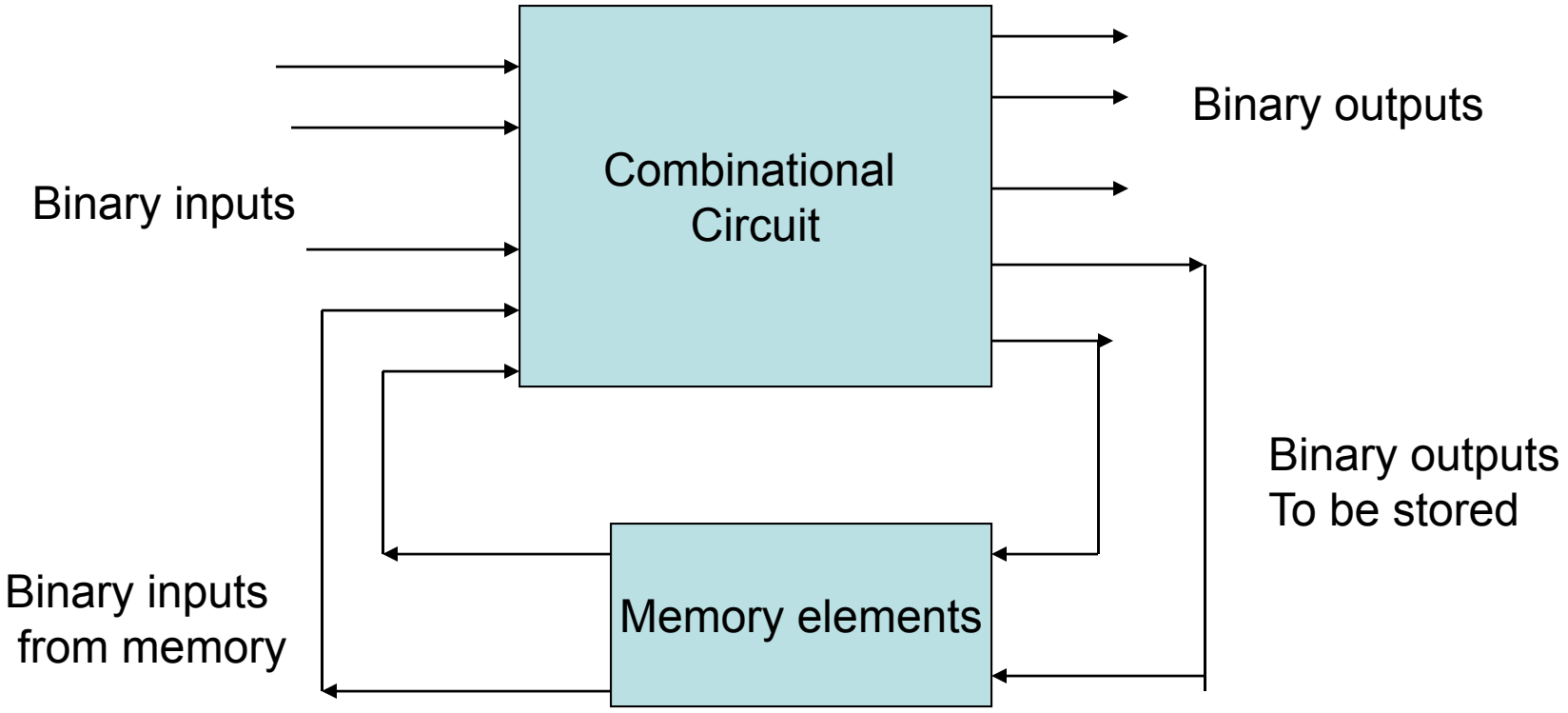
Discussion Assignment 1 : Some Hints

“Anyone who has never made a mistake has never tried anything new.”

— Albert Einstein

You are given two ropes and a lighter. Each of the two ropes has the following property: if you light one end of the rope, it will take exactly one hour to burn to the other end. It doesn’t necessarily burn at a uniform rate. How can you measure a period of 45 minutes?

Power of Binary Logic Circuits and Emergence of Software



Hindrance Function

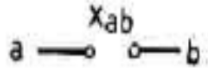


Figure 1 (left). Symbol for hindrance function

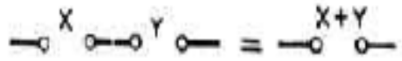


Figure 2 (right). Interpretation of addition

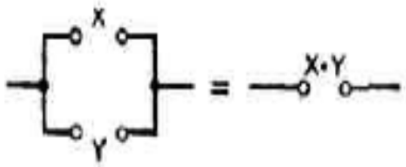


Figure 3 (middle). Interpretation of multiplication

1. *a*.	$0 \cdot 0 = 0$	A closed circuit in parallel with a closed circuit is a closed circuit.
b.	$1 + 1 = 1$	An open circuit in series with an open circuit is an open circuit.

Huntington's Postulates : Boolean Algebra

1. The class K contains at least two distinct elements.
2. If a and b are in the class K then $a + b$ is in the class K.
3. $a + b = b + a$.
4. $(a + b) + c = a + (b + c)$.
5. $a + a = a$.
6. $ab + ab' = a$ where ab is defined as $(a' + b')'$.

AND, OR and NOT is Functionally Complete

$$f(X_1,X_2,\ldots X_n) = X_1 \cdot f(1,X_2 \ldots X_n) + X_1' \cdot f(0,X_2 \ldots X_n) \ , \qquad (10a)$$

$$f(X_1,\ldots,X_n) = [f(0,X_2 \ldots X_n) + X_1] \cdot [f(1,X_2 \ldots X_n) + X_1'] \ . \qquad (10b)$$

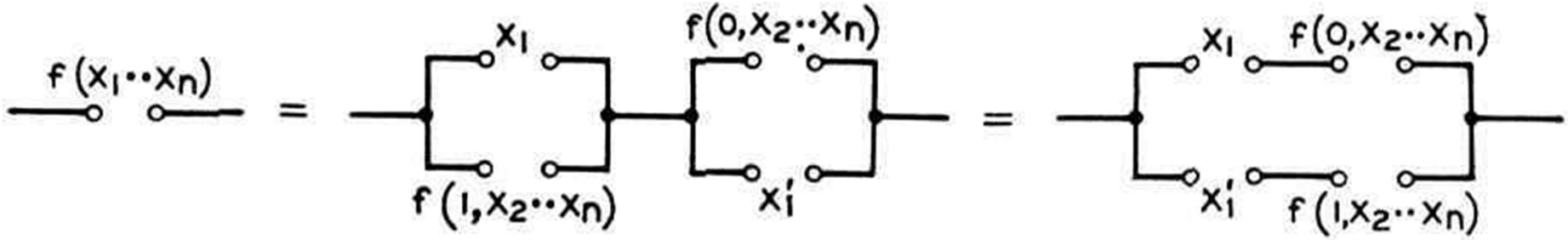


Figure 4. Expansion about one variable

Feasibility Check for Embedded System

Given an embedded system with a clock frequency of 100MHz, a memory size of 1Kbytes with read/write time of 10ns and a CPU speed on the average 3 cycles per operation, calculate the time needed to find median of a set of N, 16 bit positive numbers. An N-point median filter takes the present value and N-1 previous values and computes the median of N points and outputs the median value.

If the input to the system in the above question is a sequence of 16 bit signed voice samples, what is the maximum length of the median filter that can be implemented in real time?

Memory Hierarchy and speed

Memory System	Typical Capacity	Typical Access Time
Registers	1kB	1ns
Cache	1 MB	2 ns
Main Memory	8MB – 16GB	10ns
Disk drive	.1GB to 300GB	10ms

Haswell 4th Generation Processor

- Improved power-performance efficiency and power management. Power performance
- Improvements in power management include additional idle states, specifically the new active idle state S0ix, which enables 20x reduction in idle power.
- One key enabler for power-performance improvements is the fully integrated voltage regulator (FIVR), which also improves board space and cost.
- Haswell is built with an SoC design approach that allows fast and easy creation of derivatives and variations on the baseline.
- Graphics and media come with more scalability that lets designers build efficient configurations from the lowest to highest end.

Simple Analog Computer

Make an adder using resistors and operational amplifiers to add any two numbers in the range of 1 to 100? Make a diagram of the system which can be built and which works if implemented in the lab.

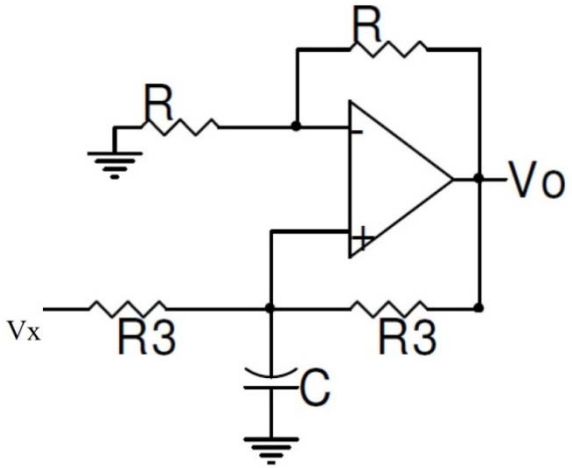
Make a circuit using Operational amplifier to generate the signal y (t) = Cos (t)

$$y(t) = Cos(t) \qquad \frac{dy(t)}{dt} = -Sin(t); y(0) = 1$$

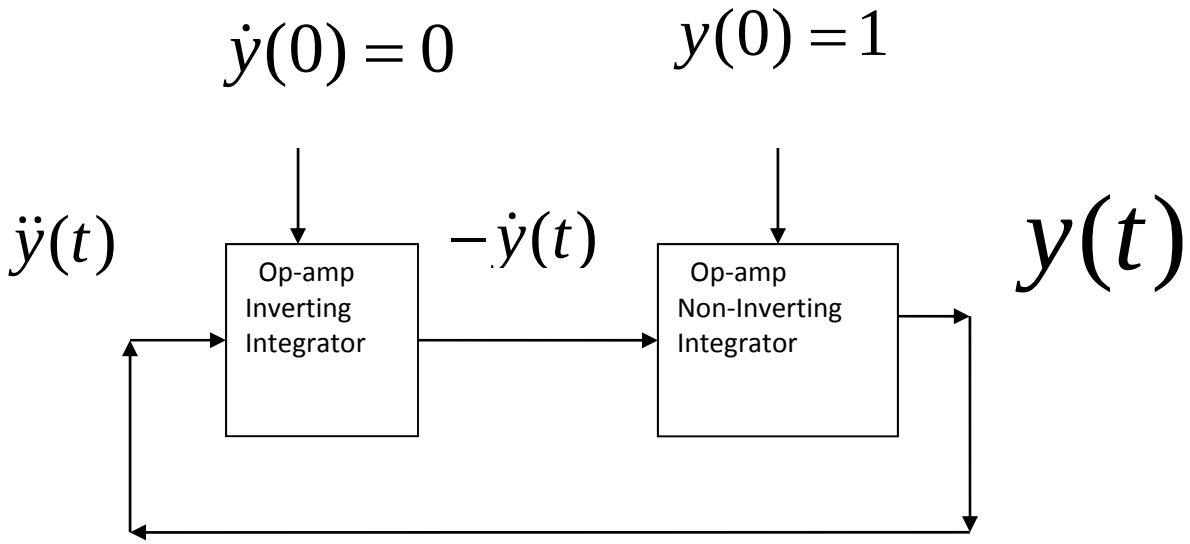
$$\ddot{y}(t) = -y(t)$$

$$\frac{d^2 y(t)}{dt^2} = -Cos(t); \dot{y}(0) = 0$$

Non-inverting Integrator



How to find Square Root and Inverse?



Note: Given any nth order differential equation we can solve it using such Op-amp circuits. Hope the idea is clear. Feel free to discuss in case of any doubts.

How to find Square Root and Inverse?

Given only addition, subtraction and multiplication give a method to find square root of number and inverse of a real number.

Newton Raphson Approach?

Newton Raphson iteration scheme for finding a root of the equation f(y) = 0 is

$y_n = y_{n-1} + \frac{f(y_{n-1})}{f'(y_{n-1})}$ with an initial value $y_0 = a$ chosen by the user.

Let A be the given number. We want to find 1/A.

For division we choose $f(y) = \frac{1}{y} - A$ whose root is 1/A.

$$f'(y) = \frac{-1}{y^2}$$

So Newton Raphson Iteration is given by,

$$y_n = y_{n-1} - \frac{(\frac{1}{y_{n-1}} - A)}{\frac{-1}{y_{n-1}^2}}$$

Or

$y_n = y_{n-1} + (y_{n-1} - Ay_{n-1}^2)$ Iteration scheme for division

Iterative Technique for Square Root

For square root we choose $f(y) = \frac{1}{y^2} - A$ where A is the given number whose square root we want,

$$f'(y) = \frac{-2}{y^3}$$

So the iteration scheme becomes

$$y_n = y_{n-1} + \frac{(y_{n-1} - A y_{n-1}^3)}{2}$$ Iteration scheme which gives inverse of the square root.

Threshold Switching Functions

Let x (1), x (2)… x (n) by binary variables. A function f(x(1),x(2),….x(n)) is called a threshold switching function iff,

$$f(x(1),x(2),\ldots x(n)) = 1 \text{ if } x(1)a(1)+x(2)a(2)+\ldots.. x(n)a(n) > T$$
$$= 0 \text{ if } x(1)a(1)+x(2)a(2)+\ldots.. x(n)a(n) <= T$$

Where a (1), a (2) … a (n) and T are real numbers called weights and threshold (T).

Show that threshold switching functions can be used to realize any combinational switching function.

Hint: Realize AND, OR and NOT gates using proper threshold functions.

Threshold Gate is a Universal Gate

AND Gate: Choose all weights as 1 and T = n – ½

OR Gate: Choose all weights as 1 and T = ½

NOT Gate: x (1) is the active input. All other inputs are set to zero. Take weight for x (1) as -1 and T as -1/2.

Note that x (1) a (1) +x (2) a (2) +….. x (n)a(n) = T is an equation in the n dimensional space which can separate two subsets which are linearly separable.

There may be more than one line which can do the job. Choose a line whose parameters are easy to implement and has enough margin for noise.

EXOR is not linearly separable. EXOR gate cannot be realized as one Threshold gate.

Majority Logic gate

Consider a majority switching circuit with n binary inputs and one binary output. Its output is one if more than n/2 of the inputs are 1 and zero otherwise. Show that one n input threshold function can implement a majority gate. Design a switching circuit to implement an n input majority switching circuit.

Counting Switching Functions

How many switching functions of n variables are possible? How many switching functions of n variables are there which actually involve all the variables? Note that f(x,y) = x+y is a function of both x and y.
While f(x,y) = x involves only x.

Shannon's Approach

It has been shown that any function may be expanded in a series consisting of a sum of products, each product being of the form $X_1 X_2 \ldots X_n$ with some permutation of primes on the letters, and each product having the coefficient 0 or 1. Now since each of the n variables may or may not have a prime, there is a total of 2^n different products of this form. Similarly each product may have the coefficient 0 or the coefficient 1 so there are 2^{2^n} possible sums of this sort. Hence we have the theorem: The number of functions obtainable from n variables is 2^{2^n}.

Functions Having Exactly n variables

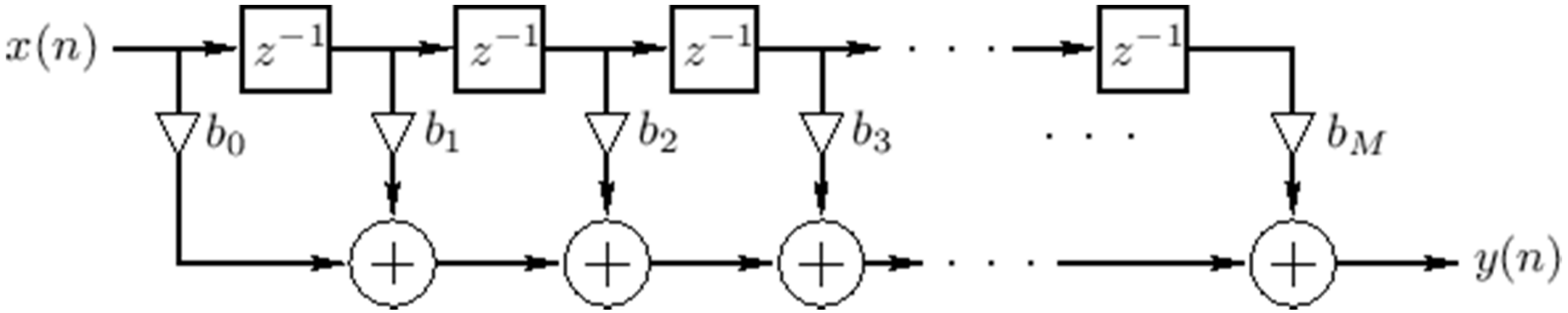
$$2^{2^n} = \sum_{k=0}^{n} \binom{n}{k} \phi(k) \ ,$$

$$\phi(n) = 2^{2^n} - \sum_{k=0}^{n-1} \binom{n}{k} \phi(k) \ .$$

How did Shannon Get this expression?

$$\phi(n) = \sum_{k=0}^{n} \binom{n}{k} 2^{2^k} (-1)^{n-k} .$$

A Simple Multiply and Add Structure (FIR Filtering)



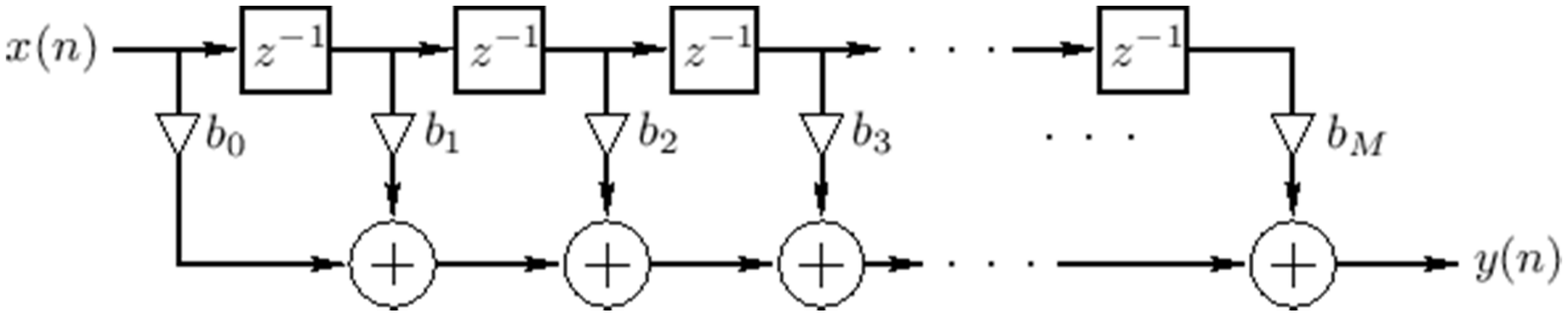
$$y(n) = \sum_{k=0}^{k=M} b_k x(n-k)$$

FIR Filtering: Normal way

```
 M+1 = Filter order
b [M] : Filter coefficients
Input [M] : Buffer for storing inputs. Initialised to zero.

For ( i = 0; i < Number of input samples; i++)
{

Y = 0;

Input[0] = read current input.

for (k = 0; k < M+1; k++)
{
if( i – k >=0) Y = Y+ b (k) * Input [i- k] ; // performing convolution
}
for (k = M; k > 0; k--){
Input [k]= Input[k-1]; // performing shifting operation
}

Output Y;

}
```

Circular Buffer and FIR Filtering

x(0), 0, 0, …… 0,0,0
x(0), x(1), 0……0,0,0
x(0),x(1),x(2)……0,0,0
….
….
….
x(0), x(1),x(2),……x(M)
x(M+1),x(1),x(2),x(3),……x(M)
….
…..



FIR Filtering with Circular Buffer

```
 M = Filter order
b [M] : Filter coefficients
Input [M] : Buffer for storing inputs. Initialised to zero.
Pointer = 0;
For ( i = 0; i < Number of input samples; i++)
{

Y = 0;

Input [Pointer] = read current input.
    // Put the newest sample into the Pointer location

for (k = 0; k < M+1; k++)
{
Y = Y+ b [k] * input [(Pointer - k) % (M+1)];
       // performing convolution between samples and corresponding
          coefficients
}
Pointer = (Pointer + 1) % (M+1);
Output Y;
}
```

An Embedded System Feasibility Check

Consider an embedded system which accepts two channel 16bit audio inputs whose frequencies are in the range of 300Hz to 17KHz and two camera inputs at a frame rate of 30 frames/seconds and 320x240 frame size with each pixel having R,G and B components having 8 bit resolution. Calculate the amount of memory needed to store 1 second input data. Embedded system does FIR filtering on audio channels with 64 coefficient FIR filter and 9 sample median filtering on the image. If each operation takes 1 clock cycle of the embedded system which runs with a clock speed of 600MHz does the system work in real time? Median filtering is done for each pixel, say p(x), by considering its 3x3 neighbors. Arrange 9 pixels including p(x) in ascending order and replace p(x) by the middle value. Such filters are useful to reduce salt and pepper noise. Assume memory access time is also 1 clock cycle.

How to find Integer Roots of a Quadratic Equation?

Give a flow chart for finding integer roots of a quadratic equation with 32 bit integer coefficients. If integer roots don't exist a message must be printed out.

aX^2 + bX + c = 0
Let the roots be a1 and a2.
If D= b*b-4ac >=0 and D is a perfect square then,

a1= (-b + sqrt(D)/2a
a2 = (-b-sqrt(D)/2a

Pentium Bug

Statistical Analysis of Floating Point Flaw in the Pentium Processor (1994) **H. P. Sharangpani, M. L. Barton, Ph.D.**
Intel Corporation November 30th 1994

The paper described in broad outline the nature of the bug, and estimated that the average user would encounter the error once every 27,000 years.

A subtle flaw in the hardware divide unit of the Pentium Processor was discovered by Intel. Subsequently, a characterization of its impact to the end-user application base was conducted. The flaw is rare and data-dependent, and causes a reduction in precision of the divide instruction and certain other operations in certain cases.

Pentium Bug

Statistical Analysis of Floating Point Flaw in the Pentium Processor (1994) **H. P. Sharangpani, M. L. Barton, Ph.D.**
Intel Corporation
November 30th 1994

The significance of the flaw depends upon

(a) the rate of use of specific FP instructions in the Pentium CPU,

(b) the data fed to them,

(c) the way in which the results of these instructions are propagated into further computation in the application; and

(d) the way in which the final results of the application are interpreted.

Pentium Bug

Statistical Analysis of Floating Point Flaw in the Pentium Processor (1994) **H. P. Sharangpani, M. L. Barton, Ph.D.**
Intel Corporation
November 30th 1994

The thorough and detailed characterization of the flaw and the subsequent investigations of its impact on applications through elaborate surveys, analyses and empirical observation lead us to the overall conclusion that the flaw is of no concern to the vast majority of users of Pentium processor based systems. A few users of applications in the scientific/engineering and financial engineering fields who require unusual precision and invoke millions of divides per day may need to employ either an updated Pentium processor without the flaw or a software workaround.

The SRT(Sweeney, Robertson & Tocher) divide algorithm

1. Sample the most significant digits of the divisor and dividend.

2. Use the samples as indexes into a lookup table to get a guess of the next quotient digits. (in this case the quotient guess can be -2, -1, 0, +1, +2).

3. Multiply the divisor by the quotient guess and subtract it from dividend (Note that this is an addition if the quotient guess was negative).

4. Save the quotient digits in the least significant digits of a quotient register.

5. Shift the remainder left by 2 and shift the quotient registers left by 2 (i.e. radix 4)

6. Sample the most significant digits of the new shifted partial remainder.

7. Go to step 2 unless you have generated enough significant quotient digits.

8. Generate the binary quotient by assembling the values in the quotient register.

9. If the last partial remainder was negative then adjust the quotient by subtracting the value 1.

Embedded System Models

“The software that directly interfaces with and controls this hardware is called a device driver.

All embedded systems that require software have, at the very least, device driver software in their system software layer.

Device drivers are the software libraries that initialize the hardware and manage access to the hardware by higher layers of software.

Device drivers are the liaison between the hardware and the operating system, middleware, and application layers.”

Embedded Systems Architecture, 2nd Edition A Comprehensive Guide for Engineers and Programmers T. Noergaard, 2012

Embedded System Models