Repeater Insertion in Tree Structured Inductive Interconnect: Software Description

Yehea I. Ismail
ECE Department
Northwestern University
Evanston, IL 60208

Eby G. Friedman
ECE Department
University of Rochester
Rochester, New York 14627

This document provides a detailed description of the software tool for repeater insertion targeting both users and programmers. Section I describes how to run the program. Section II describes the input file format while section III describes the output file format. The variables and parameters used within the program to fine tune the repeater insertion process are described in section IV. The functions used within the program are described is section V. Note that many comments have been added in the source code that describe the operation of the program.

I. Running the Program

The C source code is called buf8.c. This source code can be compiled on any machine such as a PC, sun, or IBM machine. Once compiled, the program can be invoked using the command-line command:

buf8 <inputfile> > <outputfile>

where inputfile is a text file containing the interconnect description. The outputfile is a file where the results and the AS/X file will be written to. This AS/X file can be used to accurately simulate the resulting optimum circuit generating the delays and power consumption of the repeater system.

II. Input File Format

An example of an input file is listed below:

```
root N1 3  3000
N1   N2 2  4000
N1   N3 4  2000
N2   N4 1  6000
N2   N5 6  2000
N3   N6 1  3000
N3   N7 4  4000
N4   N8 7  6000
N4   N9 4  4000
N5   N10 1  2000
N5   N11 2  1000
```

```
N6   N12 6  4000
N6   N13 4  2000
N7   N14 3  4000
N7   N15 2  4000
N8   N16 2  4000
N8   N17 5  2000
N9   N18 2  4000
N9   N19 3  5000
N10   N20 1  4000
N10   N21 2  4000
N11   N22 2  4000
N11   N23 2  4000
N12   N24 4  6000
N12   N25 2  4000
N13   N26 3  2000
N13   N27 2  4000
N14   N28 1  4000
N14   N29 2  4000
N15   N30 2  4000
N15   N31 4  3000
```

For the specific technology used (and for most typical technologies), the wires cannot just take any width. There are only few wire widths available. For each wire width w, the capacitance inductance and resistance per unit length are given by C, L, and R, respectively. The global array of pwire structures wt described in section IV includes the information for the typically use wires. Using this technique of precharacterizing the available wire types eliminates the extraction step and significantly simplifies the input file format.

Referring to the input file listing above, the first two strings are the names of the nodes between which the wire is connected. The third number is the type of the wire corresponding to one of the wires in the structure wt. For example, a 1 in the third column means that the wire has a width of 0.9 micrometer and has the impedance values in the first entry of wt in section IV. The fourth entry is the length of the wire in micrometers. Note that each wire spans a separate line and that the source node of the tree (where the input voltage is connected) has to be called “root”.

III. Output File Format

A sample output file is listed below:

```
*iterations = 6


*
*-------------------------------------------------------------
*Sink Delays are (Left to Right):
*
*0.422775
*0.418599
*0.410650
*0.414385
```

```
*0.422739
*0.412066
*0.403790
*0.403790
*0.411279
*0.402124
*0.415111
*0.421998
*0.419892
*0.410730
*0.415662
*0.413019
*
*------------------------------------------------------------
*Cost = 0.422775
*------------------------------------------------------------
*
*Optimum Buffers are (root towards leafs, left then right):
*
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  173.742401  0.000000
0.000000
*0.000000  300.899994  600.799988  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*300.899994  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*71.536476  300.899994  600.799988  0.000000  0.000000  0.000000  0.000000
300.899994
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
300.899994
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  180.460159
*480.839966  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  600.799988  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  120.959999  300.899994  0.000000  0.000000
0.000000
```

```
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*360.880005  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000
*-------------------------------------------------------------
*
*Buffer Area = 4996.218262
*
*Run Time = 72622

MODEL DESCRIPTION

MODEL LINE(IN-OUT)
ELEMENTS
PR=1
PL=1
PC=1
R, IN-NN=(PR)
L, NN-OUT=(PL)
C, OUT-0=(PC)

FEATURES
GLOBAL=(N0)


MODEL LINE_BUFFER(IN-OUT-VDD)
ELEMENTS
PR=1
PL=1
PC=1
Ph=1
WN=(0.25*Ph)
WP=(2.12*WN)
MP1=MODEL Pfet(IN-VDD-VDD-N1)(PW=WP,PL=0.25)
MN1=MODEL Nfet(IN-0-0-N1)(PW=WN,PL=0.25)
R, N1-N2=(PR)
L, N2-OUT=(PL)
C, OUT-0=(PC)

FEATURES
GLOBAL=(N0)


MODEL MAIN()
ELEMENTS
```

```
PORDER$ = (A.$)
DUM = MODEL GLOBDIST () (DROPREF)
$ = MODEL PROCESS ()

E_VDD,0-VDD=2.5
E_VP,0-N_0=(PULSE(0,2.5,0,0.03,0.03,10,40))

P_SIM=(AREA(IE_VDD,0,20)*2.5+AREA(IE_VP,0,9)*2.5)



L0=MODEL LINE(N_0-N_1)(PR=0.001875,PL=0.000273,PC=0.126700)
L1=MODEL LINE(N_1-N_2)(PR=0.001875,PL=0.000273,PC=0.126700)
L2=MODEL LINE(N_2-N_3)(PR=0.001875,PL=0.000273,PC=0.126700)
L3=MODEL LINE(N_3-N_4)(PR=0.001875,PL=0.000273,PC=0.126700)
L4=MODEL LINE(N_4-N_5)(PR=0.001875,PL=0.000273,PC=0.126700)
L5=MODEL LINE(N_5-N_6)(PR=0.001875,PL=0.000273,PC=0.126700)
L6=MODEL LINE(N_6-N_7)(PR=0.002720,PL=0.000288,PC=0.107400)
L7=MODEL LINE(N_7-N_8)(PR=0.002720,PL=0.000288,PC=0.107400)
L8=MODEL LINE(N_8-N_9)(PR=0.002720,PL=0.000288,PC=0.107400)
L9=MODEL LINE(N_9-N_10)(PR=0.002720,PL=0.000288,PC=0.107400)
L10=MODEL LINE(N_10-N_11)(PR=0.002720,PL=0.000288,PC=0.107400)
L11=MODEL LINE(N_11-N_12)(PR=0.002720,PL=0.000288,PC=0.107400)
L12=MODEL LINE(N_12-N_13)(PR=0.002720,PL=0.000288,PC=0.107400)
L13=MODEL LINE(N_13-N_14)(PR=0.002720,PL=0.000288,PC=0.107400)
L14=MODEL LINE(N_14-N_15)(PR=0.011150,PL=0.000319,PC=0.106550)
L15=MODEL LINE(N_15-N_16)(PR=0.011150,PL=0.000319,PC=0.106550)
L16=MODEL LINE(N_16-N_17)(PR=0.011150,PL=0.000319,PC=0.106550)
L17=MODEL LINE(N_17-N_18)(PR=0.011150,PL=0.000319,PC=0.106550)
L18=MODEL LINE(N_18-N_19)(PR=0.011150,PL=0.000319,PC=0.106550)
L19=MODEL LINE(N_19-N_20)(PR=0.011150,PL=0.000319,PC=0.106550)
L20=MODEL LINE(N_20-N_21)(PR=0.011150,PL=0.000319,PC=0.106550)
L21=MODEL LINE_BUFFER(N_21-N_22-VDD)(PR=0.011150,PL=0.000319,PC=0.106550,
Ph=173.742401)
L22=MODEL LINE(N_22-N_23)(PR=0.011150,PL=0.000319,PC=0.106550)
L23=MODEL LINE(N_23-N_24)(PR=0.011150,PL=0.000319,PC=0.106550)
L24=MODEL LINE(N_24-N_25)(PR=0.011150,PL=0.000319,PC=0.106550)
L25=MODEL LINE_BUFFER(N_25-N_26-VDD)(PR=0.011150,PL=0.000319,PC=0.106550,
Ph=300.899994)
L26=MODEL LINE_BUFFER(N_26-N_27-VDD)(PR=0.000305,PL=0.000170,PC=0.445000,
Ph=600.799988)
L27=MODEL LINE(N_27-N_28)(PR=0.000305,PL=0.000170,PC=0.445000)
L28=MODEL LINE(N_28-N_29)(PR=0.000305,PL=0.000170,PC=0.445000)
L29=MODEL LINE(N_29-N_30)(PR=0.000305,PL=0.000170,PC=0.445000)
L30=MODEL LINE(N_30-N_31)(PR=0.000305,PL=0.000170,PC=0.445000)
L31=MODEL LINE(N_31-N_32)(PR=0.000305,PL=0.000170,PC=0.445000)
L32=MODEL LINE(N_32-N_33)(PR=0.000305,PL=0.000170,PC=0.445000)
L33=MODEL LINE(N_33-N_34)(PR=0.000305,PL=0.000170,PC=0.445000)
L34=MODEL LINE(N_34-N_35)(PR=0.000305,PL=0.000170,PC=0.445000)
L35=MODEL LINE(N_35-N_36)(PR=0.000305,PL=0.000170,PC=0.445000)
L36=MODEL LINE(N_36-N_37)(PR=0.000305,PL=0.000170,PC=0.445000)
L37=MODEL LINE(N_37-N_38)(PR=0.000305,PL=0.000170,PC=0.445000)
L38=MODEL LINE(N_38-N_39)(PR=0.002720,PL=0.000288,PC=0.107400)
L39=MODEL LINE(N_39-N_40)(PR=0.002720,PL=0.000288,PC=0.107400)
L40=MODEL LINE(N_40-N_41)(PR=0.002720,PL=0.000288,PC=0.107400)
L41=MODEL LINE(N_41-N_42)(PR=0.002720,PL=0.000288,PC=0.107400)
L42=MODEL LINE(N_42-N_43)(PR=0.002720,PL=0.000288,PC=0.107400)
```

```
L43=MODEL LINE(N_43-N_44)(PR=0.002720,PL=0.000288,PC=0.107400)
L44=MODEL LINE(N_44-N_45)(PR=0.002720,PL=0.000288,PC=0.107400)
L45=MODEL LINE(N_45-N_46)(PR=0.002720,PL=0.000288,PC=0.107400)
L46=MODEL LINE(N_38-N_47)(PR=0.001015,PL=0.000231,PC=0.179500)
L47=MODEL LINE(N_47-N_48)(PR=0.001015,PL=0.000231,PC=0.179500)
L48=MODEL LINE(N_48-N_49)(PR=0.001015,PL=0.000231,PC=0.179500)
L49=MODEL LINE(N_49-N_50)(PR=0.001015,PL=0.000231,PC=0.179500)
L50=MODEL LINE_BUFFER(N_26-N_51-VDD)(PR=0.001250,PL=0.000249,PC=0.157000,
Ph=300.899994)
L51=MODEL LINE(N_51-N_52)(PR=0.001250,PL=0.000249,PC=0.157000)
L52=MODEL LINE(N_52-N_53)(PR=0.001250,PL=0.000249,PC=0.157000)
L53=MODEL LINE(N_53-N_54)(PR=0.001250,PL=0.000249,PC=0.157000)
L54=MODEL LINE(N_54-N_55)(PR=0.001250,PL=0.000249,PC=0.157000)
L55=MODEL LINE(N_55-N_56)(PR=0.001250,PL=0.000249,PC=0.157000)
L56=MODEL LINE(N_56-N_57)(PR=0.001250,PL=0.000249,PC=0.157000)
L57=MODEL LINE(N_57-N_58)(PR=0.001250,PL=0.000249,PC=0.157000)
L58=MODEL LINE(N_58-N_59)(PR=0.002720,PL=0.000288,PC=0.107400)
L59=MODEL LINE(N_59-N_60)(PR=0.002720,PL=0.000288,PC=0.107400)
L60=MODEL LINE(N_60-N_61)(PR=0.002720,PL=0.000288,PC=0.107400)
L61=MODEL LINE(N_61-N_62)(PR=0.002720,PL=0.000288,PC=0.107400)
L62=MODEL LINE(N_62-N_63)(PR=0.002720,PL=0.000288,PC=0.107400)
L63=MODEL LINE(N_63-N_64)(PR=0.002720,PL=0.000288,PC=0.107400)
L64=MODEL LINE(N_64-N_65)(PR=0.002720,PL=0.000288,PC=0.107400)
L65=MODEL LINE(N_65-N_66)(PR=0.002720,PL=0.000288,PC=0.107400)
L66=MODEL LINE(N_58-N_67)(PR=0.001875,PL=0.000273,PC=0.126700)
L67=MODEL LINE(N_67-N_68)(PR=0.001875,PL=0.000273,PC=0.126700)
L68=MODEL LINE(N_68-N_69)(PR=0.001875,PL=0.000273,PC=0.126700)
L69=MODEL LINE(N_69-N_70)(PR=0.001875,PL=0.000273,PC=0.126700)
L70=MODEL LINE(N_70-N_71)(PR=0.001875,PL=0.000273,PC=0.126700)
L71=MODEL LINE(N_71-N_72)(PR=0.001875,PL=0.000273,PC=0.126700)
L72=MODEL LINE(N_72-N_73)(PR=0.001875,PL=0.000273,PC=0.126700)
L73=MODEL LINE(N_73-N_74)(PR=0.001875,PL=0.000273,PC=0.126700)
L74=MODEL LINE(N_74-N_75)(PR=0.001875,PL=0.000273,PC=0.126700)
L75=MODEL LINE(N_75-N_76)(PR=0.001875,PL=0.000273,PC=0.126700)
L76=MODEL LINE_BUFFER(N_14-N_77-VDD)(PR=0.000402,PL=0.000170,PC=0.356000,
Ph=71.536476)
L77=MODEL LINE_BUFFER(N_77-N_78-VDD)(PR=0.000402,PL=0.000170,PC=0.356000,
Ph=300.899994)
L78=MODEL LINE_BUFFER(N_78-N_79-VDD)(PR=0.000402,PL=0.000170,PC=0.356000,
Ph=600.799988)
L79=MODEL LINE(N_79-N_80)(PR=0.000402,PL=0.000170,PC=0.356000)
L80=MODEL LINE(N_80-N_81)(PR=0.011150,PL=0.000319,PC=0.106550)
L81=MODEL LINE(N_81-N_82)(PR=0.011150,PL=0.000319,PC=0.106550)
L82=MODEL LINE(N_82-N_83)(PR=0.011150,PL=0.000319,PC=0.106550)
L83=MODEL LINE_BUFFER(N_83-N_84-VDD)(PR=0.011150,PL=0.000319,PC=0.106550,
Ph=300.899994)
L84=MODEL LINE(N_84-N_85)(PR=0.011150,PL=0.000319,PC=0.106550)
L85=MODEL LINE(N_85-N_86)(PR=0.011150,PL=0.000319,PC=0.106550)
L86=MODEL LINE(N_86-N_87)(PR=0.011150,PL=0.000319,PC=0.106550)
L87=MODEL LINE(N_87-N_88)(PR=0.011150,PL=0.000319,PC=0.106550)
L88=MODEL LINE(N_88-N_89)(PR=0.011150,PL=0.000319,PC=0.106550)
L89=MODEL LINE(N_89-N_90)(PR=0.011150,PL=0.000319,PC=0.106550)
L90=MODEL LINE(N_90-N_91)(PR=0.011150,PL=0.000319,PC=0.106550)
L91=MODEL LINE(N_91-N_92)(PR=0.011150,PL=0.000319,PC=0.106550)
L92=MODEL LINE(N_84-N_93)(PR=0.002720,PL=0.000288,PC=0.107400)
L93=MODEL LINE(N_93-N_94)(PR=0.002720,PL=0.000288,PC=0.107400)
L94=MODEL LINE(N_94-N_95)(PR=0.002720,PL=0.000288,PC=0.107400)
```

```
L95=MODEL LINE(N_95-N_96)(PR=0.002720,PL=0.000288,PC=0.107400)
L96=MODEL LINE(N_96-N_97)(PR=0.002720,PL=0.000288,PC=0.107400)
L97=MODEL LINE(N_97-N_98)(PR=0.002720,PL=0.000288,PC=0.107400)
L98=MODEL LINE(N_98-N_99)(PR=0.002720,PL=0.000288,PC=0.107400)
L99=MODEL LINE(N_99-N_100)(PR=0.002720,PL=0.000288,PC=0.107400)
L100=MODEL LINE_BUFFER(N_80-N_101-VDD)(PR=0.002720,PL=0.000288,PC=0.107400,
Ph=300.899994)
L101=MODEL LINE(N_101-N_102)(PR=0.002720,PL=0.000288,PC=0.107400)
L102=MODEL LINE(N_102-N_103)(PR=0.002720,PL=0.000288,PC=0.107400)
L103=MODEL LINE(N_103-N_104)(PR=0.002720,PL=0.000288,PC=0.107400)
L104=MODEL LINE(N_104-N_105)(PR=0.002720,PL=0.000288,PC=0.107400)
L105=MODEL LINE(N_105-N_106)(PR=0.002720,PL=0.000288,PC=0.107400)
L106=MODEL LINE(N_106-N_107)(PR=0.002720,PL=0.000288,PC=0.107400)
L107=MODEL LINE(N_107-N_108)(PR=0.002720,PL=0.000288,PC=0.107400)
L108=MODEL LINE(N_108-N_109)(PR=0.002720,PL=0.000288,PC=0.107400)
L109=MODEL LINE(N_109-N_110)(PR=0.002720,PL=0.000288,PC=0.107400)
L110=MODEL LINE(N_102-N_111)(PR=0.002720,PL=0.000288,PC=0.107400)
L111=MODEL LINE(N_111-N_112)(PR=0.002720,PL=0.000288,PC=0.107400)
L112=MODEL LINE(N_112-N_113)(PR=0.002720,PL=0.000288,PC=0.107400)
L113=MODEL LINE(N_113-N_114)(PR=0.002720,PL=0.000288,PC=0.107400)
L114=MODEL LINE(N_114-N_115)(PR=0.002720,PL=0.000288,PC=0.107400)
L115=MODEL LINE(N_115-N_116)(PR=0.002720,PL=0.000288,PC=0.107400)
L116=MODEL LINE(N_116-N_117)(PR=0.002720,PL=0.000288,PC=0.107400)
L117=MODEL LINE(N_117-N_118)(PR=0.002720,PL=0.000288,PC=0.107400)
L118=MODEL LINE_BUFFER(N_6-N_119-VDD)(PR=0.001250,PL=0.000249,PC=0.157000,
Ph=180.460159)
L119=MODEL LINE_BUFFER(N_119-N_120-VDD)(PR=0.001250,PL=0.000249,PC=0.157000,
Ph=480.839966)
L120=MODEL LINE(N_120-N_121)(PR=0.001250,PL=0.000249,PC=0.157000)
L121=MODEL LINE(N_121-N_122)(PR=0.001250,PL=0.000249,PC=0.157000)
L122=MODEL LINE(N_122-N_123)(PR=0.011150,PL=0.000319,PC=0.106550)
L123=MODEL LINE(N_123-N_124)(PR=0.011150,PL=0.000319,PC=0.106550)
L124=MODEL LINE(N_124-N_125)(PR=0.011150,PL=0.000319,PC=0.106550)
L125=MODEL LINE(N_125-N_126)(PR=0.011150,PL=0.000319,PC=0.106550)
L126=MODEL LINE(N_126-N_127)(PR=0.011150,PL=0.000319,PC=0.106550)
L127=MODEL LINE(N_127-N_128)(PR=0.011150,PL=0.000319,PC=0.106550)
L128=MODEL LINE_BUFFER(N_128-N_129-VDD)(PR=0.000402,PL=0.000170,PC=0.356000,
Ph=600.799988)
L129=MODEL LINE(N_129-N_130)(PR=0.000402,PL=0.000170,PC=0.356000)
L130=MODEL LINE(N_130-N_131)(PR=0.000402,PL=0.000170,PC=0.356000)
L131=MODEL LINE(N_131-N_132)(PR=0.000402,PL=0.000170,PC=0.356000)
L132=MODEL LINE(N_132-N_133)(PR=0.000402,PL=0.000170,PC=0.356000)
L133=MODEL LINE(N_133-N_134)(PR=0.000402,PL=0.000170,PC=0.356000)
L134=MODEL LINE(N_134-N_135)(PR=0.000402,PL=0.000170,PC=0.356000)
L135=MODEL LINE(N_135-N_136)(PR=0.000402,PL=0.000170,PC=0.356000)
L136=MODEL LINE(N_136-N_137)(PR=0.001250,PL=0.000249,PC=0.157000)
L137=MODEL LINE(N_137-N_138)(PR=0.001250,PL=0.000249,PC=0.157000)
L138=MODEL LINE(N_138-N_139)(PR=0.001250,PL=0.000249,PC=0.157000)
L139=MODEL LINE(N_139-N_140)(PR=0.001250,PL=0.000249,PC=0.157000)
L140=MODEL LINE(N_140-N_141)(PR=0.001250,PL=0.000249,PC=0.157000)
L141=MODEL LINE(N_141-N_142)(PR=0.001250,PL=0.000249,PC=0.157000)
L142=MODEL LINE(N_142-N_143)(PR=0.001250,PL=0.000249,PC=0.157000)
L143=MODEL LINE(N_143-N_144)(PR=0.001250,PL=0.000249,PC=0.157000)
L144=MODEL LINE(N_144-N_145)(PR=0.001250,PL=0.000249,PC=0.157000)
L145=MODEL LINE(N_145-N_146)(PR=0.001250,PL=0.000249,PC=0.157000)
L146=MODEL LINE(N_146-N_147)(PR=0.001250,PL=0.000249,PC=0.157000)
L147=MODEL LINE(N_147-N_148)(PR=0.001250,PL=0.000249,PC=0.157000)
```

```
L148=MODEL LINE(N_136-N_149)(PR=0.002720,PL=0.000288,PC=0.107400)
L149=MODEL LINE(N_149-N_150)(PR=0.002720,PL=0.000288,PC=0.107400)
L150=MODEL LINE(N_150-N_151)(PR=0.002720,PL=0.000288,PC=0.107400)
L151=MODEL LINE(N_151-N_152)(PR=0.002720,PL=0.000288,PC=0.107400)
L152=MODEL LINE(N_152-N_153)(PR=0.002720,PL=0.000288,PC=0.107400)
L153=MODEL LINE(N_153-N_154)(PR=0.002720,PL=0.000288,PC=0.107400)
L154=MODEL LINE(N_154-N_155)(PR=0.002720,PL=0.000288,PC=0.107400)
L155=MODEL LINE(N_155-N_156)(PR=0.002720,PL=0.000288,PC=0.107400)
L156=MODEL LINE_BUFFER(N_128-N_157-VDD)(PR=0.001250,PL=0.000249,PC=0.157000,
Ph=120.959999)
L157=MODEL LINE_BUFFER(N_157-N_158-VDD)(PR=0.001250,PL=0.000249,PC=0.157000,
Ph=300.899994)
L158=MODEL LINE(N_158-N_159)(PR=0.001250,PL=0.000249,PC=0.157000)
L159=MODEL LINE(N_159-N_160)(PR=0.001250,PL=0.000249,PC=0.157000)
L160=MODEL LINE(N_160-N_161)(PR=0.001875,PL=0.000273,PC=0.126700)
L161=MODEL LINE(N_161-N_162)(PR=0.001875,PL=0.000273,PC=0.126700)
L162=MODEL LINE(N_162-N_163)(PR=0.001875,PL=0.000273,PC=0.126700)
L163=MODEL LINE(N_163-N_164)(PR=0.001875,PL=0.000273,PC=0.126700)
L164=MODEL LINE(N_160-N_165)(PR=0.002720,PL=0.000288,PC=0.107400)
L165=MODEL LINE(N_165-N_166)(PR=0.002720,PL=0.000288,PC=0.107400)
L166=MODEL LINE(N_166-N_167)(PR=0.002720,PL=0.000288,PC=0.107400)
L167=MODEL LINE(N_167-N_168)(PR=0.002720,PL=0.000288,PC=0.107400)
L168=MODEL LINE(N_168-N_169)(PR=0.002720,PL=0.000288,PC=0.107400)
L169=MODEL LINE(N_169-N_170)(PR=0.002720,PL=0.000288,PC=0.107400)
L170=MODEL LINE(N_170-N_171)(PR=0.002720,PL=0.000288,PC=0.107400)
L171=MODEL LINE(N_171-N_172)(PR=0.002720,PL=0.000288,PC=0.107400)
L172=MODEL LINE(N_122-N_173)(PR=0.001250,PL=0.000249,PC=0.157000)
L173=MODEL LINE(N_173-N_174)(PR=0.001250,PL=0.000249,PC=0.157000)
L174=MODEL LINE(N_174-N_175)(PR=0.001250,PL=0.000249,PC=0.157000)
L175=MODEL LINE(N_175-N_176)(PR=0.001250,PL=0.000249,PC=0.157000)
L176=MODEL LINE(N_176-N_177)(PR=0.001250,PL=0.000249,PC=0.157000)
L177=MODEL LINE(N_177-N_178)(PR=0.001250,PL=0.000249,PC=0.157000)
L178=MODEL LINE_BUFFER(N_178-N_179-VDD)(PR=0.001250,PL=0.000249,PC=0.157000,
Ph=360.880005)
L179=MODEL LINE(N_179-N_180)(PR=0.001250,PL=0.000249,PC=0.157000)
L180=MODEL LINE(N_180-N_181)(PR=0.001875,PL=0.000273,PC=0.126700)
L181=MODEL LINE(N_181-N_182)(PR=0.001875,PL=0.000273,PC=0.126700)
L182=MODEL LINE(N_182-N_183)(PR=0.001875,PL=0.000273,PC=0.126700)
L183=MODEL LINE(N_183-N_184)(PR=0.001875,PL=0.000273,PC=0.126700)
L184=MODEL LINE(N_184-N_185)(PR=0.001875,PL=0.000273,PC=0.126700)
L185=MODEL LINE(N_185-N_186)(PR=0.001875,PL=0.000273,PC=0.126700)
L186=MODEL LINE(N_186-N_187)(PR=0.001875,PL=0.000273,PC=0.126700)
L187=MODEL LINE(N_187-N_188)(PR=0.001875,PL=0.000273,PC=0.126700)
L188=MODEL LINE(N_188-N_189)(PR=0.011150,PL=0.000319,PC=0.106550)
L189=MODEL LINE(N_189-N_190)(PR=0.011150,PL=0.000319,PC=0.106550)
L190=MODEL LINE(N_190-N_191)(PR=0.011150,PL=0.000319,PC=0.106550)
L191=MODEL LINE(N_191-N_192)(PR=0.011150,PL=0.000319,PC=0.106550)
L192=MODEL LINE(N_192-N_193)(PR=0.011150,PL=0.000319,PC=0.106550)
L193=MODEL LINE(N_193-N_194)(PR=0.011150,PL=0.000319,PC=0.106550)
L194=MODEL LINE(N_194-N_195)(PR=0.011150,PL=0.000319,PC=0.106550)
L195=MODEL LINE(N_195-N_196)(PR=0.011150,PL=0.000319,PC=0.106550)
L196=MODEL LINE(N_188-N_197)(PR=0.002720,PL=0.000288,PC=0.107400)
L197=MODEL LINE(N_197-N_198)(PR=0.002720,PL=0.000288,PC=0.107400)
L198=MODEL LINE(N_198-N_199)(PR=0.002720,PL=0.000288,PC=0.107400)
L199=MODEL LINE(N_199-N_200)(PR=0.002720,PL=0.000288,PC=0.107400)
L200=MODEL LINE(N_200-N_201)(PR=0.002720,PL=0.000288,PC=0.107400)
L201=MODEL LINE(N_201-N_202)(PR=0.002720,PL=0.000288,PC=0.107400)
```

```
L202=MODEL LINE(N_202-N_203)(PR=0.002720,PL=0.000288,PC=0.107400)
L203=MODEL LINE(N_203-N_204)(PR=0.002720,PL=0.000288,PC=0.107400)
L204=MODEL LINE(N_180-N_205)(PR=0.002720,PL=0.000288,PC=0.107400)
L205=MODEL LINE(N_205-N_206)(PR=0.002720,PL=0.000288,PC=0.107400)
L206=MODEL LINE(N_206-N_207)(PR=0.002720,PL=0.000288,PC=0.107400)
L207=MODEL LINE(N_207-N_208)(PR=0.002720,PL=0.000288,PC=0.107400)
L208=MODEL LINE(N_208-N_209)(PR=0.002720,PL=0.000288,PC=0.107400)
L209=MODEL LINE(N_209-N_210)(PR=0.002720,PL=0.000288,PC=0.107400)
L210=MODEL LINE(N_210-N_211)(PR=0.002720,PL=0.000288,PC=0.107400)
L211=MODEL LINE(N_211-N_212)(PR=0.002720,PL=0.000288,PC=0.107400)
L212=MODEL LINE(N_212-N_213)(PR=0.002720,PL=0.000288,PC=0.107400)
L213=MODEL LINE(N_213-N_214)(PR=0.002720,PL=0.000288,PC=0.107400)
L214=MODEL LINE(N_214-N_215)(PR=0.002720,PL=0.000288,PC=0.107400)
L215=MODEL LINE(N_215-N_216)(PR=0.002720,PL=0.000288,PC=0.107400)
L216=MODEL LINE(N_216-N_217)(PR=0.002720,PL=0.000288,PC=0.107400)
L217=MODEL LINE(N_217-N_218)(PR=0.002720,PL=0.000288,PC=0.107400)
L218=MODEL LINE(N_218-N_219)(PR=0.002720,PL=0.000288,PC=0.107400)
L219=MODEL LINE(N_219-N_220)(PR=0.002720,PL=0.000288,PC=0.107400)
L220=MODEL LINE(N_212-N_221)(PR=0.001250,PL=0.000249,PC=0.157000)
L221=MODEL LINE(N_221-N_222)(PR=0.001250,PL=0.000249,PC=0.157000)
L222=MODEL LINE(N_222-N_223)(PR=0.001250,PL=0.000249,PC=0.157000)
L223=MODEL LINE(N_223-N_224)(PR=0.001250,PL=0.000249,PC=0.157000)
L224=MODEL LINE(N_224-N_225)(PR=0.001250,PL=0.000249,PC=0.157000)
L225=MODEL LINE(N_225-N_226)(PR=0.001250,PL=0.000249,PC=0.157000)


FEATURES
GROUND=(0)

EXECUTION CONTROLS
ANALYZE MAIN(TRANSIENT)

RUN CONTROLS
STOPTIME=20
GRAPHICS

OUTPUTS
PRINT,PLOT
P_SIM,nN_0,nN_46,nN_50,nN_66,nN_76,nN_92,nN_100,nN_110,nN_118,nN_148,nN_156,n
N_164,nN_172,nN_196,nN_204,nN_220,nN_226,
```

The output file is an AS/X file with the information about the inserted buffers and the cost function given in the first few comment lines of the file. A comment in AS/X starts with a "*". The first line is

```
*iterations = 6
```

which gives the number of traversals for the tree before converging to an optimum solution. This is an important metric for the convergence rate of the repeater insertion algorithm.

The next set of lines gives the propagation delays at the sinks of the tree after the repeater insertion and optimization tool has finished running, *i.e.*, with the optimum repeaters inserted.

The time is given in nanoseconds and the sinks are listed in order starting by the left most sink of the tree. Note that the sink delays are very well balanced indicating good optimization.

```
*Sink Delays are (Left to Right):
*
*0.422775
*0.418599
*0.410650
*0.414385
*0.422739
*0.412066
*0.403790
*0.403790
*0.411279
*0.402124
*0.415111
*0.421998
*0.419892
*0.410730
*0.415662
*0.413019
```

The next line is

```
*-----------------------------------------------------------
*Cost = 0.422775
*-----------------------------------------------------------
```

This line gives the minimum cost achieved. In the above optimization run, the cost was selected as the maximum sink delay.

The next few lines give the sizes and positions of the inserted buffers. The buffers are listed in order starting from the left baths to the right paths. Each path is crossed starting from the root towards the sink. A zero at a buffer position indicated that no buffer is inserted at that position. Any other non-zero value at a buffer position indicates the size of the inserted buffer in *minimum size inverter units*. Hence, for a 0.25 micrometer technology, a 100 size means 25 micrometer channel width inverter.

```
*Optimum Buffers are (root towards leafs, left then right):
*
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  173.742401  0.000000
0.000000
*0.000000  300.899994  600.799988  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
```

```
*300.899994  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*71.536476  300.899994  600.799988  0.000000  0.000000  0.000000  0.000000
300.899994
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
300.899994
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  180.460159
*480.839966  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  600.799988  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  120.959999  300.899994  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*360.880005  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.000000  0.000000
*0.000000  0.000000  0.000000  0.000000
*-----------------------------------------------------------
```

The next two lines give the total buffer area in minimum size inverters and the run time of the program in milliseconds, respectively.

```
*Buffer Area = 4996.218262
*
*Run Time = 72622
```

After the comment information lines comes the AS/X commands. The first portion of this file is two subcircuits. These two subcircuits are shown below and are named LINE and LINE_BUFFER, respectively. The LINE subcircuit represents a section of a line with a resistance *R*, an inductance *L*, and a capacitance *C*. The model LINE_BUFFER is the same as LINE but with an extra buffer connected at the input of the line. Hence, if a buffer is inserted at a possible buffer position (at the beginning of a line section), then LINE_BUFFER should be use. Otherwise, LINE should be used.

```
MODEL DESCRIPTION

MODEL LINE(IN-OUT)
ELEMENTS
PR=1
PL=1
PC=1
R, IN-NN=(PR)
L, NN-OUT=(PL)
C, OUT-0=(PC)

FEATURES
GLOBAL=(N0)


MODEL LINE_BUFFER(IN-OUT-VDD)
ELEMENTS
PR=1
PL=1
PC=1
Ph=1
WN=(0.25*Ph)
WP=(2.12*WN)
MP1=MODEL Pfet(IN-VDD-VDD-N1)(PW=WP,PL=0.25)
MN1=MODEL Nfet(IN-0-0-N1)(PW=WN,PL=0.25)
R, N1-N2=(PR)
L, N2-OUT=(PL)
C, OUT-0=(PC)

FEATURES
GLOBAL=(N0)
```

The next three statements in the output file are

```
E_VDD,0-VDD=2.5
E_VP,0-N_0=(PULSE(0,2.5,0,0.03,0.03,10,40))

P_SIM=(AREA(IE_VDD,0,20)*2.5+AREA(IE_VP,0,9)*2.5)
```

which define the value of V_{DD} at 2.5 volts, the input waveform, and calculates the power consumed by the repeater system per input transition, respectively.
The next set of lines constitutes the bulk of the file and represent the connectivity of the interconnect and the inserted buffers. Example two lines are:

```
L82=MODEL LINE(N_82-N_83)(PR=0.011150,PL=0.000319,PC=0.106550)
L83=MODEL LINE_BUFFER(N_83-N_84-VDD)(PR=0.011150,PL=0.000319,PC=0.106550,
Ph=300.899994)
```

The first line is a LINE with no buffer at the beginning. The second line is a LINE_BUFFER with a buffer at the beginning of size 300 times a minimum size inverter. Note that resistances are given in kilo-ohms, inductances in micro-Henry, and capacitances in pico-farads, resulting in time in nano seconds.

The remaining lines describe the type of the analysis performed, the stop time, and the outputs to be plotted which are the sinks of the tree from left to right, respectively, as shown below.

```
EXECUTION CONTROLS
ANALYZE MAIN(TRANSIENT)

RUN CONTROLS
STOPTIME=20
GRAPHICS

OUTPUTS
PRINT,PLOT
P_SIM,nN_0,nN_46,nN_50,nN_66,nN_76,nN_92,nN_100,nN_110,nN_118,nN_148,nN_156,n
N_164,nN_172,nN_196,nN_204,nN_220,nN_226,
```

IV. Variables and Parameters

This section describes the variables and parameters in the program that can be tuned by the user to control the performance and operation of the repeater insertion tool. Currently, many parameters are defined as global variables in the C source code but can be easily read from an input file if desired.

Num_of_Bufs: This is not the number of buffers in a library. This is an internal variable that the Find_Best_Buffer functions use to determine the exact size of the buffers for minimum cost at different nodes. Buffers are just inverters.

struct wire{float R; float C; float L;struct wire * lw; struct wire* rw; float b; float Ct;float td;}: This is the basic internal wire structure. A Tree is composed of many wires connected together. A wire is a straight segment of interconnect that can be modeled with lumped elements. Long wires can be partitioned into shorter wires as explained below. Each wire has a total resistance of R, capacitance C, and inductance L, and points to a left wire and a right wire that may be connected at the end of the wire to form a tree. At the beginning of each wire segment there is a possibility of inserting a buffer of size b. The special value of zero in b means no buffer is inserted at the beginning of the given wire segment. These are the main variables that the program calculates.

Longer wires are partitioned into many segments to increase the possible buffer positions for better optimization as explained below. The variable Ct is the total capacitance down stream of the wire and td is the delay up to the input of the wire.

struct pwire{float w; float R; float L; float C;}: This structure is used to describe the types of wires that appear in the interconnect. For the specific technology used, the wires cannot just take any width. There are only few wire widths available. For each wire width w, the capacitance inductance and resistance per unit length are given by C, L, and R, respectively. The global array of pwire structures wt below includes the information for the typically use wires. Using this technique of precharacterizing the available wire types eliminates the extraction step and significantly simplifies the input file format. A sample input file was given in section II.

static struct pwire wt[]= {{0.9,6.48e-5,6.83e-7,1.375e-4},
{2.5,2.23e-5,6.374e-7,2.131e-4},{3.35,5.44e-6,5.77e-7,2.148e-4},
{4.86,3.75e-6,5.46e-7,2.534e-4},{7.24,2.5e-6,4.98e-7,3.14e-4},
{9,2.03e-6,4.62e-7,3.59e-4},{22.80,8.03e-7,3.401e-7,7.12e-4},
{30,6.1e-7,3.4e-7,8.9e-4}}: This array of pwires include the commonly used wire sizes in the given technology. See the comment on the structure pwire above.

ro: The output resistance of a minimum size CMOS inverter in the linear region with VGS=VDD. The output resistance scales inversely with the buffer size.

co: The input capacitance of minimum size CMOS inverter. Scales in proportion to the buffer size.

D: The intrinsic delay of a minimum size inverter. This delay is due to the gate driving its own capacitance and does not scale with the buffer size.

Rsat: The saturation resistance of a minimum size inverter due to channel modulation and scales inversely with the buffer size.

Isat: The saturation current of a minimum size inverter in micro amperes. Scales proportional to buffer size.

Cout: The output diffusion capacitance of a minimum size inverter. Scales in proportion to the buffer size.

bmax: The maximum allowed buffer size relative to the size of a minimum size inverter.

Costfn: This variable determines the type of cost function used. There is a choice of two cost functions here. The first one is the sum of the exp(30*td), where td is the sink delay and the summation runs over all the sinks. This function is a continuous function that minimizes the maximum path delay since the very high power in the exponential will favor minimizing the maximum delay. The other cost function is just the summation of td at all the sinks. costfn = 0 means using the exponentials.

RC: If RC is equal to 1,then inductance is ignored and the line is treated as an RC line. This variable is useful when the user is interested in knowing how much difference in the repeater solution does including inductance make as compared to just using an RC model.

imps: This is a scaling factor that can be used to change the time scale by multiplying imps by the inductances and capacitances in the circuit. imps = 1 means nano seconds here. To use pico seconds for example, imps should be equal to 1/1000.

length_per_section=501: This variable determines the maximum length allowed for any section in micrometers. Hence, the value used here means that any line longer than 0.5 mm will be partitioned into several sections. This is important both for the accuracy of the estimated delay as well as for increasing the possible buffer positions for longer wires since buffers can only be inserted at the beginning of each wire. Longer wires usually require more buffers to be inserted along the wire length. See the Underlying theory document.

V. Function Description

Function: float beff(float b)

Arguments:
b: Drawn size of the transistor channel length.

Returns: Effective transistor width due to channel width modulation with the depletion regions.

Description: This function calculates the effective buffer size for current calculation purposes. The current does not scale linearly with the buffer size due to the channel width modulation. For small transistor widths, the depletion regions significantly reduce the effective channel width. However, for large widths, the depletion regions are negligible compared to the transistor width and the current scales almost linearly with the buffer width. The technology dependent equation in the function accounts for this fact. beff is used as W/L in current calculation.

Function: void Read_Tree(FILE *fp, char* source,struct wire** lw1, struct wire **rw1)

Arguments:
fp: A pointer to the input file containing the interconnect information.
source: Name of the source node. Currently taken as "root".
lwl: Pointer to the pointer to the left wire at a dissection of a binary tree.
Rwl: Pointer to the pointer to the left wire at a dissection of a binary tree.

Returns: Void.

Description: This function fills up the internal tree structure with the information from the input file. The first (source) node of the tree has to be labeled "root" in the input file because of the Read_Tree function call in main. The internal tree structure starts by a wire root describing the root wire and pointing to the right and left wires driven by root. These left and right wires point in turn to their left and right wires. If a wire does not drive a left or a right wire, then the pointer to the nonexistent wire is set to zero. A leaf has both left and right pointers equal zero. The buffers in the tree are set to zeros (no buffers) initially. Later the repeater insertion algorithm changes the values to find the optimum buffer solution. A sample input file is given in section II.

Function: float Insert_Bufs(struct wire* root)

Arguments:
root: Pointer to root wire of the interconnect tree.

Returns: The cost after inserting the buffers.

Description: This function traverses the tree several times iteratively trying to determine the optimum buffer solution that minimizes the cost. As described in the Underlying Theory document, an iteration involves traversing the tree once assigning the optimum buffers and calculating the cost at the end. This step is performed by the function Do_Iteration2. The iterations are repeated until the cost does not improve by more than 0.1% as assigned here. The total number of iterations is typically less than six iterations. This parameter can be increased to 2% for example to reduce the run time depending on the required accuracy and run time constraints. The iterations are repeated two times here, once with an exponential cost function given by sum of the exp(30*td), where td is the sink delay and the summation runs over all the sinks. This function is a continuous function that minimizes the maximum path delay since the very high power in the exponential will favor minimizing the maximum delay. The other cost function is just the summation of td at all the sinks. costfn = 0 means using the exponentials. Any other cost function can be defined in the Cal_Cost function described later.

Function: float Do_Iteration2(struct wire* root, struct wire* w1)

Arguments:
root: Pointer to root of the tree.
w1: Pointer to the wire where the optimum buffer is to be found.

Returns: The cost after inserting the buffers.

Description: This function traverses the tree once to determine the best buffers starting form the last buffer solution as described in the Underlying Theory document. If a wire has both left and right descendants, the two buffers in these wires are calculated simultaneously which is slower but more accurate than doing each buffer alone (see the Underlying Theory Document).

Function: float Find_Best_Buffer(struct wire * root, struct wire * w1)

Arguments:
root: Pointer to root of the tree.
w1: Pointer to the wire where the optimum buffer is to be found.

Returns: The cost after inserting the best buffer.

Description: The function Find_Best_Buffer changes the buffer size at a given node to find the buffer that minimizes the total cost. The function looks at only one node at a time, but the solution is affected by the sizes of the other buffers in other nodes. That is why several traversals of the tree are required to find the optimum buffer solution. See the Underlying Theory Document. This function does not use a given library of buffers. Rather, buffers here are CMOS inverters and the sizes are determined as continuous variables relative to a minimum size inverter. There are two possible ways to accommodate a buffer library. First, modify the function to try the given set of buffers in sequence and determine the buffer that gives the minimum cost. The other way is to map each inverter to the buffer closest in size in the library.

Function: float Find_Best_2Buffers(struct wire * root, struct wire * w1, struct wire * w2)

Arguments:
root: Pointer to the root of the tree.
w1: Pointer to one of two wires where the optimum buffers are to be found.
w2: Pointer to the second wire where the optimum buffers are to be found.

Returns: The cost after inserting the two best buffers.

Description: Same as Find_Best_Buffer, but finds the optimum sizes of two inverters in two nodes simultaneously. Naturally, this function is slower than Find_Best_Buffer but is more accurate. See the Underlying Theory Document.

Function: float Cal_Cost(struct wire* w1, int Print)

Arguments:
w1: A wire pointer that the cost is calculated downstream of. Should be root for the first call.
Print: If 1, prints the data about the buffers and the cost.

Returns: Cost of sinks downstream of the given wire.

Description: This function calculates the cost of a buffered tree. First, the total capacitive loads down stream of all wires is calculated. Next the function Get_Cost is called to calculate the cost.

The input w1 should be the root wire while Print = 1 means the function Get_Cost will print the buffer solution, cost, sink delays, ..., while calculating the cost.

Function: float Cal_Loads(struct wire* w1)

Arguments:
w1: A wire pointer that the total capacitance is calculated downstream of. Should be root for the first call.

Returns: Total capacitance down stream of w1.

Description: Calculates the total capacitance down stream of the wire w1. Note that if a buffer exits downstream of w1, the capacitances after the buffer are not counted in.

Function: float Get_Cost(struct wire * w1, float tdsatin, float TRCin, float TLCin, float tdin, int Print)

Arguments:
See the function description below.

Returns: The cost.

Description: This function calculates the cost based on the equivalent Elmore delay model for RLC circuits and the nonlinear device model described in the Underlying Theory document. In the equivalent Elmore Delay model, there are two time constants to be calculated from the tree. The first time constant is the RC time constant. This RC time constant is simply Elmore delay and is calculated by adding the multiplications of the resistances from the input to the wire w1 by the total down stream capacitances. The TRCin is the RC time constant up to w1. The value TRCin+w1->R*w1->Ct represents the RC constant at the end of w1. By crossing the tree recursively starting at the root, the RC time constant can be calculated at all the nodes. The LC time constant is calculated in exactly the same manner but using the wires' inductances rather than resistances. If a buffer is encountered, the summation of RC and LC time constants is stopped and restarted again after the buffer. The variable tdin is the delay up to the wire w1. In general this delay is set to zero unless a buffer is existent at the beginning of w1. In that case tdin in the total delay from the input (root) to the input of the buffer. Hence, if a buffer is encountered downstream of a wire, the summation of the RC and LC constants up to the buffer are used to calculate the delay tdin for the wire that has the buffer. If a buffer is existent at the beginning of a subtree, the delay is calculated twice, once in the linear region and another in the saturation

region and the two solutions are combined as described in the Underlying Theory document to determine the delay tdin of w1.

Function: float DRLC(float TRC, float TLC)

Arguments:
TRC: The *RC* time constant of the circuit.
TLC: The square of the *LC* time constant of the circuit.

Returns: The delay.

Description: Calculates the equivalent Elmore Delay model. If the global variable RC is set equal to one, the functions ignores the inductance is the circuit.

Function: void Print_Results(struct wire* root)

Arguments:
root: Pointer to the root of the tree.

Returns: Void

Description: This function prints a summary of the buffer solution including the optimum buffers, the cost, sink delays, run time, number of iterations, ... A sample printout is shown in section III.

Function: void Generate_ASX_File(struct wire * root)

Arguments:
root: Pointer to the root of the tree.

Returns: Void

Description: Generates an AS/X file describing the buffered tree. This file can be readily used to simulate the buffered tree using AS/X to get the actual delays and the power consumption. A sample AS/X output file was shown in section III.
