) 12.4 pF
//Total fringing capacitance C_fringing:
    0.0000000000088 farads (or) 8.8 pF
//Total capacitance C_total: 0.0000000000212 farads
    (or) 21.2 pF
```

Scilab code Exa 8.3 Interwire Capacitance and Cross Talk

```
//Example 8.3, Page Number 446
//Interwire Capacitance and Cross Talk
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
V=5                          //Voltage transition on Y
    in volts(V)
```

```
//As computation of fringing effect in this case of
   overlapping wires is complex, we assume two sides
   of wire contribute and a voltage disturbance is
   combined with parasitic effects
Cx=25*(10^-15)                         //Capacitance
   at node 'x' in farads(F)
//Parasitic capacitance is given as (5*5*0.055fF
   +2*5*0.049fF)
Cxy=1.9*(10^-15)                        //Parasitic
   capacitance in farads(F)

//Outputs
delta_Vx=(Cx/(Cx+Cxy))*V               //Voltage drop
   on dynamic node in volts(V)
V_disturbance=V-delta_Vx                //Voltage
   disturbance at dynamic node in volts(V)

//Results
mprintf("\nVoltage drop at dynamic node delta_Vx: %
   .1f volts",delta_Vx);
mprintf("\nVoltage disturbance at dynamic node
   V_disturbance: %.1f volts",V_disturbance);

//Output
//Voltage drop at dynamic node delta_Vx: 4.6 volts
//Voltage disturbance at dynamic node V_disturbance:
   0.4 volts
```

Scilab code Exa 8.4 `Output Buffer Design`

```
//Example 8.4, Page Number 453
//Output Buffer Design
//Scilab 6.0.1
//Windows 10
clc;
```

```

//Inputs
Cl=20*(10^-12)                    //Off−chip capacitance
    in farads(F)
Ci=10*(10^-15)                    //Input capacitance in
    farads(F)
tpo=0.2*(10^-9)                  //Propogation delay of
    minimum size gate with single fanout in seconds(
    secs)
mu=2.96                  //Scaling factor

//Outputs
tp=mu*log(Cl/Ci)*tpo                    //Total
    propogation delay in seconds(secs)

//Results
mprintf("\nTotal propogation delay tp: %.10f seconds
    ",tp);

//Output
//Total propogation delay tp: 0.0000000045 seconds (
    or) 4.5 nsecs
```

Scilab code Exa 8.6 Cascode Charge Redistribution Amplifier

```
//Example 8.6, Page Number 460
//Cascode Charge−Redistribution Amplifier
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vcc=5                    //Gate source voltage in
    volts(V)
Vtn=0.74                          //Threshold voltage
```

```
in volts(V)
Vref=5                  //Reference voltage in volts(V)
Weff=1.8*(10^-6)                    //Effective width of
L=1.2*(10^-6)                    //Effective length of
delta_L=0.3*(10^-6)                           //Difference in effective length in metres(m)
Leff=L-delta_L                             //Effective length
kn=5.4*(10^-6)                              //Device parameter in Area per Voltage aquare(A/V**2)
Vt=0.74          //Threshold voltage in volts(V)

//Outputs
Vout_high=Vcc                          //High voltage level of output node in volts(V)
Vbus_high=Vref-Vtn*(Vref)             //High voltage level of output node in volts(V)
I_M3=(1/2)*(Weff/Leff)*kn*((Vcc-Vt)^2)    //Drain current in transistor M3 amperes(A)
//The low voltages has to be solved iteratively due to body-effect factor.Equating I_M3 with I_M1 yiilds Vbus_low:
Vbus_low=0.63                  //Low voltage level of output node in volts(V)
//The resulting voltage drop across M2(0.1volts) yields Vout_low:
Vout_low=0.63+0.1                  //Low voltage level of output node in volts(V)

//Results
mprintf("\nHigh voltage level of output node Vout_high: %.1f volts",Vout_high);
mprintf("\nHigh voltage level of bus node Vbus_high: %.1f volts",Vbus_high);
mprintf("\nDrain current in transistor M3 I_M3: %.8f amperes",I_M3);
mprintf("\nLow voltage level of bus node Vbus_low: %
   .2f volts",Vbus_low);
mprintf("\nLow voltage level of output node Vout_low
   : %.2f volts",Vout_low);

//Outputs
//High voltage level of output node Vout_high: 5.0
   volts
//High voltage level of bus node Vbus_high: 1.3
   volts
//Drain current in transistor M3 I_M3: 0.00009800
   amperes (or) 98 A
//Low voltage level of bus node Vbus_low: 0.63 volts
//Low voltage level of output node Vout_low: 0.73
   volts
```

Scilab code Exa 8.9 `RC Delay of Polysilicon Wire`

```
//Example 8.9, Page Number 473
//RC Delay of Polysilicon Wire
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
L=1*(10^-3)                       //Length of wire in
   metres(m)
//Resistance is obtained by:R=(R_sheet*Length of
   wire)/(Width of wire)
R=10*((1000)^2)                           //Resistance of
    RC network in ohms(   )
C_layer=0.058*(10^-15)                         //
   Interconnect layer capacitance per unit area
   between metal1 and substrate in farads per metre
   square(F/m^2)
C_fring=0.043*(10^-15)             //Fringing
```

```
    capacitance between metal1 and substrate in
    farads per metre(F/m)

//Outputs
C=(C_layer+(2*C_fring))                       //Total
     capacitance of wire in farads(F)
tp=0.38*R*C                         //Propogation delay of
     distributed RC network in seconds(secs)

//Results
mprintf("\nPropogation delay of distributed RC
    network tp: %.11f seconds",tp);

//Output
//Propogation delay of distributed RC network tp:
    0.00000000055 seconds (or) 0.54 nsecs
```

Scilab code Exa 8.10 Optimized Delay of RC Chain

```
//Example 8.10, Page Number 474
//Optimized Delay of RC Chain
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
L=5*(10^-3)                          //Length of wire in
    metres(m)
tp_buf=0.5*(10^-9)                          //Propogation
    delay of buffer in seconds(secs)
//Resistance is obtained from sheet resistance of
    polysilicon with tolerance.
R=95*((10000)^2)                            //Resistance of
    RC network in ohms(   )
//Capacitance ontained from summing fringing and
```

```
   interconnect capacitances for polysilicon wire:
C=0.15*(10^-12)                    //Capacitance of
   network in farads(F)

//Outputs
M=L*sqrt((0.38*R*C)/tp_buf)      //Number of buffer
   sections
tp=0.38*R*C*(((L/M)^2)*M)+(M-1)*tp_buf          //
   Propogation delay of distributed RC network in
   seconds(secs)

//Results
mprintf("\nNumber of buffer sections M: %.4f",M);
mprintf("\nPropogation delay of distributed RC
   network tp: %.12f seconds",tp);

//Output
//Number of buffer sections M: 5.2034
//Propogation delay of distributed RC network tp:
   0.000000004703 seconds (or) 4.703 nsecs
```

Scilab code Exa 8.13 Noise Induced by Inductive Bonding Wires and Package Pins

```
//Example 8.13, Page Number 478
//Noise Induced by Inductive Bonding Wires and
   Package Pins
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Cl=20*(10^-12)                          //Driving load
    capacitance in farads(F)
V_swing=5                               //Voltage swing
    in volts(V)
```

```
tf=0.9                                        //Final output
     reach in seconds(secs)
ti=0.1                                        //Initial
    output reach in seconds(secs)
tr=4*(10^-9)                                   //Rise time
    in seconds(secs)
L=10*(10^-9)                                   //Series
    inductance in henry(H)
dt=2*(10^-9)                                    //Output
    swing time in seconds(secs)
Ip=40*(10^-3)                                    //Peak
    current in amperes(A)

//Outputs
Iavg=(Cl*(tf-ti)*V_swing)/tr                          //
    Average curremt to drive output in amperes(A)
Vl=L*(Ip/dt)                                        //
    Voltage drop across inductor in volts(V)

//Results
mprintf("\nPropogation delay of distributed RC
    network: %.4f amperes",Iavg);
mprintf("\nVoltage drop across inductor: %.1f volts"
    ,Vl);

//Outputs
//Propogation delay of distributed RC network:
    0.0200 amperes (or) 20 mA
//Voltage drop across inductor: 0.2 volts
```

Scilab code Exa 8.16 Capacitive Termination

```
//Example 8.16, Page Number 489
//Capacitive Termination
//Scilab 6.0.1
```

```
//Windows 10
clc;

//Inputs
Z0=50                                  //Characteristic
    impedance of the line in ohms(  )
Cl=2*(10^-12)                  //Capacitive load in
   farads(F)
//Outputs
//The output rises to its final value with time
   constant:
tou=Z0*Cl                   //Time constant in seconds(
   secs)
t_charge=0.69*Z0*Cl    //Time to charge the
   capacitance in seconds(secs)

//Results
mprintf("\nTime constant    : %.13f seconds",tou);
mprintf("\nTime to charge the capacitance: %.13f
   seconds",t_charge);

//Outputs
//Time constant tou: 0.0000000001000 seconds (or)
   100.0 psecs
//Time to charge the capacitance: 0.0000000000690
   seconds (or) 69.0 psecs
```

Scilab code Exa 8.18 Thermal Bounds on Integration

```
//Example 8.18, Page Number 500
//Thermal Bounds on Integration
//Scilab 6.0.1
//Windows 10
clc;

```

```
//Inputs
delta_T=100                                  //Maximum
   temperature difference between chip and
   environment in degree celsius(C)
theta=2.5                //Thermal resistance between
    chip and environment in celsius per watt(C/W)
E=0.1*(10^-12)                              //Switching
    energy of each gate in joule(J)

//Outputs
Ng_to_tp=(delta_T/(theta*E))                       //
   Ratio of ()Ng=Number of gates on chip,tp=
   Propgation delay) in gates per seconds(Gate/secs)

//Results
mprintf("\nMaximum number of gates on chip when all
   gates are simultaneously: %.1f gates per second",
   Ng_to_tp);

//Output
//Maximum number of gates on chip when all gates are
    simultaneously: 400000000000000.0 or 4*(10^5)
   gates/nanoseconds
```

Chapter 9

Timing Issues In Digital Circuits

Scilab code Exa 9.5 Synchronizers and Mean Time to Failure

```
//Example 9.5, Page Number 537
//Synchronizers and Mean Time−to−Failure
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
T_phi=10*(10^-9)                                    //
   Sampling period in seconds(secs)
T1=10*(10^-9)                                       //
   Waiting time in seconds(secs)
T2=20*(10^-9)                                       //
   Waiting time in seconds(secs)
Tsignal=50*(10^-9)                                  //
   Time period of the signal in seconds(secs)
tr=1*(10^-9)                                        //Rise
   time in seconds(secs)
tou=310*(10^-12)                                    //
   Time constant in seconds(secs)
```

```
Vswing=5                                       //
   Voltage swing of signal in volts(V)
V_transition=1                            //Transition
   difference in volts(V){Vtransition=V_ih−V_il}

//Outputs
P_init=((V_transition/Vswing)/Tsignal)*tr  //Error
   probability in errors per second(Error/Sec)
N_sync1=(P_init*(exp(-T1/tou)))/T_phi               //
   Averege number of synchronization errors per
   second for T1 in inverse seconds(errors/secs)
Mean_time_failure1=1/N_sync1            //Mean time to
    failure in seconds(secs)
//If wating period(T) is doubled:
N_sync2=(P_init*(exp(-T2/tou)))/T_phi               //
   Averege number of synchronization errors per
   second for T2 in inverse seconds(errors/secs)
Mean_time_failure2=1/N_sync2            //Mean time to
    failure in seconds(secs)

//For a typical CMOS inverter with voltage swing of
   5V, the V_IH−V_IL computed is 1V
//Results
//in seconds
mprintf("\nAverege number of synchronization errors
   per second N_sync1: %.9f errors/sec",N_sync1);
mprintf("\nMean time to failure for waiting time T1
   Mean_time_failure1: %.1f seconds",
   Mean_time_failure1);
//If no synchronizer was used, the MTF would only
   have been 2.5  secs !

//in years
mprintf("\nMean time to failure for waiting time T1
   Mean_time_failure1(in years): %.2f years",
   Mean_time_failure1/(365*24*60*60));
mprintf("\nMean time to failure for waiting time T2
   Mean_time_failure2(in years): %.2f years",
   Mean_time_failure2/(365*24*60*60));

//Outputs
//Averege number of synchronization errors per
   second N_sync1: 0.000000004 errors/sec
//Mean time to failure for waiting time T1
   Mean_time_failure1: 255528546.7 seconds (or) 2.55
    X 10^8 secs
//Mean time to failure for waiting time T1
   Mean_time_failure1(in years): 8.10 years
//Mean time to failure for waiting time T2
   Mean_time_failure2(in years): 828194294345529.87
   years (or) 8.2 X 10^14 years
```

Chapter 10

Designing Memory And Array Structures

Scilab code Exa 10.3 `Voltage Swings in NOR and NAND ROMs`

```
//Example 10.3, Page Number 565
//Voltage Swings in NOR and NAND ROM's
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
Vdd=5                        //Supply voltage in volts(
    V)
Vto_PMOS=-0.75                    //Threshold voltage in
     volts(V)
//Vto_NMOS is assumed as abs(Vto_PMOS)
W_L_NMOS=9                           //(W/L) ratio of
    NMOS FET after considering lateral diffusion
    differences
V_ol=2.5          //Transition logic low level voltage
     in volts(V)

//Outputs
```

```
W_L_PMOS_NOR=(1-sqrt(1-(V_ol/(Vdd-Vto_PMOS))))*3*
   W_L_NMOS

//Due to series chaining , the value of Vol is a
   function of both size of memory and programming.
   For this series chaining a longer device has to
   be used for N transistors which unacceptable
   results for larger arrays.So,NAND ROM's are
   rarely used for arrays with more than 8 rows.
//(8X8):(W/L)p=0.62
//(512X512):(W/L)p=0.0097

//Results
mprintf("\n(W/L) ratio of PMOS FET: %.2f seconds",
   W_L_PMOS_NOR);

//Output
//(W/L) ratio of PMOS FET: 6.70 seconds
```

Scilab code Exa 10.4 Word and Bit Line Parasitics

```
//Example 10.4, Page Number 566
//Word and Bit Line Parasitics
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
C_layer_poly=0.058*(10^-15)                //
   Interconncet layer capacitance between
   polysilicon to substrate per unit area in farads
   per cell(F/m^2)
C_layer_metal=0.031*(10^-15)               //
   Interconncet layer capacitance between metal to
   substrate per unit area in farads per cell(F)
```

```
C_fring=0.043*(10^-15)                    //Fringing capacitance per cell in farads(F)
R_sheet_metal=0.07                                  //Sheet resistance per unit area in ohm per area( /  )
R_sheet_diffusion=10                                 //Sheet resistance per unit area in ohm per area( /  )
lamda=1                                     //Channel modulation in volt inverse(1/V)

//Outputs
//NOR
//Word-Line Parasictics
Resistance_cell_word_NOR=(7/2)*R_sheet_diffusion                          //Resistance per unit memory cell in ohms(  )
wire_capacitance_cell_word_NOR=(((7*lamda)*(2*lamda))*((0.6)^2)*C_layer_poly)+2*((7*lamda)*0.6)*C_fring   //Wire capacitance of unit cell in farads(F)
Gate_capacitance_cell_NOR=((4*lamda)*(2*lamda))*((0.6)^2)*1.76*(10^-15)           //Gate capacitance of ell in farads(F)
//Bit-Line Parasitcics
Resistance_cell_bit_NOR=(8.5/4)*R_sheet_metal                          //Resistance per unit memory cell in ohms(  )
wire_capacitance_cell_bit_NOR=((8.5*lamda)*(4*lamda))*((0.6)^2)*C_layer_metal+2*((8.5*lamda)*0.6)*C_fring   //Wire capacitance of unit cell in farads(F)
drain_capacitance_cell_NOR=((((3*lamda)*(4*lamda))*((0.6)^2)*(0.3))+(2*(3*lamda)*0.6*0.8))*(0.375*(10^-15))+((4*lamda)*0.6*(0.43*(10^-15)))
//NAND
//Word-Line Parasictics
Resistance_cell_word_NAND=(6/2)*R_sheet_diffusion
```

```
                    //Resistance per unit memory
   cell in ohms(  )
wire_capacitance_cell_word_NAND=(((6*lamda)*(2*lamda
   ))*((0.6)^2)*C_layer_poly)+2*((6*lamda)*0.6)*
   C_fring   //Wire capacitance of unit cell in
   farads(F)
Gate_capacitance_cell_NAND=((3*lamda)*(2*lamda))
   *((0.6)^2)*(1.76*(10^-15))          //Gate
   capacitance of ell in farads(F)
//Bit-Line Parasitcics
Resistance_cell_bit_NAND=10*(10^3)
                    //Resistance per unit memory
   cell in ohms(  )
//wire_capacitance_cell_bit:Included in diffusion
   capacitance in farads(F)
source_drain_capacitance_cell_NAND=(((3*lamda)*(3*
   lamda))*((0.6)^2)*0.3*(10^-15))+(2*3*lamda
   *0.6*0.8*0.345*(10^-15))+(3*lamda*2*lamda*((0.6)
   ^2)*1.7*(10^-15))



//Results
mprintf("\nResistance per unit NOR memory cell
   Resistance_cell_word_NOR: %.1f ohms",
   Resistance_cell_word_NOR);
mprintf("\nWire capacitance of unit NOR cell
   wire_capacitance_cell_word_NOR: %.18f farads",
   wire_capacitance_cell_word_NOR);
mprintf("\nGate capacitance of NOR cell
   Gate_capacitance_cell_NOR: %.18f farads",
   Gate_capacitance_cell_NOR);
mprintf("\nResistance per unit NOR memory cell
   Resistance_cell_bit_NOR: %.2f ohms",
   Resistance_cell_bit_NOR);
mprintf("\nWire capacitance of unit NOR cell
   wire_capacitance_cell_bit_NOR: %.18f farads",
   wire_capacitance_cell_bit_NOR);
mprintf("\nGate capacitance of NOR cell
   drain_capacitance_cell_NOR: %.18f farads",
   drain_capacitance_cell_NOR);
mprintf("\nResistance per unit NAND memory cell
   Resistance_cell_word_NAND: %.1f ohms",
   Resistance_cell_word_NAND);
mprintf("\nWire capacitance of unit NAND cell
   wire_capacitance_cell_word_NAND: %.18f farads",
   wire_capacitance_cell_word_NAND);
mprintf("\nGate capacitance of NAND cell
   Gate_capacitance_cell_NAND: %.18f farads",
   Gate_capacitance_cell_NAND);
mprintf("\nResistance per unit NAND memory cell
   Resistance_cell_bit_NAND: %.1f ohms",
   Resistance_cell_bit_NAND);
mprintf("\nGate capacitance of NAND cell
   source_drain_capacitance_cell_NAND: %.16f farads"
   ,source_drain_capacitance_cell_NAND);


//Outputs
//Resistance per unit NOR memory cell
   Resistance_cell_word_NOR: 35.0 ohms
//Wire capacitance of unit NOR cell
   wire_capacitance_cell_word_NOR:
   0.000000000000000654 farads (or) 0.654 fF
//Gate capacitance of NOR cell
   Gate_capacitance_cell_NOR: 0.000000000000005069
   farads (or) 5.069 fF
//Resistance per unit NOR memory cell
   Resistance_cell_bit_NOR: 0.15 ohms
//Wire capacitance of unit NOR cell
   wire_capacitance_cell_bit_NOR:
   0.000000000000000818 farads (or) 0.818 fF
//Gate capacitance of NOR cell
   drain_capacitance_cell_NOR: 0.000000000000002598
   farads (or) 2.598 fF

//Resistance per unit NAND memory cell
```

```
   Resistance_cell_word_NAND: 30.0 ohms
//Wire capacitance of unit NAND cell
   wire_capacitance_cell_word_NAND:
   0.00000000000000560 farads (or) 0.56 fF
//Gate capacitance of NAND cell
   Gate_capacitance_cell_NAND: 0.000000000000003802
   farads (or) 3.802 fF
//Resistance per unit NAND memory cell
   Resistance_cell_bit_NAND: 10000.0 ohms (or) 10
   k
//Gate capacitance of NAND cell
   source_drain_capacitance_cell_NAND:
   0.0000000000000056 farads (or) 5.6 fF
```

Scilab code Exa 10.5 Propagation Delay of NOR RAM

```
//Example 10.5, Page Number 569
//Propagation Delay of NOR RAM
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
M=512                      //Number of cells
C_gate_word=5.1*(10^-15)                    //Word Gate
   capacitance in farads(F)
C_wire_word=0.65*(10^-15)                   //Word Wire
    capacitance in farads(F)
C_word=C_gate_word+C_wire_word                     //
   Capacitance for generating word in farads(F)
R_word=35                              //Resistance in
    ohms(  )
C_gate_bit=2.6*(10^-15)                    //Word Gate
   capacitance in farads(F)
C_wire_bit=0.8*(10^-15)                    //Word Wire
```

```
   capacitance in farads(F)
Kp=5.3*(10^-6)                    //Device paramter in
   ampere per voltage square(A/V^2)
Kn=19.6*(10^-6)                   //Device paramter in
   ampere per voltage square(A/V^2)
//W_L_NMOS=(2.4/1.2)                     //(W/L) ratio of
   NMOS transitor
//W_L_PMOS=(6/1.2)                   //(W/L) ratio of
   PMOS transitor
W_Leff_NMOS=(2.4/0.9)                   //(W/Leff) ratio
   of NMOS transitor
W_Leff_PMOS=(6/0.9)                   //(W/Leff) ratio of
    PMOS transitor


//Outputs
//Word-Line Delay
t_word=0.38*(R_word*C_word)*((M)^2)                //
   Delay of didtributed RC line containg M cells in
   seconds(secs)
//Bit-Line Delay
C_bit=M*(C_gate_bit+C_wire_bit)               //
   Capacitance for generating a bit in farads(F)
Iav_HL=(1/2)*(W_Leff_NMOS)*(Kn)*((((4.25)^2)/2)
   +((4.25*3.75)-(((3.75)^2)/2)))-((1/2)*(
   W_Leff_PMOS)*(Kp)*((4.25*1.25)-(((1.25)^2)/2)))
           //Average high to low level transition
   current in amperes(A)
t_HL=(C_bit*1.25)/Iav_HL                      //High to
   low response time in seconds(secs)
//By similar above procedure Iav_LH can be computed
   and is given by:
Iav_LH=0.36*(10^-3)      //Average low to high level
   transition current in amperes(A)
t_LH=(C_bit*1.25)/Iav_LH


//Results
```

```
mprintf("\nDelay of didtributed RC line containg M
   cells t_word: %.10f seconds",t_word);
mprintf("\nCapacitance for generating a bit C_bit: %
   .13f farads",C_bit);
mprintf("\nAverage high to low level transition
   current Iav_HL: %.6f amperes",Iav_HL);
mprintf("\nHigh to low response time t_HL: %.10f
   seconds",t_HL);
mprintf("\nAverage low to high level transition
   current Iav_LH: %.6f amperes",Iav_LH);
mprintf("\nLow to high response time t_LH: %.10f
   seconds",t_LH);



//Outputs
//Delay of didtributed RC line containg M cells
   t_word: 0.0000000200 seconds (or) 20 nsecs
//Capacitance for generating a bit C_bit:
   0.0000000000017 farads (or) 1.7 picosecs
//Average high to low level transition current
   Iav_HL: 0.000389 amperes (or) 0.389 mA
//High to low response time t_HL: 0.0000000056
   seconds (or) 5.6 nsecs
//Average low to high level transition current
   Iav_LH: 0.000360 amperes (or) 0.360 mA
//Low to high response time t_LH: 0.0000000060
   seconds (or) 6.0 nsecs
```

Scilab code Exa 10.9 `IT DRAM Read out`

```
//Example 10.9, Page Number 588
//IT DRAM Read-out
//Scilab 6.0.1
//Windows 10
```

```
clc;

//Inputs
C_bit=1*(10^-12)                      //Bit line
   capacitance in farads(F)
V_pre=2.5                  //Bit precharge voltage in
   volts(V)
Cs=50*(10^-15)                           //Cell capacitance
    in farads(F)
V_logic_high=3.5             //Cell voltage when logic
    high in volts(V)
V_logic_low=0              //Cell voltage when logic
   low in volts(V)

//Outputs
delta_Vswing_logic_low=(V_logic_low-V_pre)*(Cs/(Cs+
   C_bit))      //Voltage swing in bit line when
   logic low in volts(V)
delta_Vswing_logic_high=(V_logic_high-V_pre)*(Cs/(Cs
   +C_bit))                 //Voltage swing in bit
   line when logic high in volts(V)

//Results
mprintf("\nVoltage swing in bit line when logic low
   delta_Vswing_logic_low: %.4f volts",
   delta_Vswing_logic_low);
mprintf("\nVoltage swing in bit line when logic high
    delta_Vswing_logic_high: %.3f volts",
   delta_Vswing_logic_high);


//Outputs
//Voltage swing in bit line when logic low
   delta_Vswing_logic_low: −0.1190 volts (or) −119.0
    mV
//Voltage swing in bit line when logic high
   delta_Vswing_logic_high: 0.048 volts (or) 48 mV

```

```
//NOTE:
//  V  (1)=60 mV according to the textbook , but above
     calculations using equation 10.5 on page no 588
    gives   V  (1)=48 mV
```

Scilab code Exa 10.11 Column Decoders

```
//Example 10.11, Page Number 596
//Column Decoders
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
//Consider a 1024:1 decoder
K=10                //Number of inputs of decoder
K_precoded=5                //Number of precoded bits
N_tree=2*((2^K_precoded)-1)              //Number of
    devices required in tree decoder
N_pass=(2^K)                    //Number of devices
    in pass deocoder
N_pre=6*(2^K_precoded)              //Number of devices
    for 5 bit precoding

//Outputs
N_dec=N_pre+N_pass+N_tree              //Number of
    devices in column decoder

//Results
mprintf("\nNumber of devices in column decoder N_dec
    : %.1f",N_dec);



//Outputs
```

```
//Number of devices in column decoder N_dec: 1278.0
```

Scilab code Exa 10.12 `Differential Sense Amplifier`

```
//Example 10.12, Page Number 599
//Differential Sense Amplifier
//Scilab 6.0.1
//Windows 10
clc;

//Inputs
W_L_NMOS=10                    //(W/L) ratio of NMOS
    transitor
W_L_PMOS=10                    //(W/L) ratio of PMOS
    transitor
I_bias=100*(10^-6)             //biasing current in
    amperes(A)
lamda_NMOS=0.06                      //Channel modulation
    in volt inverse(1/V)
lamda_PMOS=0.19                      //Channel modulation
    in volt inverse(1/V)
Kn=19.6*(10^-6)                //NMOS device parameter
    in ampere per volt square(A/V^2)

//Outputs
A=(1/(lamda_NMOS+lamda_PMOS))*sqrt((4*Kn*W_L_NMOS)/
    I_bias)    //Gain of differential sense amplifier

//Results
mprintf("\nGain of differential sense amplifier A: %
    .2f",A);

//Output
//Gain of differential sense amplifier A: 11.20
```