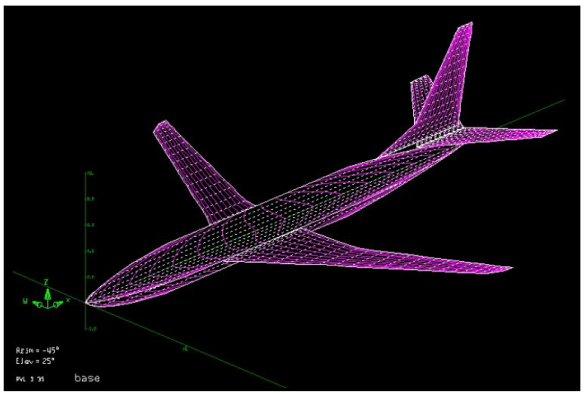
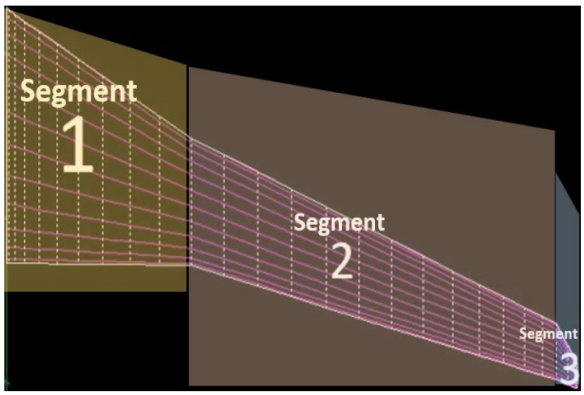
Circuits design in aerospace systems





Index;

Algorithms design for large circuits

- *(Cost reduction)*
- *(Synthesis tools*)

Dynamical systems in Spaceships

- *Electronic diagrams*
- *EDA-TOOLS(VHDL-DESIGN)*
- *Bare metal scripting*

Second part()

- *AVR microcontrollers*
- *Bare metal notes*
- *RODOS*

Third part()

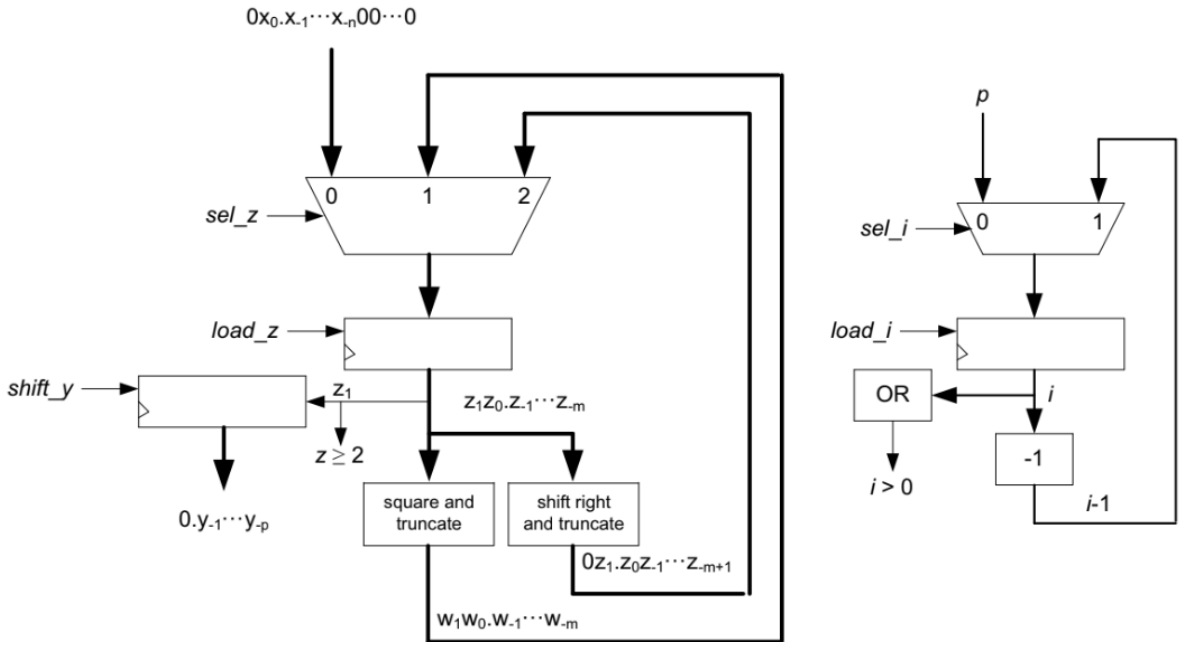
- *Structural design for aerospace projects;*
- *Github-aerospace designs*
- *Aircraft-design*
- *Rockets-models with python*

Algorithms-descriptive-approach

-The main topics are circuit segmentation, combinational circuit to pipelined circuit transformation and interconnection of pipelined circuits and self-timed circuits.
The optimal implementation of loops is a basic aspect of the synthesis of digital circuits.Combinational and sequential implementations in data path connectivity (buses), first-in first-out (FIFO) files, register files, arithmetic and logic unit (ALU), hierarchical description and sequential implementation (lower cost and longer time)

Automation: modern Electronic Design Automation tools have the capacity to directly generate circuits from algorithmic descriptions, with performances—latency, cost, consumption.

Registers: registers that store the algorithm variables,
• computation resources that execute operations,
• connections that transfer data between registers and computation resources.



Control unit is generally modeled by a finite-state machine, implements programs executed by the processor and it generates control signals(load, reset, shift left) control inputs of arithmetic operations (multiply,sum, subtract) Also, multiplexers programming inputs with some variables in the components receiving the operations from the data paths with some values zero or positive.

Arithmetic Blocks: Data path, control units and the resolution of different optimizations
To implement and install arithmetic circuits in the spaceship. Another important aspect is the communication protocol that allows the system to interchange data with other systems.the data path could also be considered as being a finite-state machine. Its internal states
are all the possible register contents. the number of internal states is
Really big in aerospace systems compared with another smaller quantities of dimensional data
Given by matrix outputs. the output state only depends on the internal state

```
Tdatastate ¼ t4 þ t1
Tdatadata ¼ t4 þ t2 þ t3;
```

where t1 is the computation time of the next internal state, t2 the computation time of the commands, t3 the maximum delay.

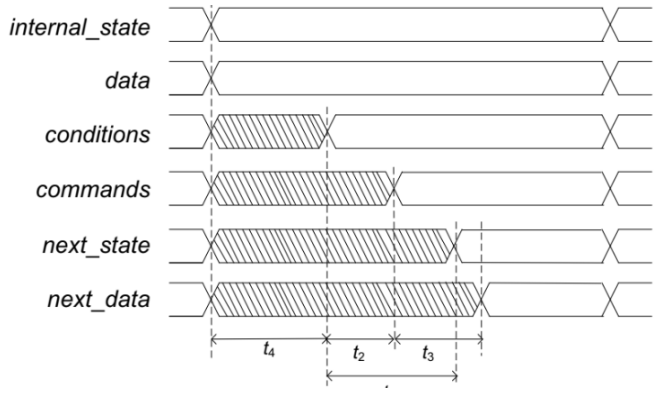
The clock period must satisfy a certain time(T) conditions

```
Tclk [max t f g 4 þ t1; t4 þ t2 þ t3 :
Tclk [ max t f g 4 þ t1; t2 þ t3 :
Tdatacommands ¼ t4 þ t2 and Tcommandsdata ¼ t3;
```

Operation scheduling consists of defining which particular operations are in course of execution during every clock cycle. Precedence relation defines which operations must be completed before starting a new one. Initial operands of some operation with variables B, R are important
To measure internal states in electronic aerospace systems:

```
x1 þ x2 þ x3 ¼ y1 þ y2:
```

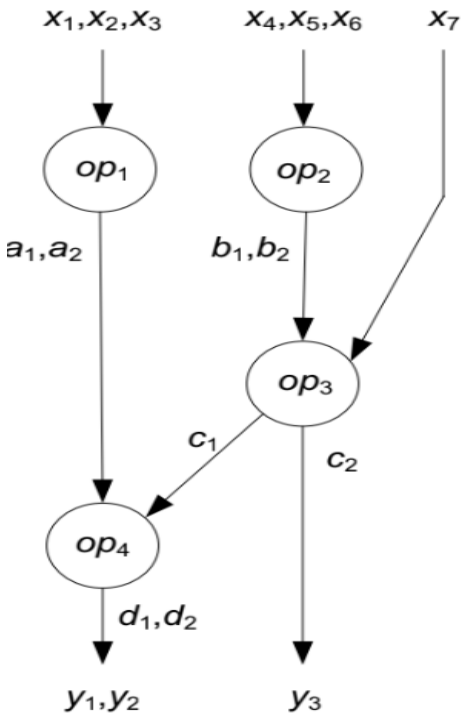
Synthesis-diagram-states:



The basic component of a carry-save adder is a 1-bit full adder: it is a combinational circuit with three binary inputs x, y and c, and two binary outputs z and d

```
x10 þ x20 þ x30 ¼ 2  y21 þ y10; x11 þ x21 þ x31
¼ 2  y22 þ y11;
x12 þ x22 þ x32 ¼ 2  y23 þ y12; x13 þ x23 þ x33
¼ 2  y24 þ y13
```

To compute y1, y2 and y3, the following operations are executed (op1 to op4 are labels)

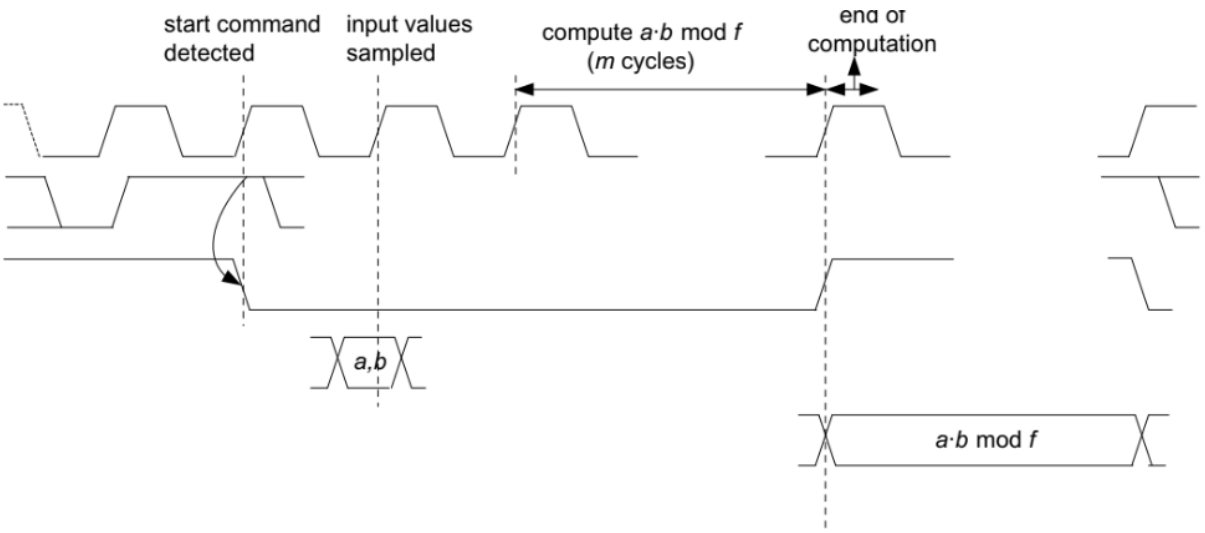


Description:

A combinational circuit made up of four carry-save adders and whose structure is the same as that Its computation time is equal to 3 TFA and its cost to 4 CCSA, being
CCSA the cost of a carry-save adder. So that the minimum clock cycle of a synchronous circuit including this 7-to-3 counter should be greater than 3 TFA.

```
a1 þ a2 ¼ x1 þ x2 þ x3;
b1 þ b2 ¼ x4 þ x5 þ x6;
c1 þ c2 ¼ b1 þ b2 þ x7;
d1 þ d2 ¼ a1 þ a2 þ c1;
```

```
0: (a1, a2) = CSA(x1, x2, x3),
(b1, b2) = CSA(x4, x5, x6);
1: (c1, c2) = CSA(b1, b2, x7);
2: (y1, y2) = CSA(a1, a2, c1), y2 = c2;
```

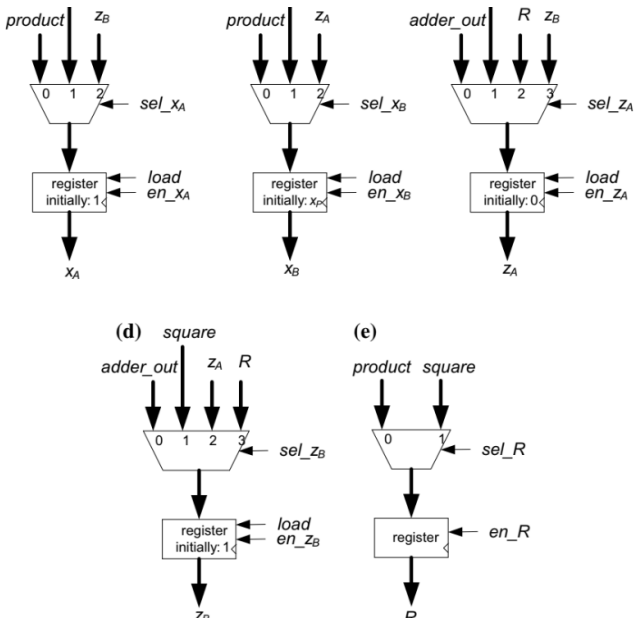


Bare metal algorithmic-series

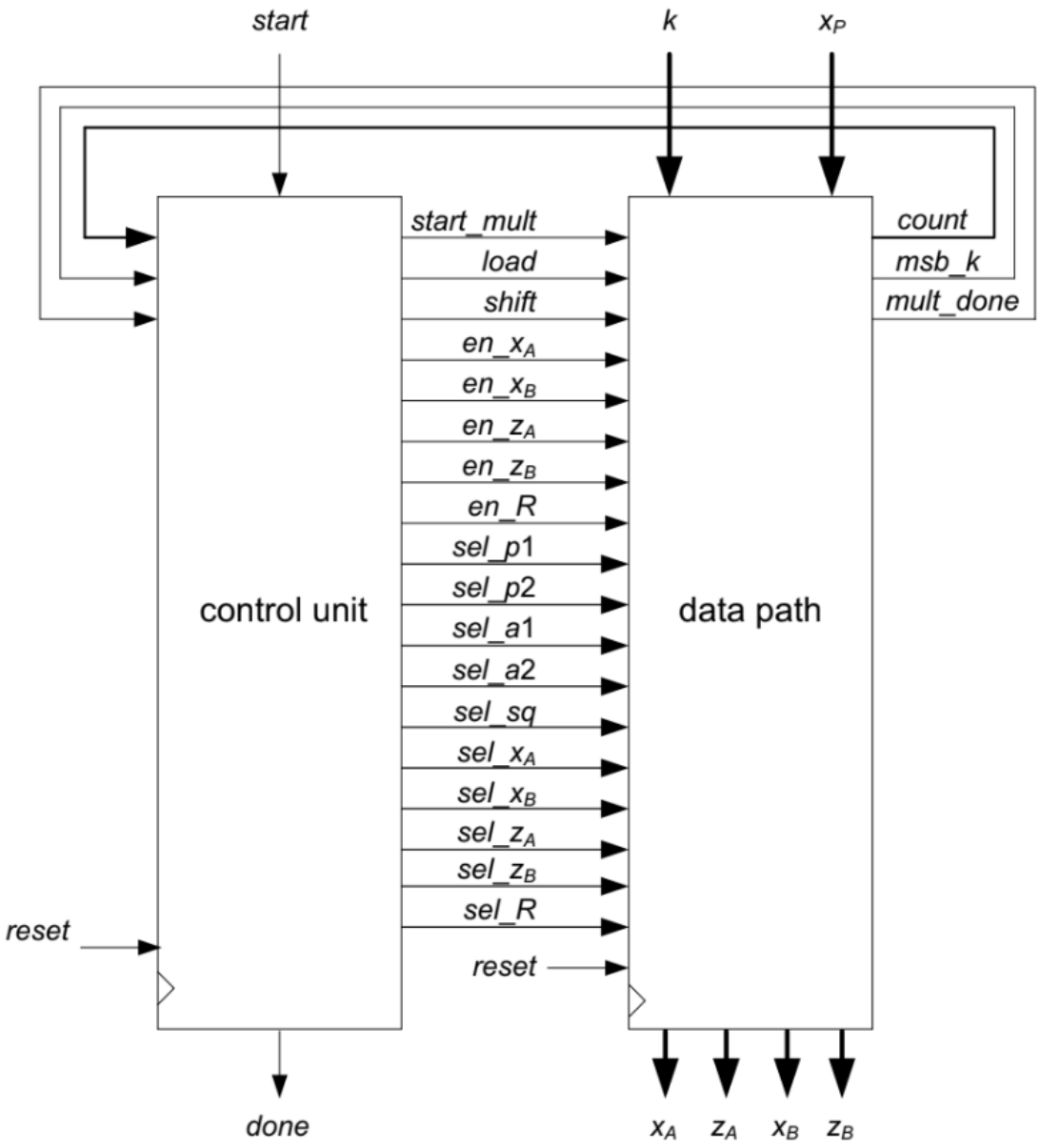
```
 [M, M + 1]
b [M, M + 1]
c [M + 1, M + 2]
d [M + 2, final]
e [3M, 3M + 1]
f [2M, 3M + 1]
g [3M + 1, final]
h [2M, 2M + 1]
i [2M + 1, final]
j [1, 2]
k [2, 3]
l [3, final]
xA [initial, M + 1]
zA [initial, M + 1]
xB [initial, 1]
zB [initial,1]
```

Algorithm structure for aerospace systems:

```
xA = 1, zA = 0, xB = xP, zB = 1;
for i in 1 .. m loop
if km-i = 0 then
zB = xA+zA, start(Z = xAzB);
zB = zB
2;
zB = zB
2;
wait until done;
R = Z;
start (Z = xBzA);
wait until done;
xB = Z;
zA =R+xB, start (Z = xAzA);
zA = zA
2;

wait until done;
xA = Z;
R=xA
2,

start(Z = RxB);
wait until done;
xB = Z;
start (Z = xPzA);
wait until done;
xA = Z;
(xA, zA, xB, zB) = (zB, R, xA+xB, zA);
```



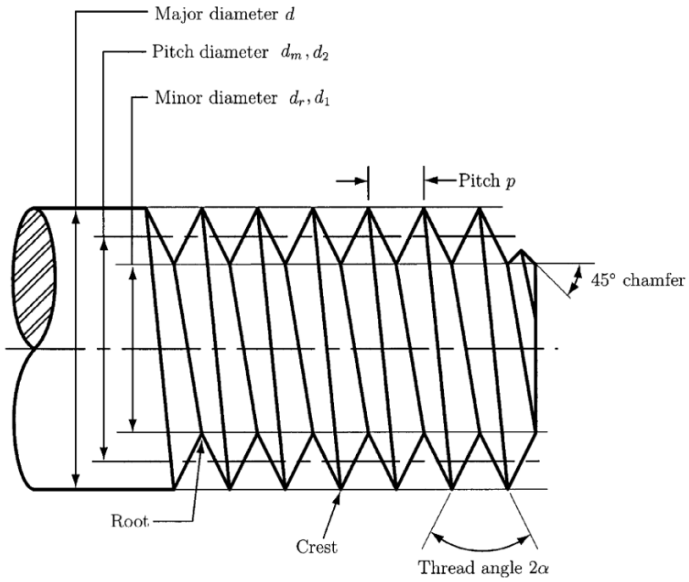
Scalar product electronic diagrams




```
SIGNAL s, p : BIT;
...
VARIABLE v : BIT;
VARIABLE t : TIME;
```

```
...
WAIT ON s; -- Wait for value changes on s
WAIT ON s UNTIL s = '1'; -- Wait for a rising edge on s
WAIT UNTIL s = '1'; -- Wait for a rising edge on s
```

Dynamical systems in Spaceships



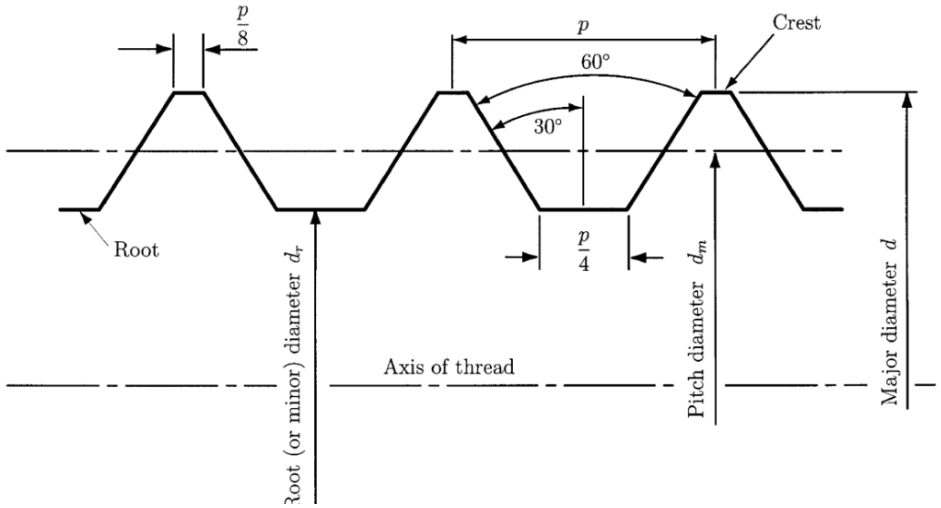
ure 1.1

Screw Thread;

Pitch, denoted by p, is the distance, parallel to the screw axis, between corresponding points on adjacent thread forms having uniform spacing. j Major diameter, denoted by d, is the largest (outside) diameter of a screw thread. j Minor diameter, denoted by dr or d1, is the smallest diameter of a screw thread. j Pitch diameter, denoted by dm or d2, is the imaginary diameter for which the widths of the threads and the grooves are equal.

Spaceship structures;

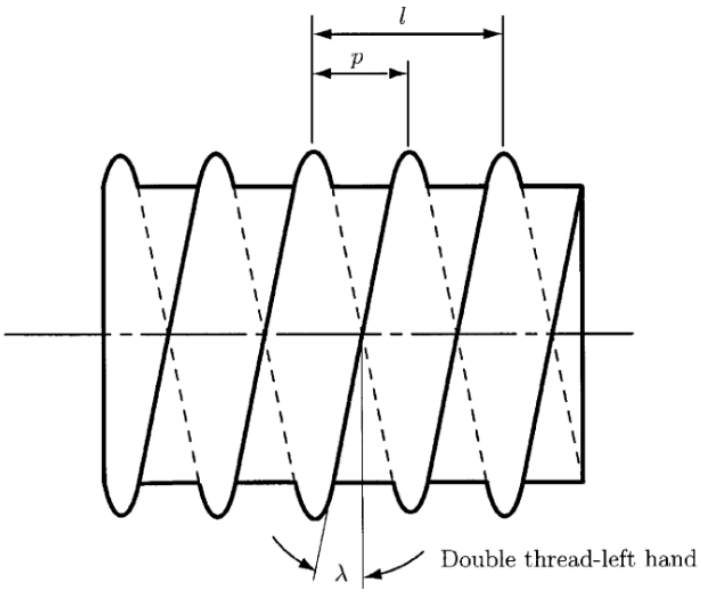
We are going to design some mapping and spaceship mechanical systems where could implement electronic algorithms for VHDL systems, RODOS, bare metal and VXworks to compute math and electronic variables inside our spaceship design.



Space-ships materials

A screw with two or more threads cut beside each other is called a multiple-threaded screw. The lead is equal to twice the pitch for a double-threaded screw, and to three times the pitch for a triple-threaded screw. The pitch p, lead l, and lead angle l.

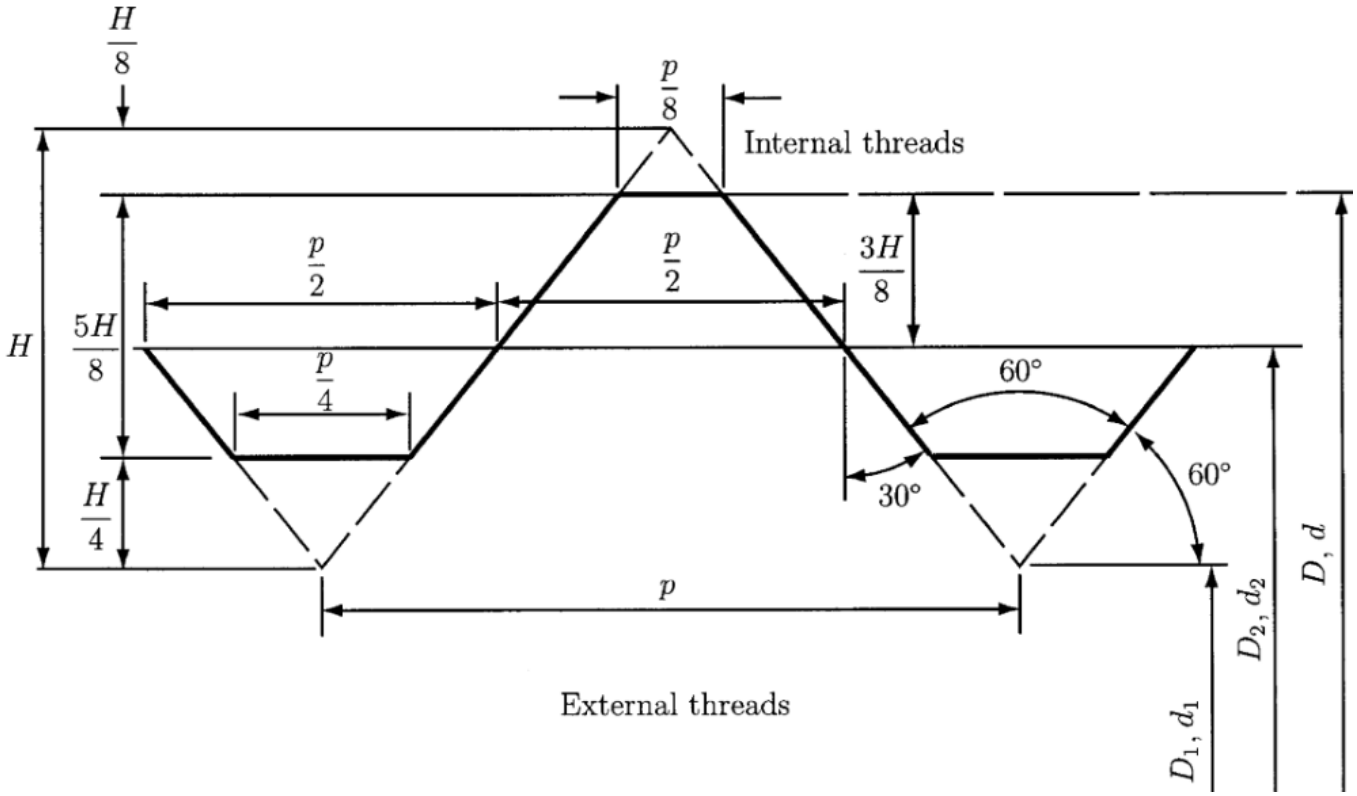
(d) is the basic major diameter of an internal (external) thread, D1...d1† is the basic minor diameter of an internal (external) thread, D2...d2† is the basic pitch diameter:



Screw size in the Uni®ed system is designated by the size number for major diameter, the number of threads per inch, and the thread series. In order to build an aerospace object or

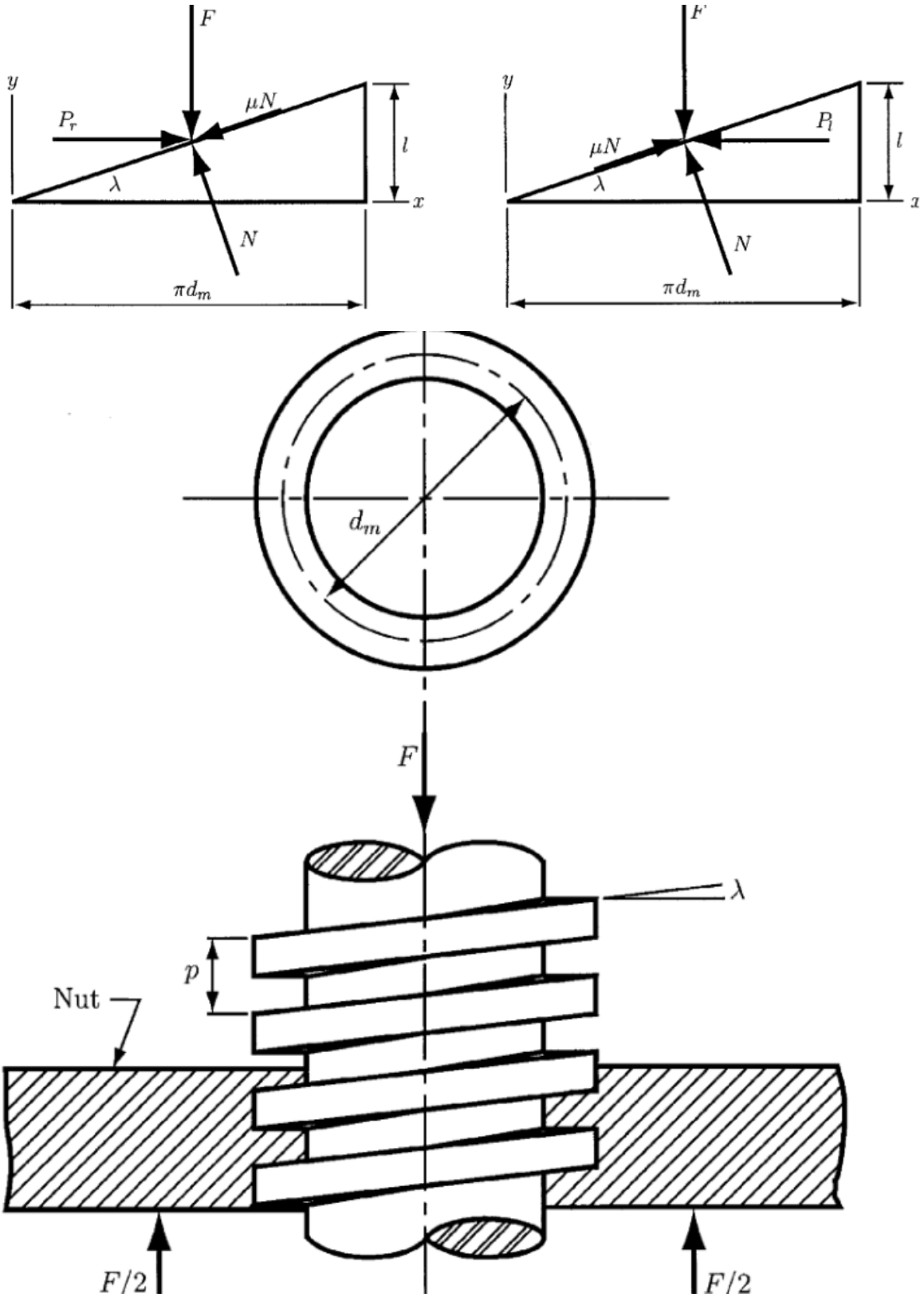
spaceship in which we can implements algorithmic circuits exposed in the first section of this text we need to implement some diagrams and mathematical functions to make the space-system works as expected by the mechanical hierarchies and controls:

Power screws are used to convert rotary motion to linear motion of the meshing member along the screw axis. These screws are used to lift weights.xert large forces (presses, tensile testing machines). The power screws can also be used to obtain precise positioning of the axial Movement. A square-threaded power screw with a single thread having the pitch diameter dm, the pitch p, and the helix angle l is considered



The screw is loaded by an axial compressive force F and the diameter dm

The spaceship rotates across some diameter center and the internal structure is mapped in Internal threads. We also need some bionic materials inside the structure and some rocket steel materials in our space objects buildings. High resistance materials, impedance electronic devices such resistors,capacitors and some filters for the electronic systems are needed to build spaceships.


F
y
P_r
μN
l
λ
x
N
πd_m
F
y
P_l
μN
λ
l
x
N
πd_m
d_m
F
λ
p
Nut
F/2
F/2

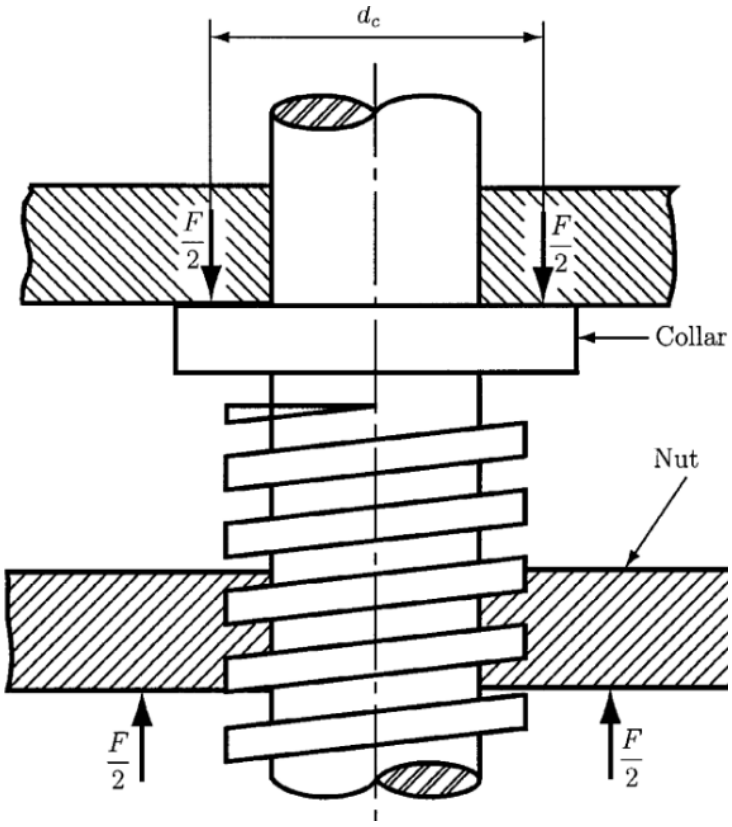
Design of spaceships & Circuits

Steps

1. The lead, the pitch (mean) diameter and the minor diameter
2. The torque required to raise the load
3. The torque required to lower the load
4. The efficiency

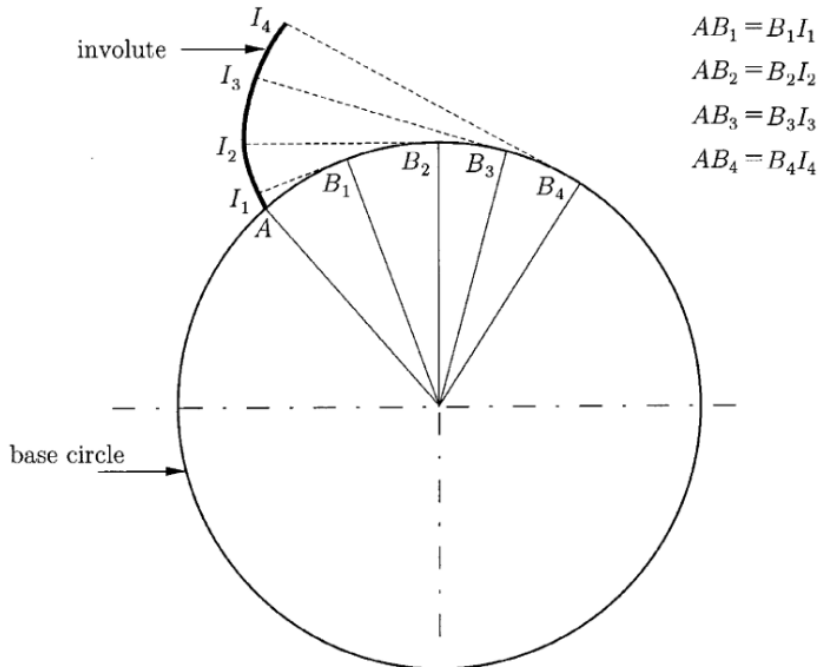
Equilibrium forces-variables(PFx, Pr, y, N, 1, y, cos1, sin1,k=0)

```
PFx ^ Pr ÿ N sin l ÿ mN cos l ^ 0 …
PFy ^ F ‡ mN sin l ÿ N cos l ^ 0:
```



The basic requirement of gear-tooth geometry is the condition of angular velocity ratios that are exactly constant, that is, the angular velocity ratio between a 30-tooth and a 90-tooth gear must

be precisely 3 in every position. The action of a pair of gear teeth satisfying this criteria is named conjugate gear-tooth action.

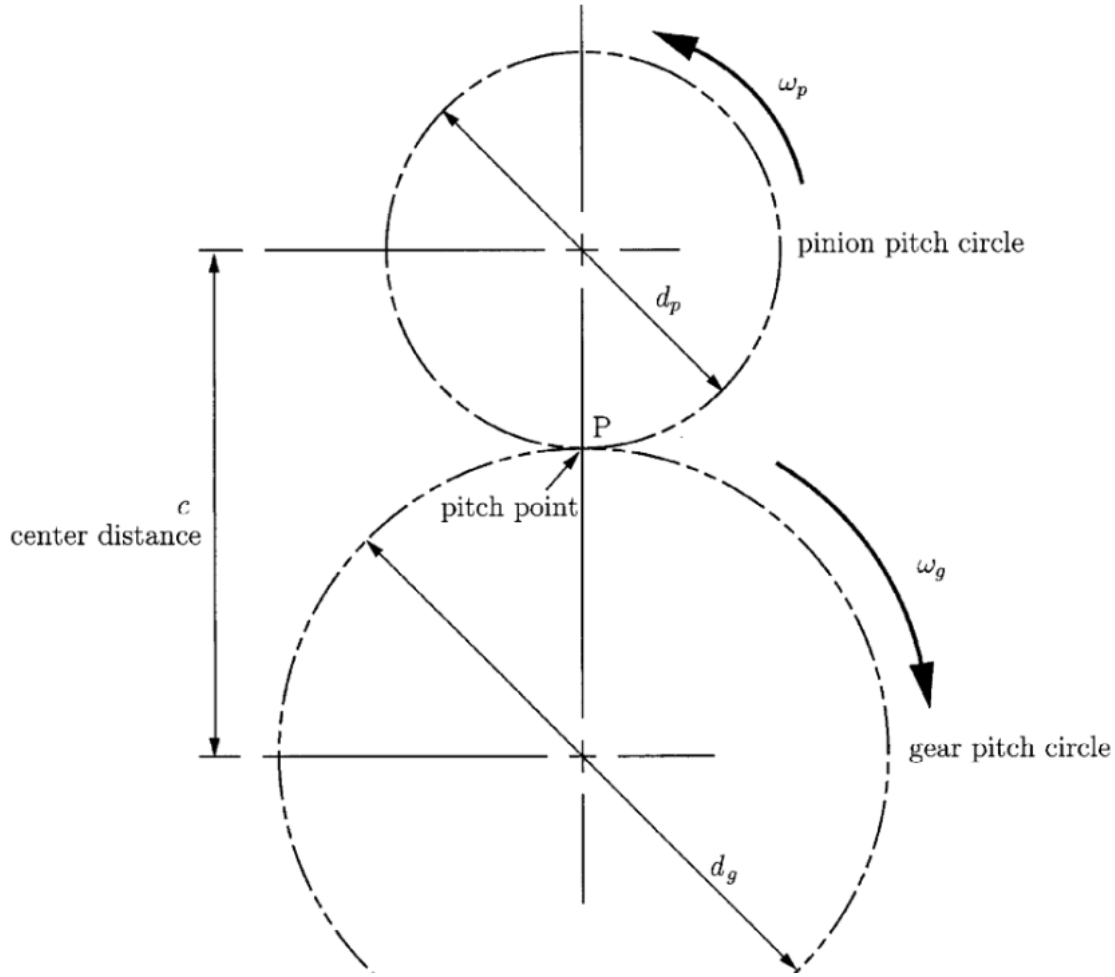


The circle that represents the cylinder is the base circle. The math operation represents an involute generated from a base circle of radius rb starting at the point A. The radius of curvature of the involute at any point.

```
B1 B2 B3 B4
I1 I2 I3 I4^
          formula:
root_square(r 2 ẏ r 2
b)
```

The variable (w) is the angular velocity and d is the pitch diameter; the minus sign indicates that the two gears rotate in opposite directions. The pitch circles are the two circles, one for each gear, that remain tangent throughout the Machine Components engagement cycle. The point of tangency is the pitch point. The diameter of The pitch circle is the pitch diameter. If the angular speed is expressed in rpm, then the symbol n is preferred instead of o. The diameter (without a qualifying adjective) of a gear always refers to its pitch diameter. We can compute the variable(w) or the angular velocity in python(Angular velocity in Python)

Pitch joins representations(Profiles)

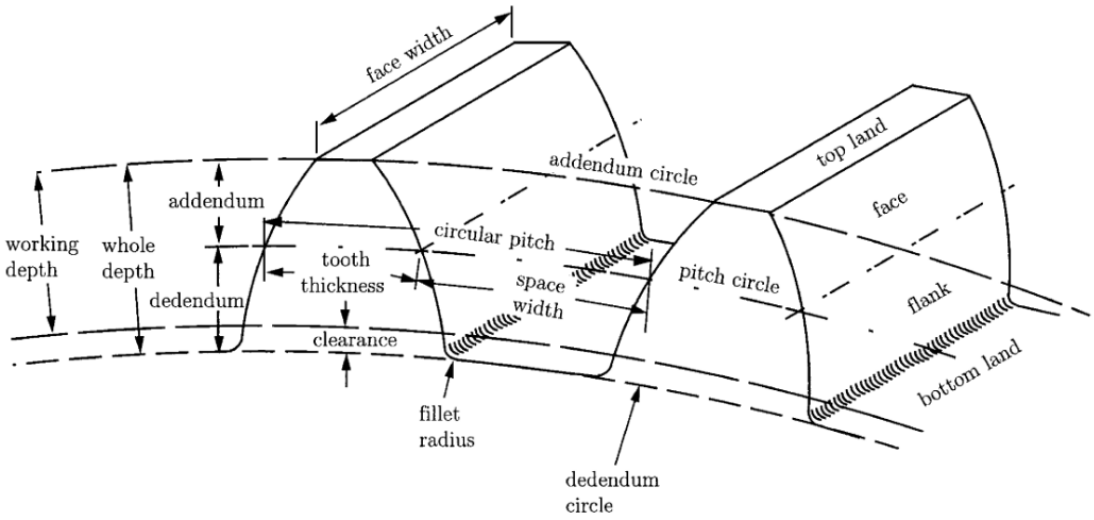


Center

c ˆ …dp ‡ dg †=2 ˆ rp ‡ rg

the addendum circle, ra ˆ r ‡ a. Similarly, the tooth profiles are extended inward from the pitch circle a distance called the dedendum, b.

The involute portion can extend inward only to the base circle. The Bullet at the base of the tooth merges the profile into the dedendum circle. The profile decreases the bending stress concentration. The clearance is the amount by which the dedendum in a given gear exceeds the addendum of its mating Gear. The circular pitch is designated as p and is measured in inches (English units) or millimeters (SI units). If N is the number of teeth in the gear (or pinion)



If N is the number of teeth in the gear (or pinion), then

```
p ^ pd=N ; p ^ pdp =Np ; p ^ pdg =Ng
```

Gear-tooth size are diametral pitch, Pd (used only with English units) and module, m (used only with SI). Diametral pitch is defined as the number of teeth per inch of pitch diameter:

```
Pd ^ N =d; Pd ^ Np =dp ; Pd ^ Ng =dg
```

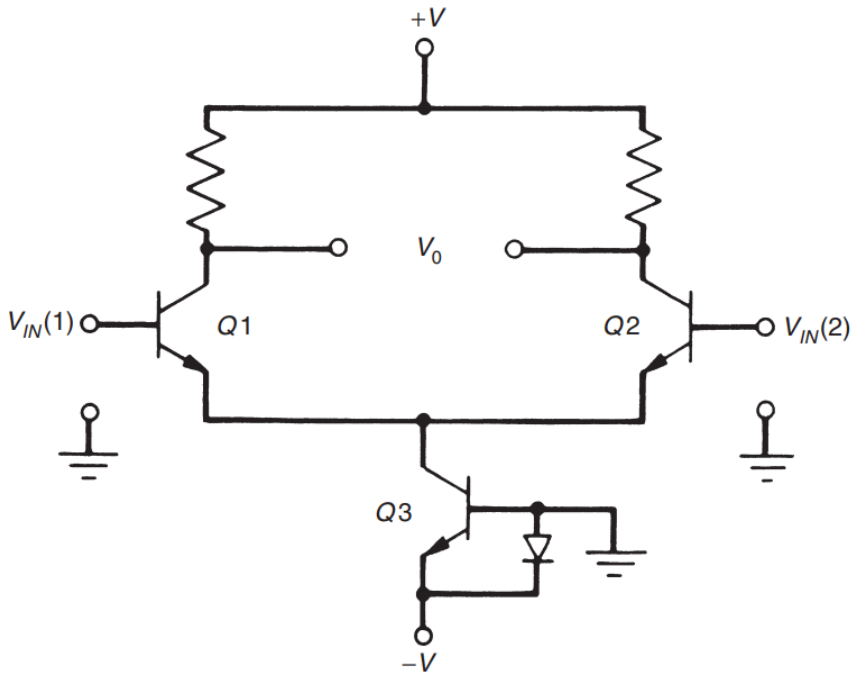
Module m, which is essentially the complementary of Pd , is de®ned as the pitch diameter in millimeters divided by the number of teeth (number of millimeters of pitch diameter per tooth):

```
m ^ d=N ; m ^ dp =Np ; m ^ dg =Ng
pPd ^ p … p in inches; Pd in teeth per inch†
p=m ^ p … p in millimeters; m in millimeters per tooth†
m ^ 25:4=Pd
```

Electromagnetic simulations over aerospace modules:

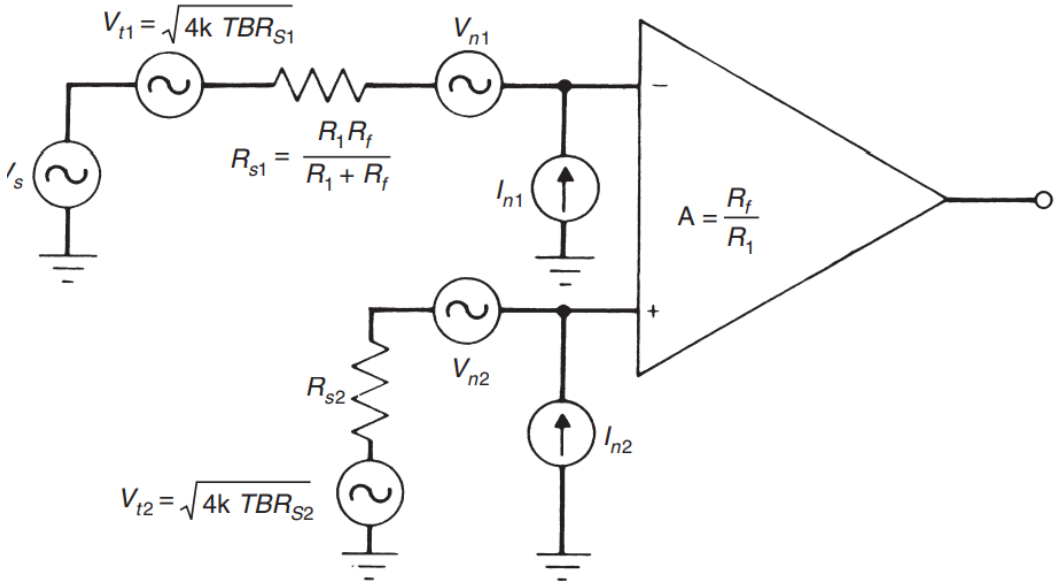
Typical input circuit schematic of an integrated circuit (IC) operational amplifier. Transistor Q3 acts as a constant current source to provide dc bias for the input transistors Q1 and Q2.A discrete bipolar transistor stage preceding an op-amp can often provide lower noise performance along with the other advantages of the operational amplifier. Op-amps do have

the advantage of a balanced input with low temperature drift and low input offset currents. The noise characteristics of an operational amplifier can best be modeled by using the equivalent input noise voltage and current Vn–In



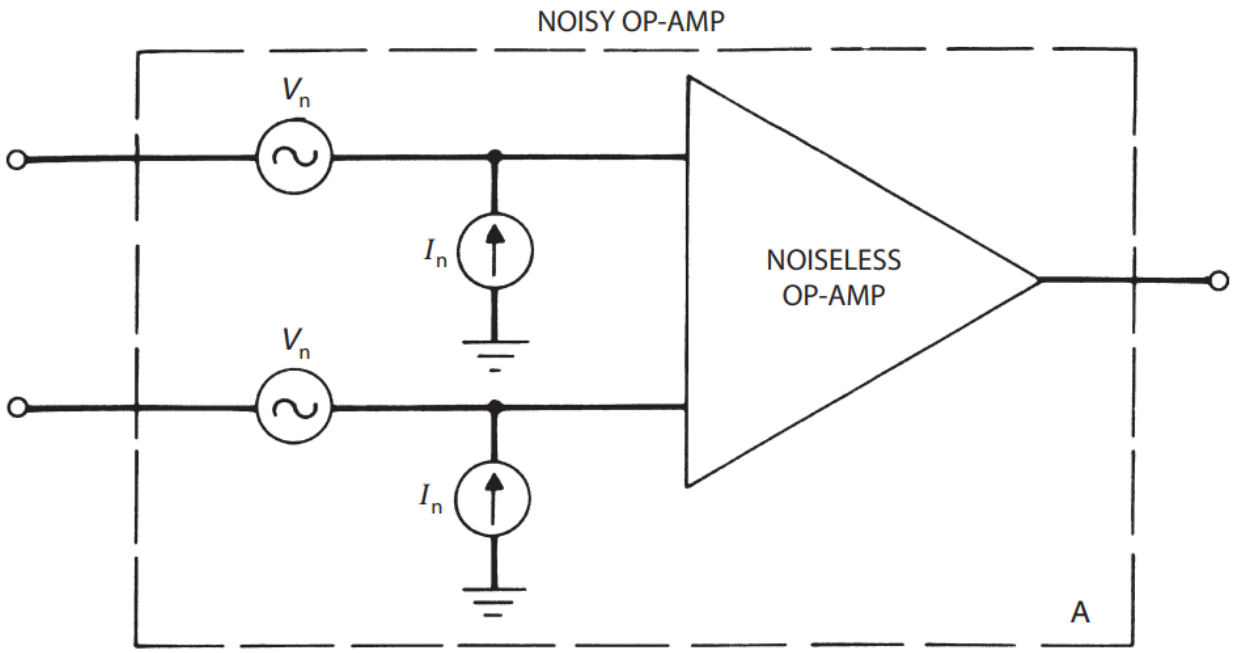
The two noise voltage sources

```
ðV0 nÞ 2 ¼ V2 n1 þ V2 n2
```



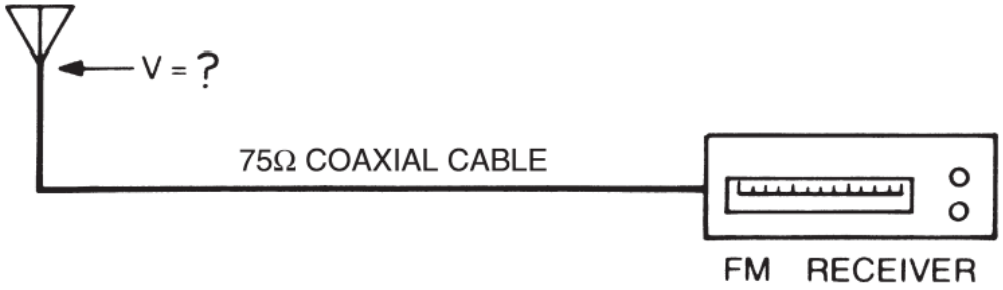
Description:

To obtain optimum noise performance (maximum signal-to-noise ratio) from an op-amp, the total equivalent input noise voltage Vnt should be minimized. the noise voltage given by the manufacturer is the combined value V0 n, whereas the noise current is the value that applies to each input separately In.



STEPS

- If the source resistance is a variable and the source voltage a constant in the design of a circuit, minimizing noise factor does not necessarily produce optimum noise performance.
- For a given source resistance, the least noisy circuit is the one with the lowest noise factor.
- For the best noise performance the output signal-to-noise ratio should be maximized, this is equivalent to minimizing the total input noise voltage (Vnt).
- The concept of noise factor is meaningless when the source is a pure reactance. For best noise performance a low-source resistance should be used (assuming the source voltage remains constant).
- Noise performance may be improved by transformer coupling the source resistance to a value equal to Rs=Vn/In



A digital system is also a radio-frequency (rf) system with significant noise and interference potential. Although most digital designers are knowledgeable about the subject of digital design, they are not always well equipped to handle the design and analysis of rf systems, which is exactly what they are designing.analog-circuit designers are now designing digital circuits, and they may not realize that different techniques are required for grounding, power distribution, and interconnection. The magnitude of the voltage generated when current changes through an inductor is:

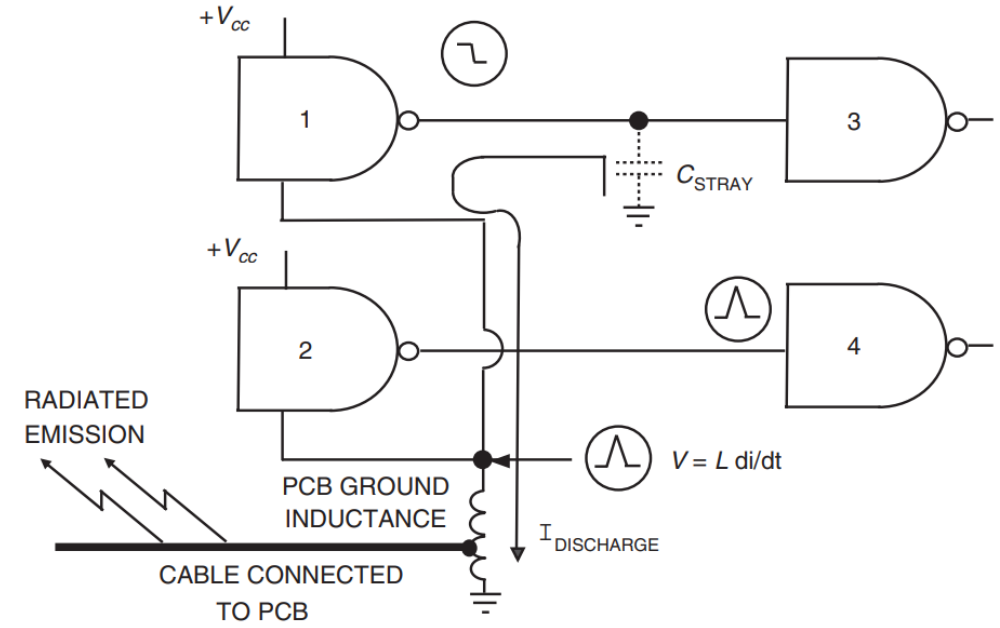
¼ L di dt ;

where L is the inductance and di/dt is the rate of change of current. For example, consider the case where the power-supply wiring has an inductance of 50 nH. If the transient current, when a logic gate switches, is 50 mA and the gate switches in 1 ns, the noise voltage generated across the power-supply wiring when this one gate changes states revolution joints in aerospace systems. Considering electromagnetic capability (EMC), however, it is better to think in terms of the frequency domain.the bandwidth of a digital signal can be related to the rise time tr;

BW ¼ 1 ptr :

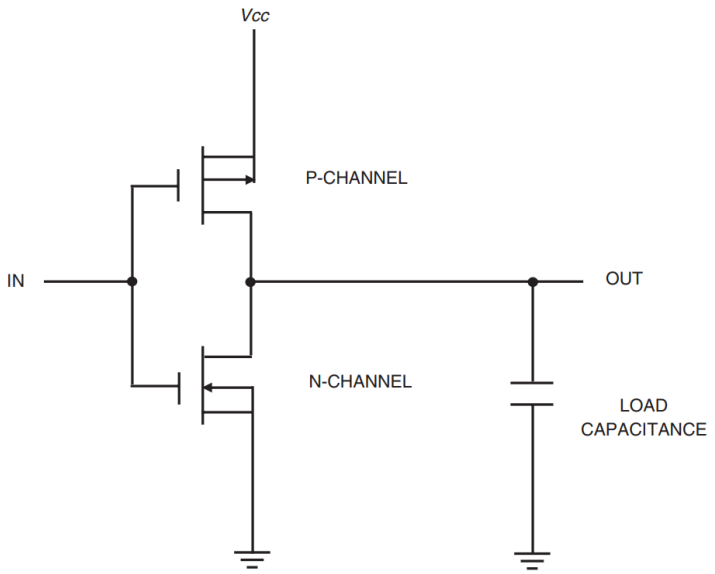
A rise time of 1 ns is equivalent to a bandwidth of 318 MHz. For complementary metal–oxide semiconductor (CMOS) circuits, the noise margin is about 0.3 times the Vcc voltage, or 1.5 V for a 5-V supply. Therefore, digital circuits have an inherent immunity to low-level noise pickup. However, as a consequence of the trend toward lower and lower supply voltages for digital logic. In digital circuits, the internal noise sources are usually the major concerns. Internal 380 DIGITAL CIRCUIT GROUNDING noise in digital circuits is the result of the following: (1) ground bus noise (often referred to as ‘‘ground bounce’’), (2) power bus noise, (3) transmission line reflections, and (4) crosstalk.

Circuit for inductance variables and positive charges. Inductance measures, R-resistance and C collision rates on higher frequencies in gates 1,2,3,4 with the cable connected to PCB in the case of aerospace systems electro-magnetic fields measurements;



The discharge path from the stray capacitance through the output of gate 1 and the ground conductors contains little resistance. Also,as other examples include an additional damping with a resistor, or ferrite bead, in the output of the gate will decrease this ringing.

For example, a CMOS inverter logic gate with a totem-pole output circuit (a pull-up transistor on top of a pull-down transistor)



Magnitudes are measured to f 50 to 100 mA per logic gate. Large integrated circuits, such as microprocessors can have transient power supply currents in excess of 10 A. This current is referred to by different names such as overlap current, contention current, or shoot-through current. A similar effect occurs with TTL, as well as most other logic families, however the peak transient currents are lower, because TTL has a current-limiting resistor in series.

To minimize the noise generated by these two internal noise sources, all digital logic systems must be designed with the following: 1. A low impedance (inductance) ground system 2. A source of charge (decoupling capacitors) near each logic IC. To minimize the noise from transient ground currents, the impedance of the ground must be minimized. A typical printed circuit board (PCB) trace (a 1-oz copper conductor 0.006 in. wide, and 0.02 in. from a return conductor) has a resistance of 82 mO/in.

```
Frequency (MHz)
Rise Time (ns)
Impedance (O) 1
```

To control inductance, it is helpful to understand how it depends on the physical properties of the circuit. Inductance is directly proportional to the length of a conductor.

Low inductance:

```
L ¼ 0:005 ln 4h d mH=in
```

The width w needed for a flat conductor to have the same inductance as a round conductor of diameter d can be determined:

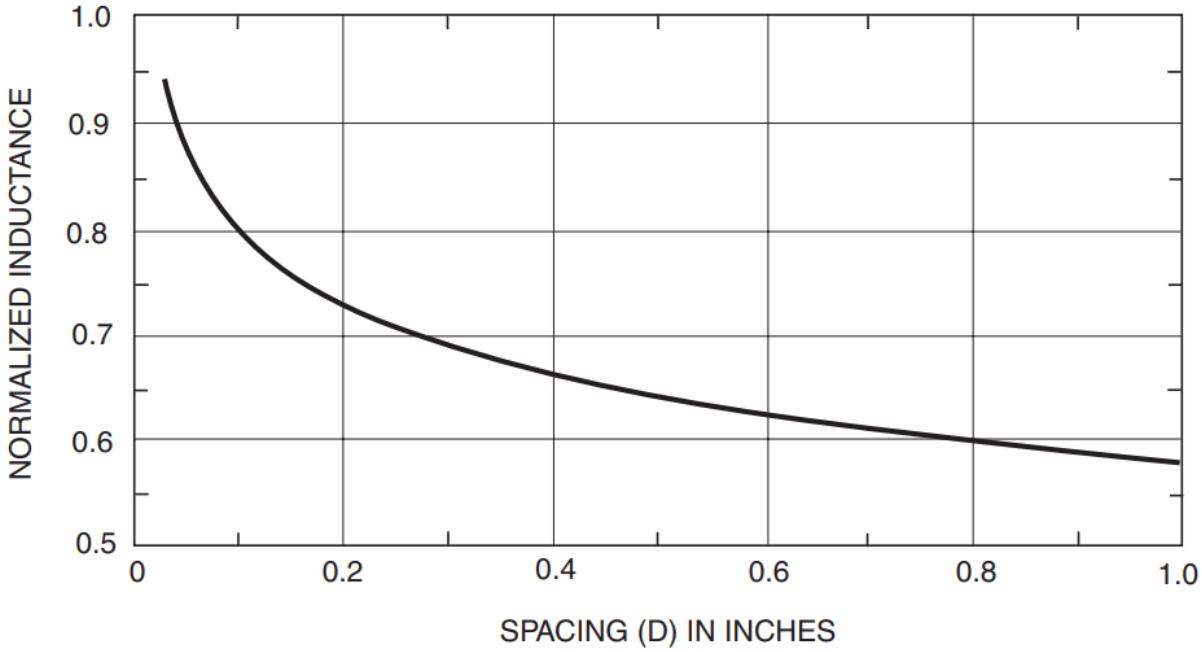
```
2w ¼ pd
```

It is difficult to achieve a large decrease in inductance by increasing the conductor diameter or width. In a typical case, doubling the diameter or width (an increase of 100%) will only decrease the inductance by 20%.

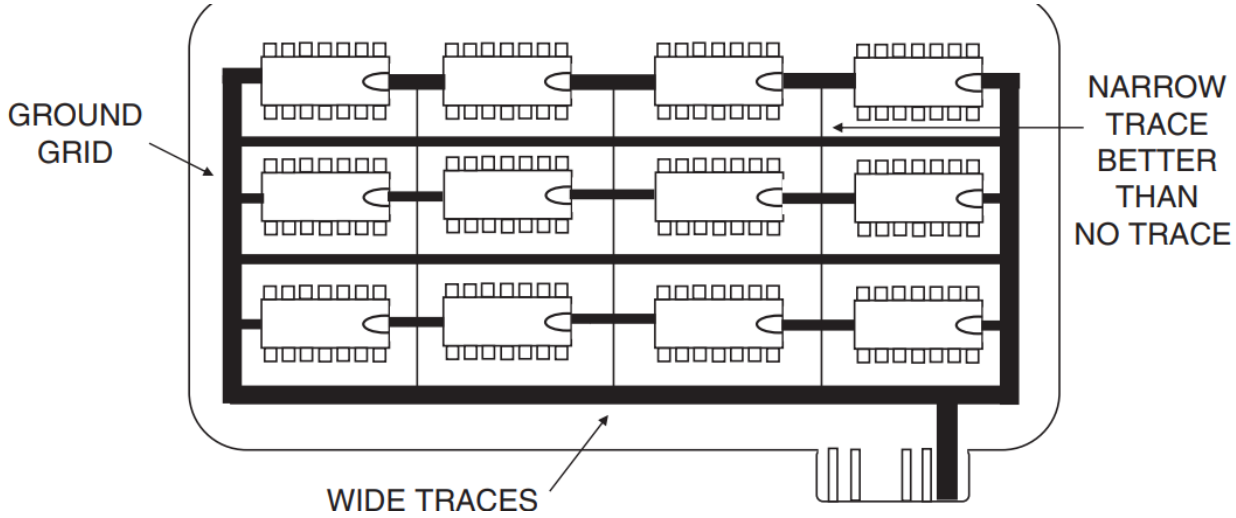
```
Lt ¼ L1 þ M 2 :
```

If the conductors are spaced far apart (loosely coupled), then the mutual inductance becomes small, and the total inductance approaches one-half the original inductance of a single

conductor. The effect of conductor spacing on mutual inductance must therefore be determined. A high-speed digital circuit ground system must provide a low impedance (low-inductance) connection between all possible combinations of ICs that communicate with each other. The most practical way to accomplish this is to provide as many alternative (parallel) ground paths as possible.



The impedance of an inductance (inductive reactance) is directly proportional to frequency.

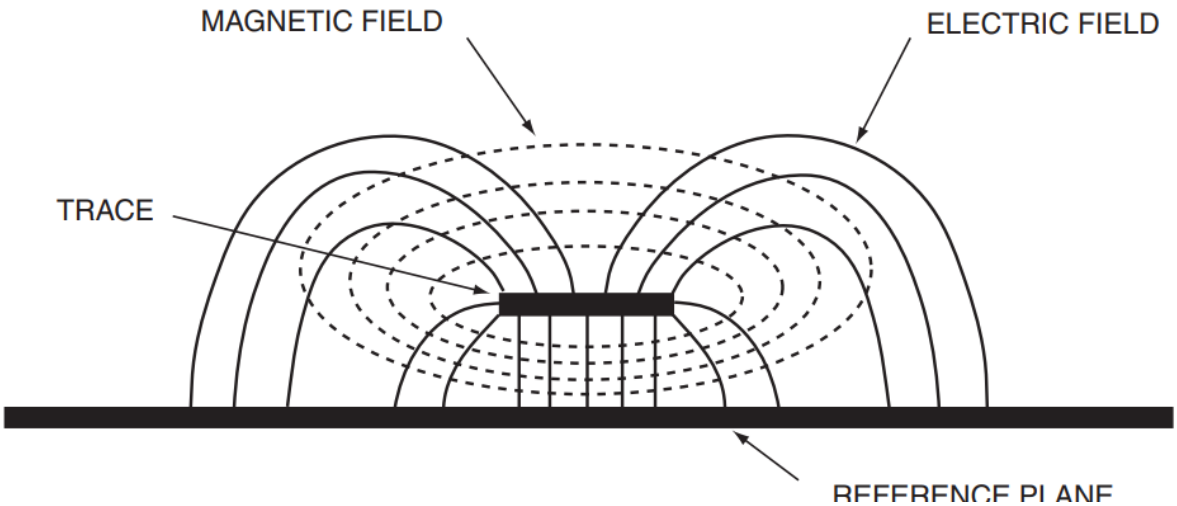


Another method of reducing inductance is to minimize the area of the loop enclosed by the current flow. Two conductors with current in opposite directions (e.g., a signal and its ground return trace) have a total loop inductance Lt, equal to:

```
Lt ¼ Ll þ L2 2M
```

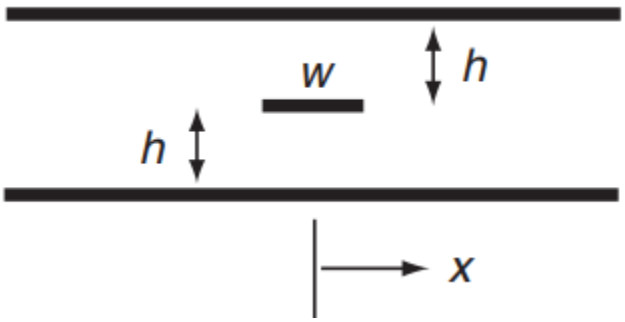
where L1 and L2 are the partial self-inductances of the individual conductors and M is the partial mutual-inductance between them. Placing signal and return current paths close together is therefore an effective way of reducing inductance. This can be done with a tightly twisted pair or a coaxial cable. With this configuration, inductances of less than 1 nH/in are possible.J(x) is the current density, and I is the total current in the loop. The current density of Eq. 10-13 is the distribution necessary to produce minimum inductance. The current density will be the same regardless of frequency; the only restriction is that the frequency is high enough that the resistance of the plane is negligible compared with the inductive reactance.

A method to minimize the total inductance, two conductors that carry current in the same direction (such as two ground conductors) should be separated. However, two conductors that carry current in the opposite direction (such as power and ground, or signal and ground conductors) should be placed as close together as possible.

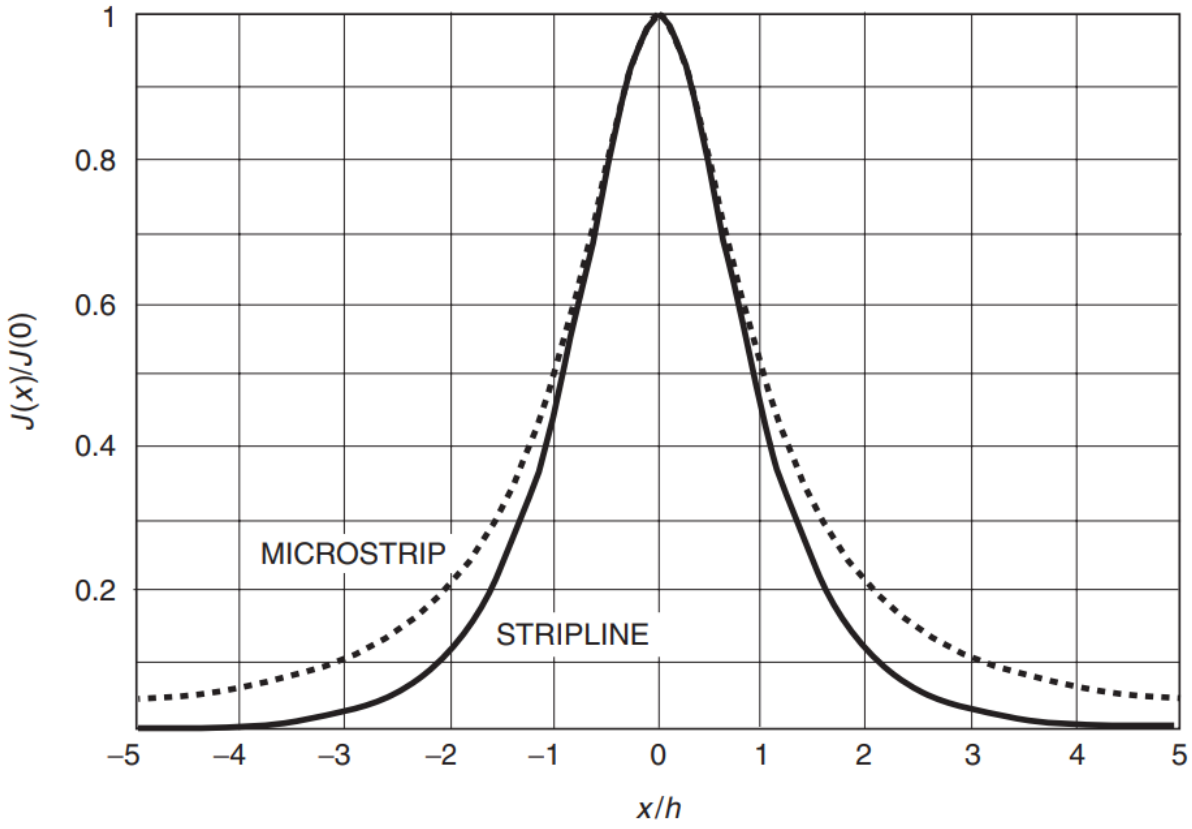


These concepts applied to mechanical systems hierarchically give some comprehension and results about aerospace objects such spaceships, rockets and military airplanes where the inductance, the collision resistance and the electromagnetic and electrostatic fields plays an important role on designing concepts such AVR microcontrollers, circuits and methods to reduce the inductance in electronic systems, devices, microprocessors for aircrafts etc..

the stripline current density to the microstrip current density. It plots the normalized current density J(x)/J(0) from the figure(h) and twice the normalized current density from figure(h) (which represents the total stripline current in both planes) versus x/h. As can be observed, for the case of a stripline, the current does not spread out nearly as far as in the case of a microstrip line.



MICROSTRIP FREQUENCIES



Static electricity can be created in many different ways,* but the most common is by contact and subsequent separation of materials. The materials may be solids, liquids, or gasses. When two nonconductors (insulators) are in contact, some charge (electrons) is transferred from one material to the other.. The relationship between charge, voltage, and capacitance is:

V ¼ Q C:

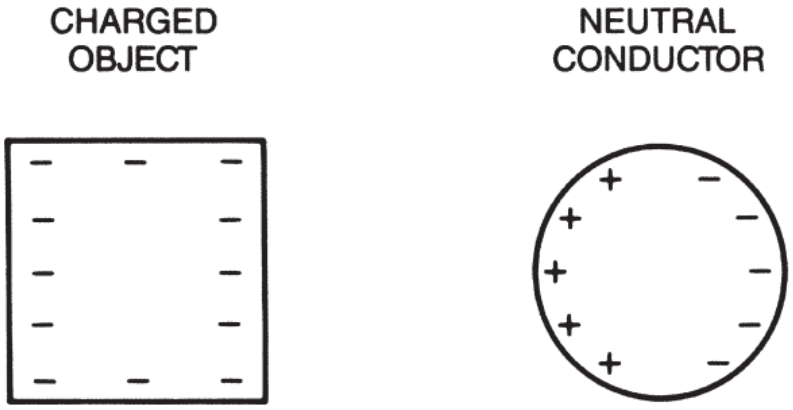
When the materials are close together, the capacitance is large; hence, the voltage is low. As the materials are separated, the capacitance decreases and the voltage increases.

STEPS

- A charge is generated on an insulator.
- This charge is transferred to a conductor by contact or induction.
- The charged conductor comes near a metal object and a discharge occurs.

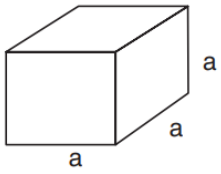
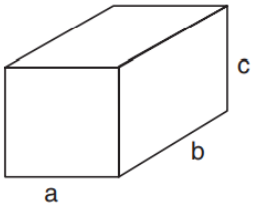
Polarity

An electrically charged object (insulator or conductor) is surrounded by an electrostatic field. If a neutral conductor is brought into the vicinity of a charged object, the electrostatic field will cause the balanced charges.



The polarity of charge opposite to that on the charged body will be on the surface of the neutral conductor nearest the charged body, and the opposite polarity charge will be on the surface farthest away. The conductor will remain neutral, however, with equal amounts of positive and negative charge. When the neutral object is moved away from the charged object, the positive and negative charges will recombine. All objects have a free-space capacitance of their own, the object itself being one of the plates and the second plate being located at infinity. This

represents the minimum capacitance that an object can have. The free-space capacitance of even an irregularly shaped object is determined primarily by its surface area.

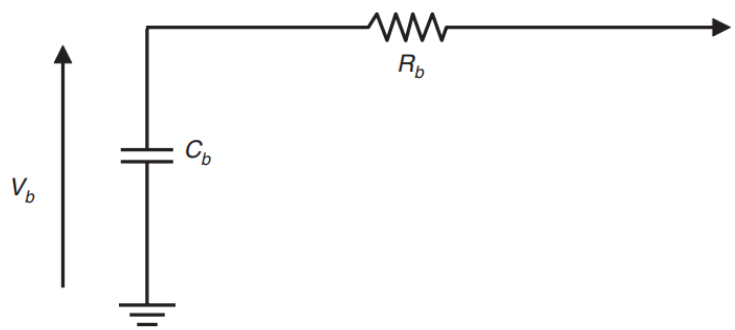
SPHERE		$S = 4\pi r^2$
CUBE		$S = 6a^2$
RECTANGULAR BOX		$S = 2[ab + bc + ac]$

Capacitance variable:

The capacitance between two parallel plates is equal to

```
C ¼ eA D
```

where A is the area of the plates and D is the distance between the plates. The solution is to measure the total capacitance of an object is then the combination of the free-space capacitance plus the parallel plate capacitance.



RANGE OF VALUES	
C_b	50 to 250 pF
R_b	500 to 10k Ω
V_b	0 to 20 kV

The dielectric constant for the material and s is the conductivity. The decay time can also be written in terms of the surface resistivity of the material and is

```
t ¼ er
Material Surface Resistivity (O/Square) Conductive 0 to 105
Static dissipative 105 to 109 Antistatica 109 to 1014
Insulativea W1014
```

Surface resistivity

Surface resistivity is measured with a fixture having two electrodes that form the opposite sides of a square. As long as the spacing between the electrodes is the same as the length of the electrodes, the resistance will be the same regardless of the length of the electrodes. That is, if the two electrodes are 3 cm long, they must be placed 3 cm apart. Antistatic materials are the slowest to dissipate charge. Nevertheless, they are useful because they can dissipate charge faster than it is generated and therefore prevent an object from accumulating a charge. An example of this is a pink polyethylene bag. To prevent triboelectric charging, the surface resistivity of a material should not exceed 1012 O per square.

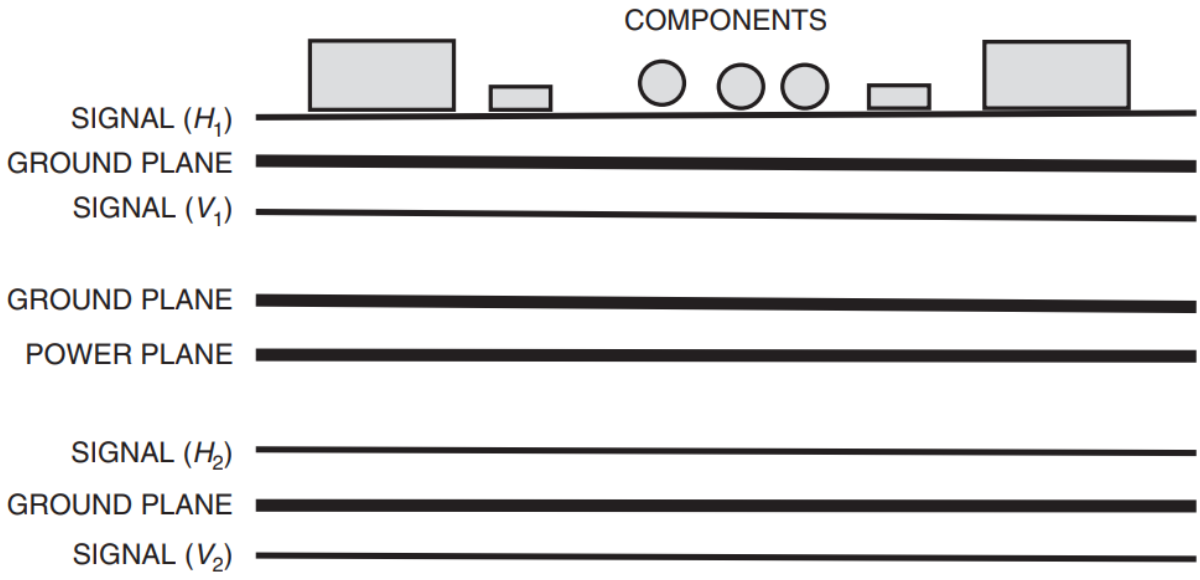
Steps to design circuits for the aerospace field

Effective design of the enclosure

- The first step in designing equipment to be immune to ESD is to prevent the direct discharge from flowing through the susceptible circuitry.

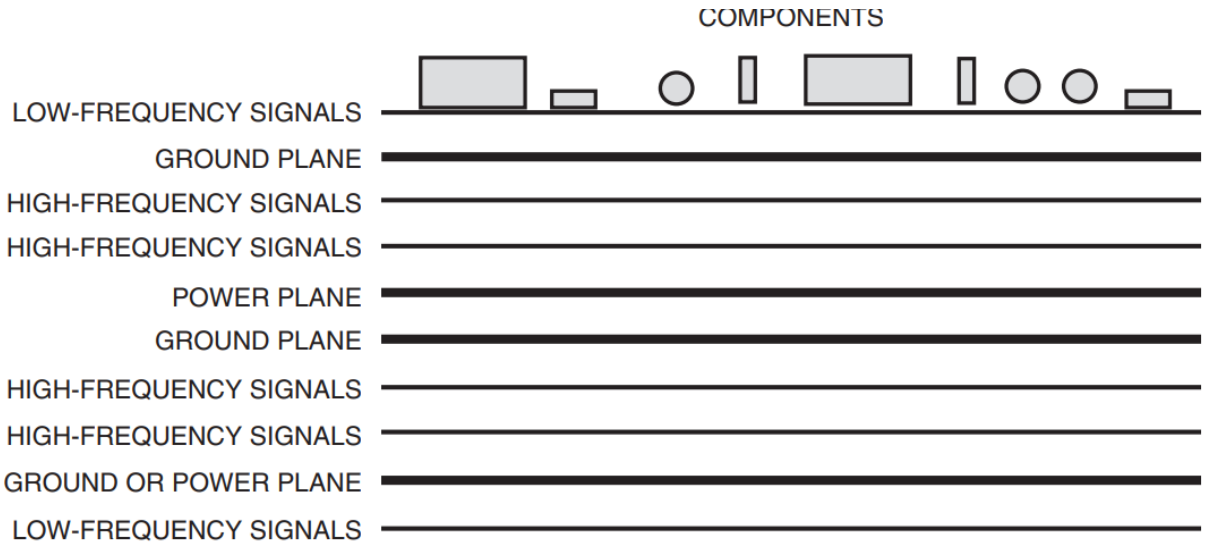
- Cable shielding Providing transient protection on all conductors of unshielded external cables
- Second, harden sensitive circuits, such as: Resets Interrupts Other critical control inputs Third, write transient hardened software capable of detecting, and if possible correcting, errors in the following: Program flow Input/output (I/O) data Memory
- By direct conduction 2. By field coupling, including a. Capacitive coupling b. Inductive coupling.
- Hard errors 2. Soft errors 3. Transient upset
- In the case of a product in a metallic enclosure, the enclosure can be used as an alternative path for the ESD current. To divert the ESD current effectively from sensitive circuits, all metallic components of the enclosure must be bonded together.

AVR microcontrollers



eight-layer board that can be used if split-power planes are required. It has two split-power planes and four routing layers. Typical layer spacing for this stackup might be 0.006 in/0.006 in/ 0.015 in/0.006 in/0.015 in/0.006 in/0.006 in. There is very little EMC advantage to using a board with more than eight layers. More than eight layers are usually only used when additional signal routing layers are required. If six routing layers are needed, then a 10-layer board should be used.

High layer count boards (10 plus) require thin dielectrics (typically 0.006 in or less on a 0.062 in thick board) and therefore they automatically have tight coupling between all adjacent layers and satisfy objectives #2 and #3. When properly stacked and routed, they can meet five or even all six of the objectives, and will have excellent EMC performance and signal integrity.



XTAL frequencies in AVR PORTS in C to calculate frequencies

Using Timer1, write a program that toggles pin PORTB.5 every second, while at the same time transferring data from PORTC to PORTD. Assume XTAL = 8 MHz.

Solution:

```
#include "avr/io.h"
#include "avr/interrupt.h"

int main ()
{
      DDRB |= 0x20;           //make DDRB.5 output

      OCR0 = 40;
      TCCR0 = 0x09;           //CTC mode, internal clk, no prescaler

      TIMSK = (1<<OCIE0);     //enable Timer0 compare match int.
      sei ();                 //enable interrupts

      DDRC = 0x00;            //make PORTC input
      DDRD = 0xFF;            //make PORTD output

      while (1)               //wait here
            PORTD = PINC;
}

ISR (TIMER0_COMP_vect)        //ISR for Timer0 compare match
{
      PORTB ^= 0x20;          //toggle PORTB.5
}
```

PCB capacitance technology for layers 5 and 6, which improves the high frequency power/ground plane decoupling. Figure 16-24 is another possible stackup for a 10-layer board. This configuration gives up the closely spaced power/ground plane pair. In return, it provides three signal-routing-layer.

If a register/bit field is read-only, attempting to change it will have no effect
• If a register/bit field is write-only, it will normally always read as zero
• Some registers/bit fields have special write modes like set, clear, or toggle.

The following example is made for ATmega328P but should work with most ATtiny or ATmega devices. This program starts TIMER0 in fast PWM mode. The output of the PWM signal on PD6 is controlled by PB0. 1 on PB0 enables PWM output, 0 on PB0 disables it.When applying the following stimuli, it will create a log file that starts logging the value of PINB and PIND. After 20 cycles, it also starts to log TCNT0.

```
#include<avr/io.h>
int main(void)
{
 DDRB = 0x00;
 DDRD = 0xFF;
 PORTD = 0x00;
 OCR0A = 0x20;
 TCCR0A = (1<<COM0A0 | 1<<WGM01 | 1<<WGM00);
 TCCR0B |= (1<<CS00);
 while(1)
 {
 if(PINB & 0x01)
 TCCR0B |= (1<<WGM02);
 else
 TCCR0B &= ~(1<<WGM02);
 }
}
```

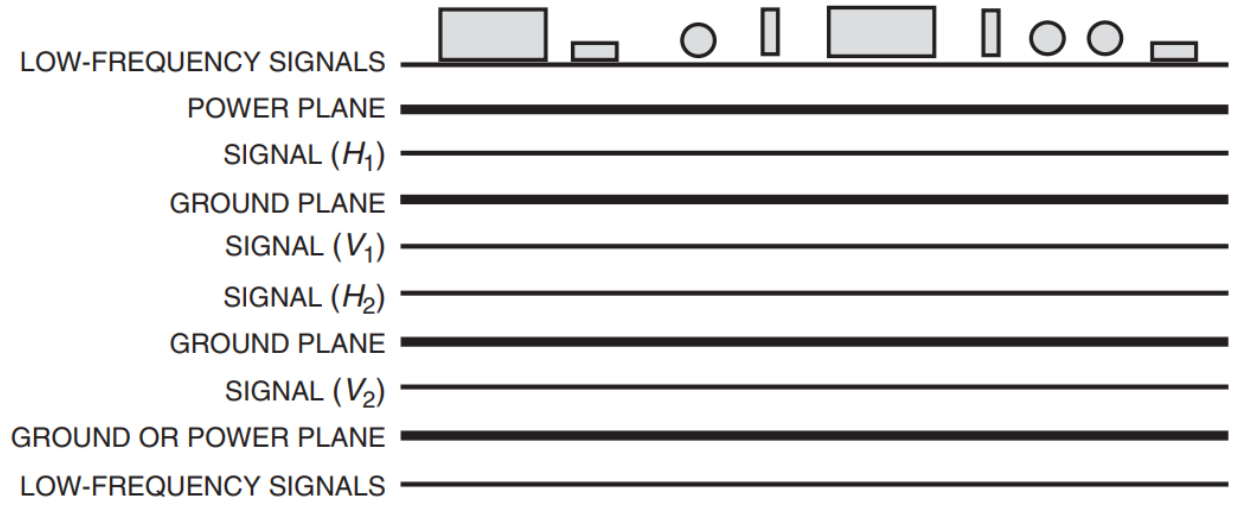
```
// Set pin PB0 to '1', Start PWM output PINB |= 0x01 #5 // Stop logging TIMER0
counter $unlog TCNT0 #200 // Set pin PB0 to '0', Stop PWM output PINB &= 0xFE
#200 $stoplog $break
```

MUX ARCHITECTURE:

```
entity MUX4 is
  port (SEL       : in  STD_LOGIC_VECTOR(1 downto 0);
        A, B, C, D: in  STD_LOGIC_VECTOR(1 downto 0);
        F         : out STD_LOGIC_VECTOR(1 downto 0));
```

```
end MUX4;

architecture BEHAVIOUR of MUX4 is
  signal SEL1, SEL1B, SEL0, SEL0B: STD_LOGIC;
begin
  SEL1  <= SEL(1);
  SEL1B <= not SEL(1);
  SEL0  <= SEL(0);
  SEL0B <= not SEL(0);

  F <= (A and SEL1B and SEL0B) or
       (B and SEL1B and SEL0 ) or
       (C and SEL1  and SEL0B) or
       (D and SEL1  and SEL0 );
end;
```

SIGNALS/PINS/MICROPROCESSOR/AVR




```
TCNT0 = 0x09 TCNT0 = 0x0a #1 TCNT0 = 0x0b #1 TCNT0 = 0x0c #1 TCNT0 = 0x0d #1
TCNT0 = 0x0e #1 TCNT0 = 0x0f #1 PINB = 0x01 TCNT0 = 0x10 #1 TCNT0 = 0x11 #1
TCNT0 = 0x12 #1 TCNT0 = 0x13
```

PCB designer is faced with the choice of either shielding critical signal layers by burying them between planes (objective to program PINS) or of routing critical signals on two layers that are adjacent to the same plane:

```
-- Component declaration...
  component MUX4
    port (SEL, A, B, C, D: in V2;
          F : out V2);
```

```
  end component;
```

The architecture process to tests the components of AVR

```
-- a process declaration...
  P: process
  begin
    SEL <= "00";
    wait for 10 NS;
    SEL <= "01";
    wait for 10 NS;
    SEL <= "10";
    wait for 10 NS;
    SEL <= "11";
    wait for 10 NS;
    wait;
  end process P;

  -- Concurrent assignments...
  A <= "00";
  B <= "01";
  C <= "10";
  D <= "11";
```

The layer spacing The assigning of signal layer pairs for orthogonal routing of signals The assignment of signals (clock, bus, high speed, low frequency, etc.) to which signal-routing-layer pairs.

Layers for circuit-design

- **2/ 2/ 0/ Low cost 0/ 4/ 2/ 2/**
- Improved EMC and SI performance 1 to 4/6 4/ 2 Improved EMC and SI, as well as two additional routing layers 3/ 8/ 4/ 4 Improved EMC and SI performance 5/ 10/ 6/ 4
- Two additional routing layers 4 or 5 12 6 4 Improved EMC and SI performance 5 or 6/ 12/ 8/ 4/ Two additional routing layers 5/ 14/ 8/ 6 Improved EMC and SI performance 6

Program, read, and verify memories:

- Program, read, and verify fuses
- Program, read, and verify lockbits

Programming example with reset PORTS

`$break`		Break program execution. Stimuli file(s) remain open, and stimuli will be resumed when program execution is resumed.
`$repeat`	*number*	Start a repeat loop, repeat number times until `$endrep` directive
`$endrep`		End of a repeat loop
`$log`	*IO-register mask*	Set up register logging. If `mask` is specified, the log will only update when the bits in the mask change. The `mask` will be OR'ed with any previous mask for the same address. Logging will not start until `$startlog` directive is executed.
`$unlog`	*IO-register mask*	Stop register logging. If `mask` is specified, only the bits in the mask will stop being logged.
`$startlog`	*filename writemode*	Start logging to the named file. The `writemode` is optional, the default mode is to append to the file. **Table 1-3. Log Writemodes** Type: `a` — Append to file (default) Type: `o` — Overwrite any existing file
`$stoplog`		Stop logging
`$fuse`	*address value*	Set fuse byte at `address` to `value`. [2] Fuse addresses generally start at 0.
`$reset`	*type*	Reset device. Possible reset types are listed below. **Table 1-4. Reset Types**

```
reset: rjmp start start: ldi r16, 0xff out DDRA, r16 // PORTA => output clr
r0 out DDRB, r0 // input <= PORTB loop: in r0, PINB // Requires stimuli for
any action inc r0 out PORTA, r0 rjmp loop

-- Component instantiation..testbench CPU arquitecture
  M: MUX4 port map (SEL, A, B, C, D, F);
```

Vxworks environment to program avionics designs CPU

<table>
<tr><th>Directive</th><th>Arguments[1]</th><th>Description</th></tr>
<tr><td>$memload</td><td>file segment nocheck</td><td>Load the contents of file into the memory. On AVR designs, you can specify segment to select where to load the data. If you add nocheck to the end of the command, any checksum errors in the file will be ignored.
Table 1-5. Memory Segments<table><tr><th>Segment</th><th>Description</th></tr><tr><td>s</td><td>Data memory (default)</td></tr><tr><td>f</td><td>Flash</td></tr><tr><td>e</td><td>EEPROM</td></tr><tr><td>i</td><td>I/O</td></tr></table></td></tr>
<tr><td>$memdump</td><td>file adr size segment</td><td>Dump the contents of the memory to file, starting at adr and dumping size number of bytes. Optionally, specify segment to select which memory to dump.
Table 1-6. Memory Segments<table><tr><th>Segment</th><th>Description</th></tr><tr><td>s</td><td>Data memory (default)</td></tr><tr><td>f</td><td>Flash</td></tr><tr><td>e</td><td>EEPROM</td></tr><tr><td>i</td><td>I/O</td></tr></table></td></tr>
</table>

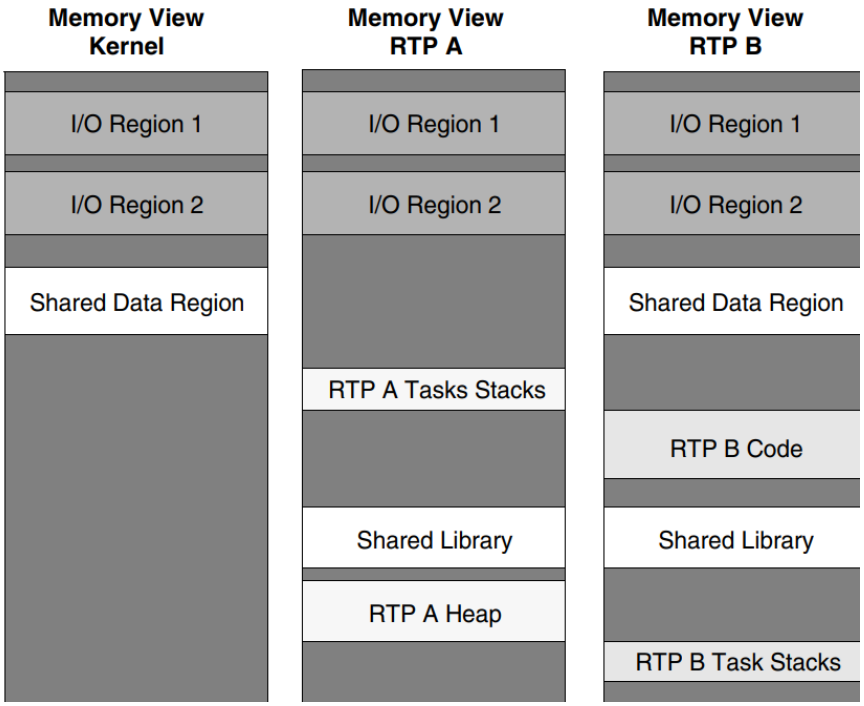
STEPS

- Determine the number of signal routing layers required
- Determine how to handle multiple dc voltages Determine the number of power planes required for the various system voltages.
- Determine whether multiple voltages will be on the same power plane layer, thereby requiring a split-plane, and routing restrictions on the adjacent layers Assign each signal layer pair to a solid reference plane.

KERNEL

The first memory view corresponds to the memory space accessible by kernel tasks. The second and third memory views correspond to the memory space accessible by tasks executing in RTP A, respectively RTP B. Note that the grayed areas are only accessible during system calls.32-Bit VxWorks can be configured to provide support for real-time processes on a system based on a processor without an MMU, or based on a processor with MMU but with the MMU disabled. VxWorks must, however, be configured with the INCLUDE_RTP component parameter

RTP_OVERLAPPED_ADDRESS_SPACE set to FALSE (as with the flat RTP virtual memory configuration:



There is no memory protection. That is, memory cannot be write-protected, and neither the kernel or any process are protected from other processes. The address space is limited to the available system RAM, which is typically smaller than it would be available on systems with MMU-based address translation enabled. Because of the smaller address space, a system is more likely to run out of large contiguous blocks of memory due to fragmentation. Not all processors and target boards can be used with the MMU disabled.

RTP

Real-time process (RTP) applications have a simple structural requirement that is common to C programs on other operating systems—they must include a main( ) routine. VxWorks provides C and C++ libraries for application development, and the kernel provides services for user-mode applications by way of system calls.VxWorks RTP executables are named with a .vxe file-name extension. They can be either relocatable or absolutely-linked objects depending on the virtual memory model in use on the VxWorks system. By default RTP applications are created as relocatable executables so as to be usable with either the flat virtual memory model or the overlapped virtual memory model.

RTP applications can be used for both the uniprocessor (UP) and symmetric multiprocessing (SMP) configurations of VxWorks. They must, however, only use the subset of APIs provided by VxWorks SMP.

```
int main ( int argc, /* number of arguments */ char * argv[], /*
null-terminated array of argument strings */ char * envp[], /* null-terminated
array of environment variable strings */ void * auxp /* implementation
specific auxiliary vector */ );
```

The argv[0] argument is typically the relative path to the executable. The envp and auxp arguments are usually not required by application code, but are used by the system as follows:

The envp argument is used for passing VxWorks environment variables to the application
And this kernel and memory examples are useful to work with C,C++ compilers doing automation with the EDA tools or with Atmel microchips in microchip studio:

```
#include <vxworks/kernLib.h>
#include <vxworks/errnoLib.h>
#include "tickLib.h"
#include "taskLib.h"

static int switch_slicing(struct threadobj *thobj, struct
timespec *quantum)
{
	struct sched_param_ex param_ex;
	int policy;

	param_ex.sched_priority = threadobj_get_priority(thobj);

	if (quantum) {
		policy = SCHED_RR;
		param_ex.sched_rr_quantum.tv_sec = quantum->tv_sec;
		param_ex.sched_rr_quantum.tv_nsec = quantum->tv_nsec;
	} else {
		policy = param_ex.sched_priority ? SCHED_FIFO :
SCHED_OTHER;
	}

	return threadobj_set_schedparam(thobj, policy, &param_ex);
}

STATUS kernelTimeSlice(int ticks)
{
	struct timespec quantum, *p = NULL;
	struct wind_task *task;
```

Suggestion for programmers;

I recommend for programming purposes use the VXworks kernels written in C or bare metal
In c++ the use of compilers and microchip simulators such WOOKI for arduino or C analysis such GDB debugger tools and strace utility to analyze the Atmega family microchips or Atmel core16 that are used in VXworks kernels and the CPU partitions to emulate the behavior of frequencies in the microchips that we can implement in Avionics projects or Aerospace ships, satellites or rockets.
Pyrocket is a standard python library to analyze and emulate rockets,fuels and some measurements for the aerospace field also.

Summary:
bare metal compiler
C compilers and EDA tools
*Arduino for Atmega32 or Atmel microchip analysis
*Instructions analysis and testing with compilers, EMACS..

An example of Vxworks libraries and kernels to work with the C compiler or EDA playground for analysis and testbench: [Memory-Partitions-kernel/VXworks]

```
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <memory.h>
#include <boilerplate/lock.h>
#include <boilerplate/ancillaries.h>
#include <copperplate/heapobj.h>
#include <vxworks/errnoLib.h>
#include <vxworks/memPartLib.h>
#include "memPartLib.h"
#define mempart_magic    0x5a6b7c8d
static struct wind_mempart *find_mempart_from_id(PART_ID partId)
{
      struct wind_mempart *mp = mainheap_deref(partId, struct wind_mempart);

      if (mp == NULL || ((uintptr_t)mp & (sizeof(uintptr_t)-1)) != 0 ||

          mp->magic != mempart_magic)

             return NULL;

      /*

       * XXX: memory partitions may not be deleted, so we don't need

       * to protect against references to stale objects.

       */
```

```
	return mp;

}
```

Because kernel mode and user mode have different instruction sets and MMU settings, RTP applications—which run in user mode—cannot directly access kernel routines and data structures (as long as the MMU is on). System calls provide the means by which applications request that the kernel perform a service on behalf of the application, which usually involves operations on kernel or hardware resources. System calls are transparent to the user, but operate as follows: For each system call, an architecture-specific trap operation is performed to change the CPU privilege level from user mode to kernel mode. Upon completion of the operation requested by the trap, the kernel returns from the trap, restoring the CPU to user mode. Because they involve a trap to the kernel, system calls have higher overhead than library routines that execute entirely in user mode. Note that if VxWorks is configured without a component that provides a system call required by an application, ENOSYS is returned as an errno by the corresponding user-mode library API.

System-calls(testing-file)VXworks

```
#include "tickLib.h"
#include <boilerplate/lock.h>
#include <vxworks/errnoLib.h>
#include <vxworks/sysLib.h>
int sysClkRateGet(void)

{
	unsigned int resolution;
	struct service svc;

	CANCEL_DEFER(svc);
	resolution = clockobj_get_resolution(&wind_clock);
	CANCEL_RESTORE(svc);

	return 1000000000 / resolution;

}
STATUS sysClkRateSet(int hz)

{
	struct service svc;
	int ret;
```

Tasks associated to msgQlib.c file:
Request-response-VXworks_servers

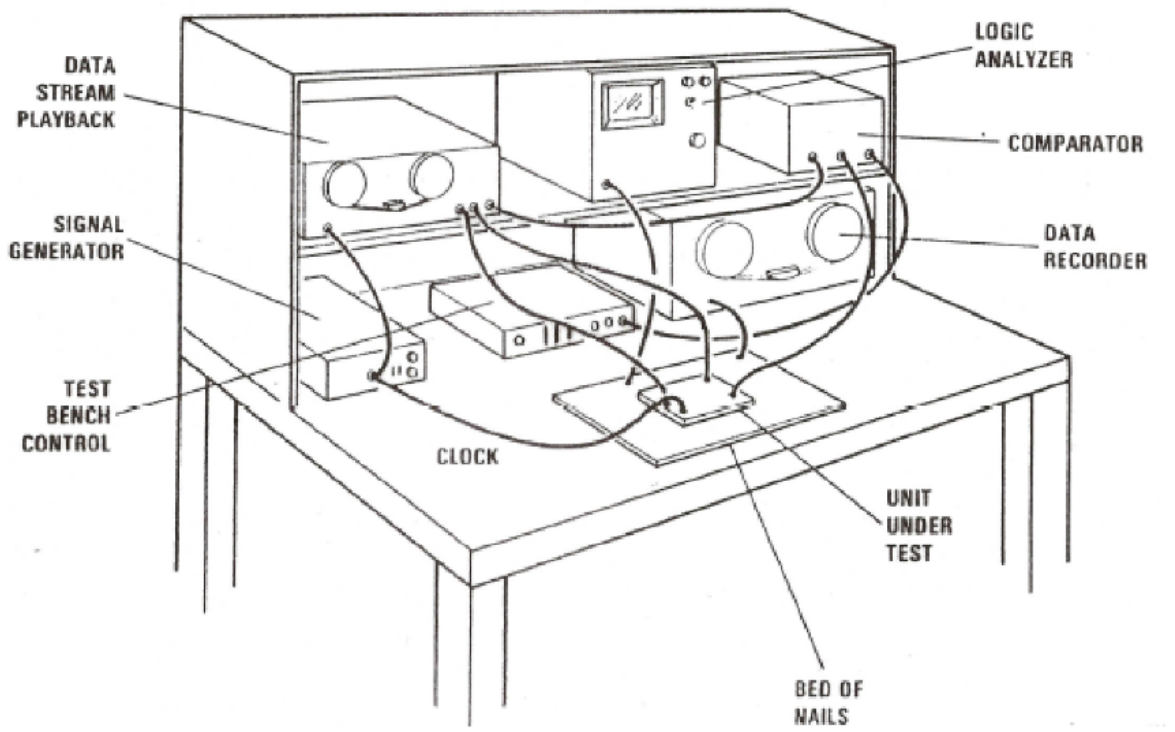
```
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <assert.h>
#include <memory.h>
#include <copperplate/heapobj.h>
#include <copperplate/threadobj.h>
#include <vxworks/errnoLib.h>
#include "reference.h"
#include "taskLib.h"
#include "msgQLib.h"
#include "tickLib.h"
#define mq_magic   0x4a5b6c7d

struct msgholder {
      int size;
      struct holder link;
      /* Payload data follows. */
};

static struct wind_mq *find_mq_from_id(MSG_Q_ID qid)
{
      struct wind_mq *mq = mainheap_deref(qid, struct wind_mq);

      if (mq == NULL || ((uintptr_t)mq & (sizeof(uintptr_t)-1)) != 0 ||
          mq->magic != mq_magic)
            return NULL;

      return mq;
}
```

The VxWorks kernel shell provides facilities for monitoring system calls. For more information, see the VxWorks Kernel Programmer's Guide: Target Tools, the VxWorks Application and the syscall monitor entry in the VxWorks Kernel Shell Command Reference, and the sysCallMonitor( ) entry in the VxWorks Kernel API Reference. VxWorks Libraries VxWorks distributions include libraries of routines that provide APIs for RTP applications. Some of these routines execute entirely in the process in user mode. Others are wrapper routines that make one or more system calls, or that add additional functionality to one or more system calls. For example, printf( ) is a wrapper that calls the system call write( ). The printf( ) routine performs a lot of formatting and so on, but ultimately must call write( ) to output the string to a file descriptor. Library routines that do not include system calls execute in entirely user mode, and are therefore more efficient than system calls, which include the overhead of a trap to the kernel. Task-info(kernels-architecture in C)GDB

Testbench-examples

```
/ Shows two loosely-timed initiators both with temporal decoupling and quantum
keeper

// Shows a bus with multiple initiators and multiple targets (four memories)
// Routes transactions to target and back using address decoding built into
the bus
// Propagates DMI calls on both forward and backward paths,
// with 'invalidate' being broadcast to every initiator

// Shows transaction pooling using a memory manager

#include "top.h"

int sc_main(int argc, char* argv[])
{
  Top top("top");
  sc_start();
  return 0;
}
```

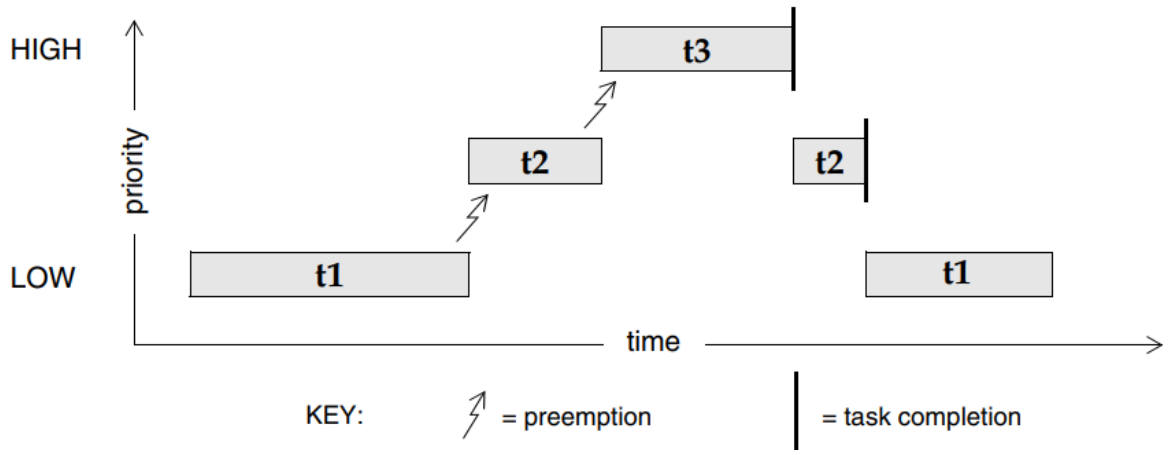
The Test Bench Concept



Bit reset & bit clock example to catch some frequencies in Mhz or Kz, testing frequencies for modules in microchips;

Algorithmic design-RAM-tests:

```
LIBRARY IEEE;
    USE IEEE.STD_LOGIC_1164.ALL;
    USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY RAM_TB IS
-- empty
END ENTITY;

ARCHITECTURE BEV OF RAM_TB IS

SIGNAL DATAIN : STD_LOGIC_VECTOR(7 DOWNTO 0):="00000000";
SIGNAL ADDRESS : STD_LOGIC_VECTOR(7 DOWNTO 0):="00000000";
SIGNAL W_R : STD_LOGIC:='0';
SIGNAL DATAOUT : STD_LOGIC_VECTOR(7 DOWNTO 0);

-- DUT component
COMPONENT RAM IS
    PORT(DATAIN : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
         ADDRESS : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
         W_R : IN STD_LOGIC;
         DATAOUT : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
         );
END COMPONENT;

BEGIN

  -- Connect DUT
  UUT: RAM PORT MAP(DATAIN, ADDRESS, W_R, DATAOUT);

  PROCESS
  BEGIN
    -- Write data into RAM
    WAIT FOR 100 ns;
    ADDRESS<="10000000";
    DATAIN<="01111111";
    WAIT FOR 100 ns;
    ADDRESS<="01000000";
    DATAIN<="10111111";
    WAIT FOR 100 ns;
```



ERRORS-ASSERTIONS:

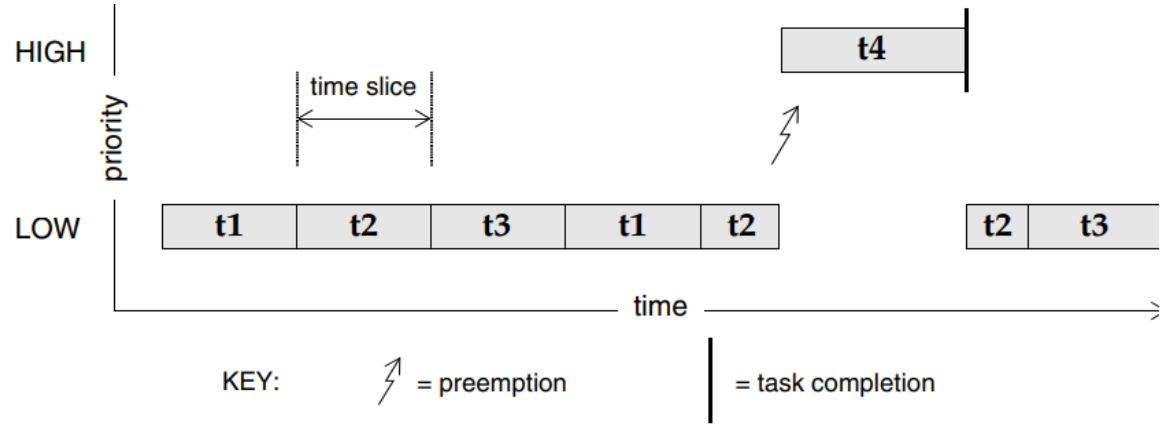
```
assert (A = (B + C)) -- expect A = B+C (any boolean condition) report "Error
message" severity NOTE;  Match data types for A, B, C  Print "Error message"
if assert condition FALSE (condition is not what we expected)  Specify one of
four severity levels: NOTE, WARNING, ERROR, FAILURE
```

Code

```
//including interfcae and testcase files
`include "interface.sv"

//-------------------------[NOTE]---------------------------------
//Particular testcase can be run by uncommenting, and commenting the rest
`include "random_test.sv"
//`include "wr_rd_test.sv"
//`include "default_rd_test.sv"
//----------------------------------------------------------------

module tbench_top;

  //clock and reset signal declaration
  bit clk;
  bit reset;

  //clock generation
  always #5 clk = ~clk;

  //reset Generation
  initial begin
    reset = 1;
```

Test-analysis:

We can analyze tests with some algorithms refers to the clock function and the nano seconds Throughout the time constant and the definition of wait for the high frequencies in Mz that comes from the microchip Atmel or Atmel. Other functions are after LT and Wait for LT.
LT is registered or a component section from the chip.




```
-- Simple 50% duty cycle clock clk <= not clk after T ns; --T is constant or
defined earlier -- Clock process, using "wait" to suspend for T1/T2 process
begin clk <= '1'; wait for T1 ns; -- clk high for T1 ns clk <= '0'; wait for
T2 ns; -- clk low for T2 ns end process; -- Alternate format for clock
waveform process begin clk <= '1' after LT, '0' after LT + HT; wait for LT +
HT; end process;
```

Next text - vxworks descriptions, Atmel studio (browser)

Waves book + Atmel START (next document on circuit analysis algorithms)