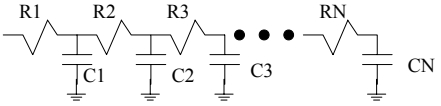
Scripting with Perl

- 'Scripting' usually means to control the execution of one or more other programs via some external program
  - I.e, a Perl program that executes some external sequence of programs is 'scripting' the actions of those programs
- Why script the execution of other programs? There are many reasons – a couple are:
  - Execute another program or sequence of programs with different parameter sets
  - Do regression testing in which you test the program with a set of input values, and compare the output against a 'golden' output

A Sample Application

Controlling wire delays in integrated circuits is important because as transistors have gotten smaller, global wire delays have become significant.

A wire in an integrated circuit has both resistance and capacitance and can be viewed as a distributed RC network:

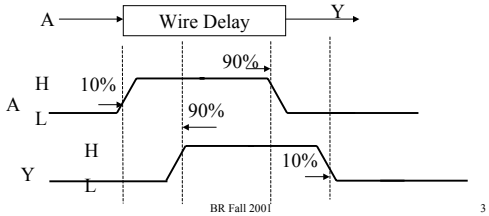


The resistances/capacitances are the Rs/Cs of different segments of the wire and are not equal to each other.

Estimating Wire Delay

One estimate of the wire delay is $0.9 * R_{tot} * C_{tot}$ where R_{tot} and C_{tot} is the lumped Resistance/Capacitance of the wire (the sum of all of the R's, C's).

This estimation is the 10% to 90% delay of the wire – the delay measured from the 10% point on the input waveform to the 90% point on the output waveform

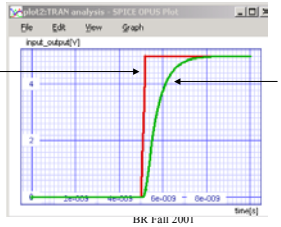


Simulating Wire Delay

Usually, the topology of a wire is more complex in that it will have multiple endpoints.

Computation of the delays from the driver to multiple endpoints may require running an external program such as SPICE to find the delays.

Input waveform

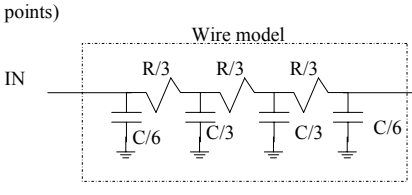


Output waveform

A Problem

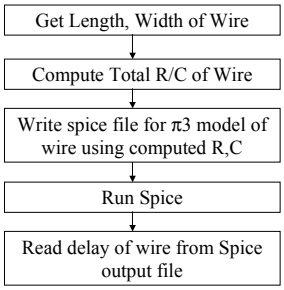
Given a wire length and width, use Perl to run Spice to simulate the delay of the wire.

We will assume a simple wire, and use a π3 model for simulation. In reality, simulation is only necessary for complex wires (complex topology with multiple branching segments and end points)



R, C is total resistance, capacitance of wire.

Perl Script Flow



wiredelay.pl script – top level

Process values needed to compute wire R/C

```
#!/usr/bin/perl -w
use English;

## parameters for wire from TSMC 0.18u process, M4 wire,
## these are typical

$substrate_cap = 8.0e-18;  # af/um**2 (atto farad is  1.0e=-18)
$fringe_cap = 23.0e-18;  # af/um (atto farad is  1.0e=-18)
$resistance = 0.08;  # ohms per square

if (@ARGV < 1) {
 print("Usage: $PROGRAM_NAME  wire_length wire_width \n");
 print("Will calculate wire delay based upon spice simulation \n");
 print("Wire_length, Wire_width in microns \n");
 exit(0);
}
$w_len = $ARGV[0];
$w_wid = $ARGV[1];

$delay = &get_wiredelay($w_len,$w_wid);
print "Simulated 10%-90% Wire Delay: $delay\n";
exit(0);
```

Subroutine call that does all the work

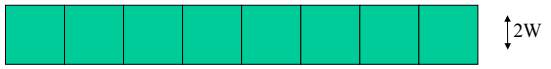
Resistance Computation of a Wire

The resistance of wire is dependent on length, width.
Resistance is specified by ohms/square (R_{sq} =ohms per square)

Compute the number of squares by dividing Length by Width.

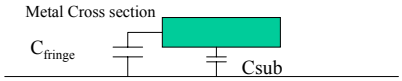
R wire = R_{sq} * L / W

What if we double the wire width and keep the same L?



R wire_new = R_{sq} * L / 2W = R wire old/2

Capacitance Computation of a Wire



Cfringe is the capacitance of the side of the wire to the substrate - this is specified in farads per micron.

Csub is the capacitance of the bottom of the wire to the substrate. This is specified in farads per square micron.

Cwire = Cfringe *Length + Csub * Length *Width

get_wiredelay subroutine

```
sub get_wiredelay
{
    ($w_len,$w_wid) = @_;
    my($total_cap,$total_res);  # local variables
    my(@words);
    my($this_delay);
# compute capacitance
    $total_cap = $w_len * $fringe_cap + w_len*$w_wid*$substrate_cap;
    $total_res = $w_len/$w_wid * $resistance;
    $est_delay = $total_cap*$total_res*0.9;
    print ("Total Cap: $total_cap, Total Res: $total_res\n");
    print ("Estimated 10%-90% Wire Delay: $est_delay\n");
```

Subroutine accepts length, width of wire as parameters.

Computes total capacitance, total resistance. Also calculates estimated wire delay.

Spice Opus

SPICE is a circuit simulator originally developed by University of California, Berkeley.

Do not worry if you have never heard of SPICE - I provide the simulation files, and we will treat it as a black box that will output a wire delay given an input simulation file.

For Spice simulation, will use an enhanced version called SPICE Opus that runs under Win32.

Your CDROM has a zip file under the 'misc' directory called *spice_opus203.zip* . Unzip this into a temporary directory and it will create four temporary directories called 'disk1', 'disk2', 'disk3', 'disk4'. Execute 'setup.exe' in the 'disk1' directory in order to install SPICE Opus. The default installation directory is C:\SpiceOpus .

Wire Delay Simulation in SPICE Opus

The file *wiresim.sp* is the Spice input file for the wire delay simulation.

This file includes a second file called *pi3net.sp* that actually determines the topology and R/C values of the wire model to be used.

The *get_wiredelay* subroutine in the *wiredelay.pl* script generates the *pi3net.sp* file with the needed R/C components required for the π3 model in Spice format. The R/C values are based upon the total R,C values previously computed.

get_wiredelay subroutine – creating the *pi3net.sp* file

```
#create input file for spice, use the PI3 model.
    open(OUTPUT,">pi3net.sp");
    print OUTPUT ("*PI3 Network\n");
    print OUTPUT ("*Resistors\n");
    printf OUTPUT ("r1 a  c %e\n",$total_res/3);
    printf OUTPUT ("r2 c  d %e\n",$total_res/3);
    printf OUTPUT ("r3 d  b %e\n",$total_res/3);
    print OUTPUT ("*Capacitors\n");
    printf OUTPUT ("c1 a  0 %e\n",$total_cap/6);
    printf OUTPUT ("c2 c  0 %e\n",$total_cap/3);
    printf OUTPUT ("c3 d  0 %e\n",$total_cap/3);
    printf OUTPUT ("c4 b  0 %e\n",$total_cap/6);
    printf OUTPUT ("\n");
```

These statements written to the pi3net.sp file create the needed RC network in SPICE format to simulate the wire delay via the π3 model .

Running Spice Opus

To run Spice Opus, can do the following from a command window:

C:\spiceopus\bin\spice3.exe

This will open up a Spice Opus command window, can then type in a spice file name to execute.

Returned delay time

A plot window also is opened to display the input/output waveforms.

get_wiredelay subroutine – running SPICE Opus

We need to run SPICE Opus in batch mode and redirect its output to a file so that we can parse the delay time:

```
### run spice
    $cmd = 'c:\spiceopus\bin\spice3.exe -o tmp.dat -b wiresim.sp';
    `$cmd`;
```

The –o option redirects the output to a file

The –b option runs the the spice file in batch mode.

Backquotes used to execute this command from the perl script.

get_wiredelay subroutine – output Parsing

Need to parse the output file for the delay value. Delay is printed as "Rise_time=*delay_value*

```
### get wire delay
    open (INPUT,"tmp.dat");
    while (<INPUT>){
        chop;
        if (/^Rise_time(.*)/) {
            @words=split '=';
            $this_delay = $words[1];
            last;
        }
    }
    close(INPUT);
    return($this_delay);
}
```

Find line that starts with 'Rise_time'

Split line on '=', second word is delay time

Return delay value to main program.

Running *getwiredelay.pl*

16 mm wire length, 0.4u wide:

```
H:\ece3732\spice>perl wiredelay.pl 16000 0.4
Total Cap: 4.192e-013  Total Res:3200
Estimated 10%-90% Wire Delay: 1.207296e-009
Simulated 10%-90% Wire Delay: 1.48194E-009
```

16 mm wire length, 0.8u wide:

```
H:\ece3732\spice>perl wiredelay.pl 16000 0.8
Total Cap: 4.704e-013  Total Res:1600
Estimated 10%-90% Wire Delay: 6.77376e-010
Simulated 10%-90% Wire Delay: 8.69914E-010
```

Noticed that wire delay decreased when wire width doubled – Resistance decreased by two.

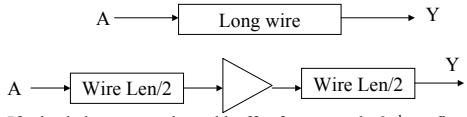
Substrate capacitance increases by factor of two, but fringing cap remains the same and total wire capacitance is dominated by fringing cap.

Reducing Wire Delay of Long Wires

Can reduce wire delay by making wire wider if fringing capacitance dominates wire capacitance.

If substrate capacitance dominates, then increasing width decreases R, but also increases C by same amount so total RC remains the same.

Can also reduce wire delay by splitting a wire into segments and putting a buffer between each segments.



If wire is long enough, and buffer fast enough, 2nd configuration will have less delay.

Extending *wiredelay.pl*

- Could read an external data file that would specify the topology of a complex wire as well as R,C values of different segments
  - This data file produced by another program that extracts this information from the mask data of the integrated circuit
- Add capability to *wiredelay.pl* to experiment with buffer placements to reduce delay