4. How a Generator Works...

Generating a Zeroflag 8 bit

(boundary)
$NET art_zer8
$INP pa1x0 pa1x1 pa1x2 pa1x3
pa1x4 pa1x5 pa1x6 pa1x7
$OUT pa2x4

(core...)
(tree- root, hierarchie level 1)
$SUB nand s_0 0 4
$INP pa2x2 pa2x6
$OUT pa2x4

(next lower level follows...)
(hierarchie level 2)
$SUB nor s_1 0 2
$INP pa2x1 pa2x3
$OUT pa2x2

$SUB nor s_2 0 6
$INP pa2x5 pa2x7
$OUT pa2x6

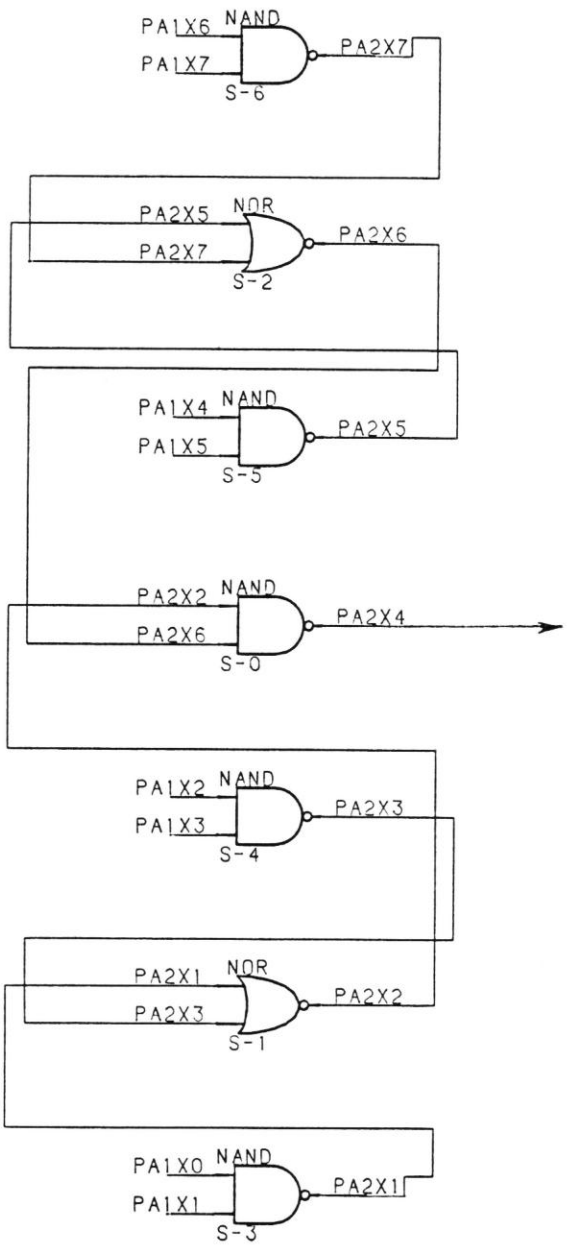
(next lower level follows...)
(hierarchie level 1)
$SUB nand s_3 0 1
$INP pa1x0 pa1x1
$OUT pa2x1

$SUB nand s_4 0 3
$INP pa1x2 pa1x3
$OUT pa2x3

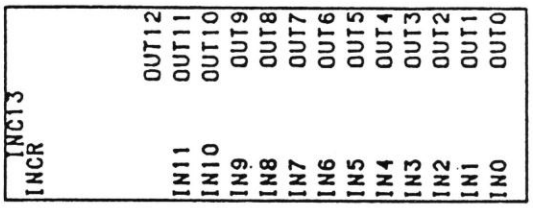
$SUB nand s_5 0 5
$INP pa1x4 pa1x5
$OUT pa2x5

$SUB nand s_6 0 7
$INP pa1x6 pa1x7
$OUT pa2x7

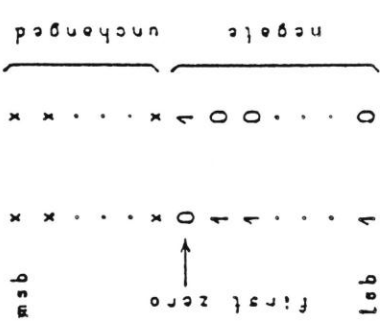
$END

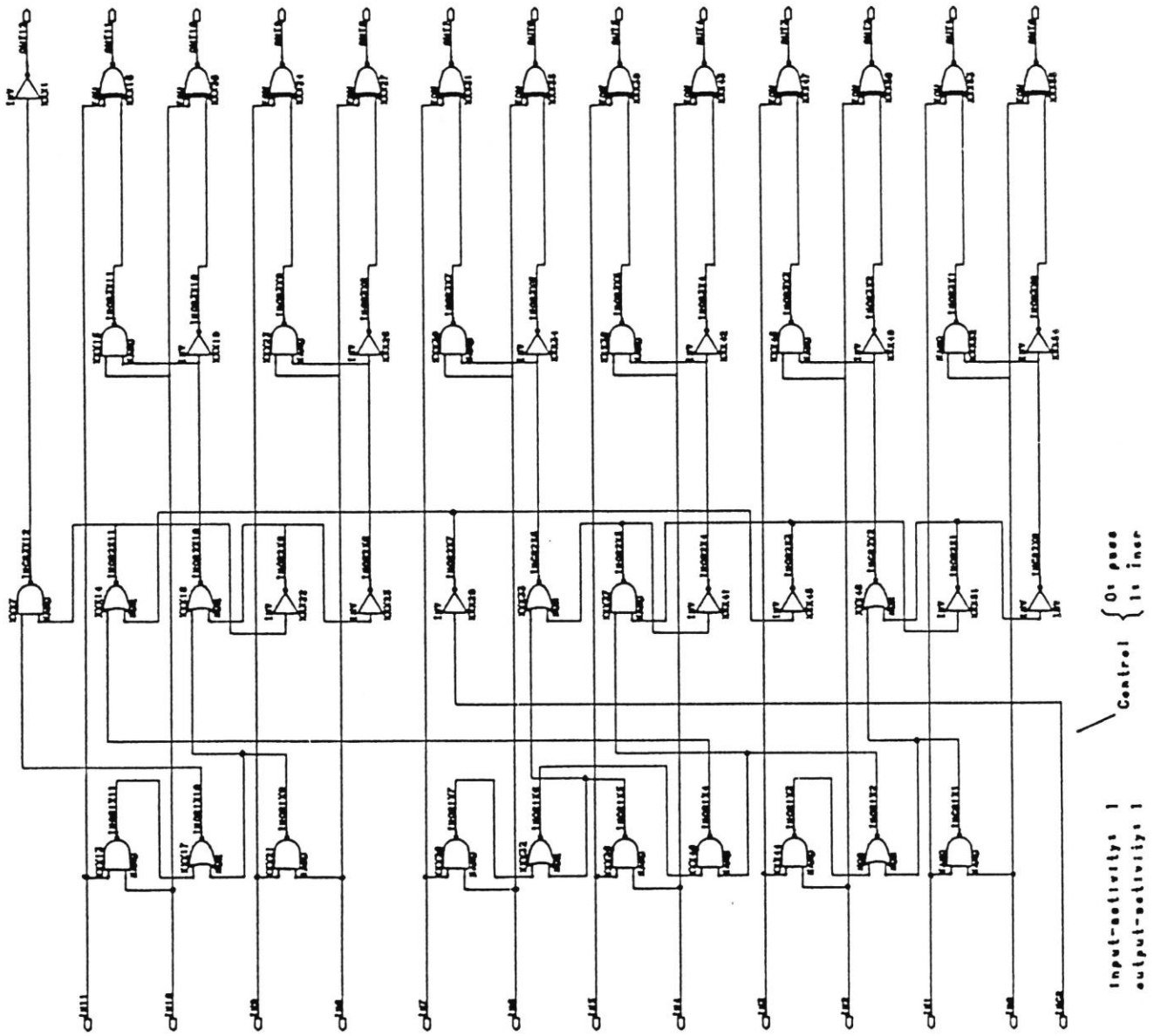


MULTIPLE TREES



Symbol INC13

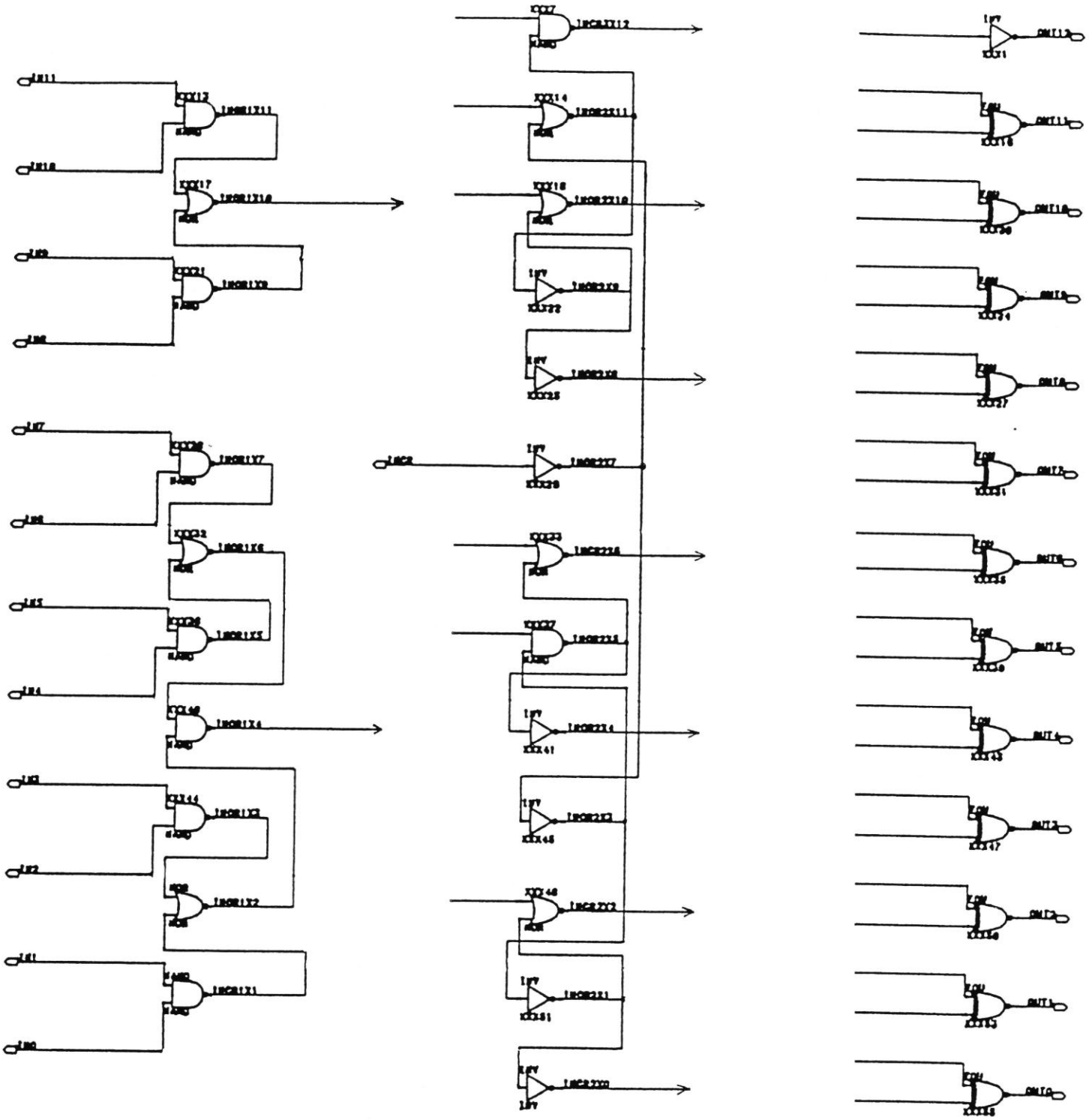




227 transistors

delay: 9 gates

INCREMENTER 13 BIT, SCHEME AND SYMBOL



a) b) c)

INCREMENTER 13 BIT: PARTIAL TREE STRUCTURES (a...c)

a) FIRST ZERO SEARCH TREE (column 1)

b) ZERO/ ONE SEPARATE TREE (column 2 and 3)

c) INVERT RESULT TREE (column 4)

ALGORITHMUS

1) Bestimmung der Busbreite als Binärzahl $b_{k-1}...b_0$

2) Ermittlung der Baumtiefe k_{max} als Stellenzahl der binären Busbreite

3) Netzkopfbehandlung
- **Berechnung der Randpin- Koordinaten und Namen**
- **Rückgabe der Parameterliste als Kommentar**

4) Bestimmung freier Matrixpositionen

5) Berechnung der Elemente

for (i = 0; i <= busbreite; i++)

{

a) Bestimmung der Elemente der nullten Spalte (x = 0)

Anordnung der Operatoren p(x,y) auf jedem j-tem Platz mit der Bitposition

$p_i(x,y) = (0, i * j)$ **mit** $j \in \{1,2,3...\}$

(0 =< i j =< busbreite)

b) Bestimmung der Elemente der ersten Spalte (x = 1)

Anordnung der Schalterpositionen s(x,y) von der höchsten Hierarchieebene absteigend; Bestimmung des Schaltertyps q entsprechend Hierarchieebene, Buffer und L/H- Aktivität

$s_i(x,y,q) = (1, 2^{k-1} i + 2^{k-2}, q)$

.

. **(Bestimmung weiterer Spalten)**

.

c) Benennung der abgehenden Leitung(en) des Gatters

d) Korrektur abgeschnittener Schalter

e) Nachbehandlung I/O- seitiger L/H- Aktivitäten

f) Kontrolle des statthaften Fanout

g) falls erforderlich: Einfügung eines Fanout- Korrekturbaumes in freie Matrixposition

}

6) Konvertierung in gewähltes Netzlistenformat, Zeile in File schreiben

7) Ausgabe in gewähltes Netzlistenformat

GENERATED NETLIST DATA FORMAT *.NBS

```
! 16,4,0,0,1
!
!________ increment ___________
!
! data input activity: high
! data output activity: high
! data bus width: 16 bit
!

s: test_inc16  #
          e:in(0:15),incr   #
          a:out(0:15)
!
ic_1_(1)  in(0)-in(1)-incr1(1)  =nand
ic_1_(3)  in(2)-in(3)-incr1(3)  =nand
ic_1_(5)  in(4)-in(5)-incr1(5)  =nand
ic_1_(7)  in(6)-in(7)-incr1(7)  =nand
ic_1_(9)  in(8)-in(9)-incr1(9)  =nand
ic_1_(11)  in(10)-in(11)-incr1(11)  =nand
ic_1_(13)  in(12)-in(13)-incr1(13)  =nand
ic_1_(2)  incr1(1)-incr1(3)-incr1(2)  =nor
ic_1_(6)  incr1(5)-incr1(7)-incr1(6)  =nor
ic_1_(10)  incr1(9)-incr1(11)-incr1(10)  =nor
ic_1_(4)  incr1(2)-incr1(6)-incr1(4)  =nand
ic_2_(0)  incr2(1)-incr2(0)  =inv
ic_2_(2)  incr2(1)-incr1(1)-incr2(2)  =nor
ic_2_(4)  incr2(5)-incr2(4)  =inv
ic_2_(6)  incr2(5)-incr1(5)-incr2(6)  =nor
ic_2_(8)  incr2(9)-incr2(8)  =inv
ic_2_(10)  incr2(9)-incr1(9)-incr2(10)  =nor
ic_2_(12)  incr2(13)-incr2(12)  =inv
ic_2_(14)  incr2(13)-incr1(13)-incr2(14)  =nor
ic_2_(1)  incr2(3)-incr2(1)  =inv
ic_2_(5)  incr2(3)-incr1(2)-incr2(5)  =nand
ic_2_(9)  incr2(11)-incr2(9)  =inv
ic_2_(13)  incr2(11)-incr1(10)-incr2(13)  =nand
ic_2_(3)  incr2(7)-incr2(3)  =inv
ic_2_(11)  incr2(7)-incr1(4)-incr2(11)  =nor
ic_2_(7)  incr-incr2(7)  =inv
ic_3_(0)  incr2(0)-incr3(0)  =inv
ic_3_(1)  incr2(0)-in(0)-incr3(1)  =nand
.
.
.
ic_4_(6)  incr3(6)-in(6)-out(6)  =antiv
ic_4_(7)  incr3(7)-in(7)-out(7)  =antiv
ic_4_(8)  incr3(8)-in(8)-out(8)  =antiv
ic_4_(9)  incr3(9)-in(9)-out(9)  =antiv
ic_4_(10)  incr3(10)-in(10)-out(10)  =antiv
ic_4_(11)  incr3(11)-in(11)-out(11)  =antiv
ic_4_(12)  incr3(12)-in(12)-out(12)  =antiv
ic_4_(13)  incr3(13)-in(13)-out(13)  =antiv
ic_4_(14)  incr3(14)-in(14)-out(14)  =antiv
ic_4_(15)  incr3(15)-in(15)-out(15)  =antiv
##
```

GENERATED NETLIST DATA FORMAT *.TRA

```
$NET test_bsh16
$INP  sh0x0 sh0x1 sh0x2 sh0x3 sh0x4 sh0x5 sh0x6 sh0x7 sh0x8 sh0x9
      sh0x10 sh0x11 sh0x12 sh0x13 sh0x14 sh0x15 c1x0 c1x1 c1x2 c1x3

$OUT  sh4x0 sh4x1 sh4x2 sh4x3 sh4x4 sh4x5 sh4x6 sh4x7 sh4x8 sh4x9
      sh4x10 sh4x11 sh4x12 sh4x13 sh4x14 sh4x15

$SUB inv4      inv4     0 -1
$INP c1x0
$OUT nc0x0

$SUB inv4      inv4     1 -1
$INP c1x1
$OUT nc0x1

$SUB inv4      inv4     2 -1
$INP c1x2
$OUT nc0x2

$SUB inv4      inv4     3 -1
$INP c1x3
$OUT nc0x3

$SUB inv4      inv4     0 -2
$INP nc0x0
$OUT c0x0

$SUB inv4      inv4     1 -2
$INP nc0x1
$OUT c0x1

$SUB inv4      inv4     2 -2
$INP nc0x2
$OUT c0x2

$SUB inv4      inv4     3 -2
$INP nc0x3
$OUT c0x3

$SUB mux       mux      0 0
$INP sh0x0 sh0x1 c0x0 nc0x0
$OUT sh1x0

$SUB mux       mux      0 1
$INP sh0x1 sh0x2 c0x0 nc0x0
$OUT sh1x1

.
.
.
$SUB mux       mux      3 15
$INP sh3x15 sh3x7 c0x3 nc0x3
$OUT sh4x15

$END
```

5. TECHNOLOGY MAPPING

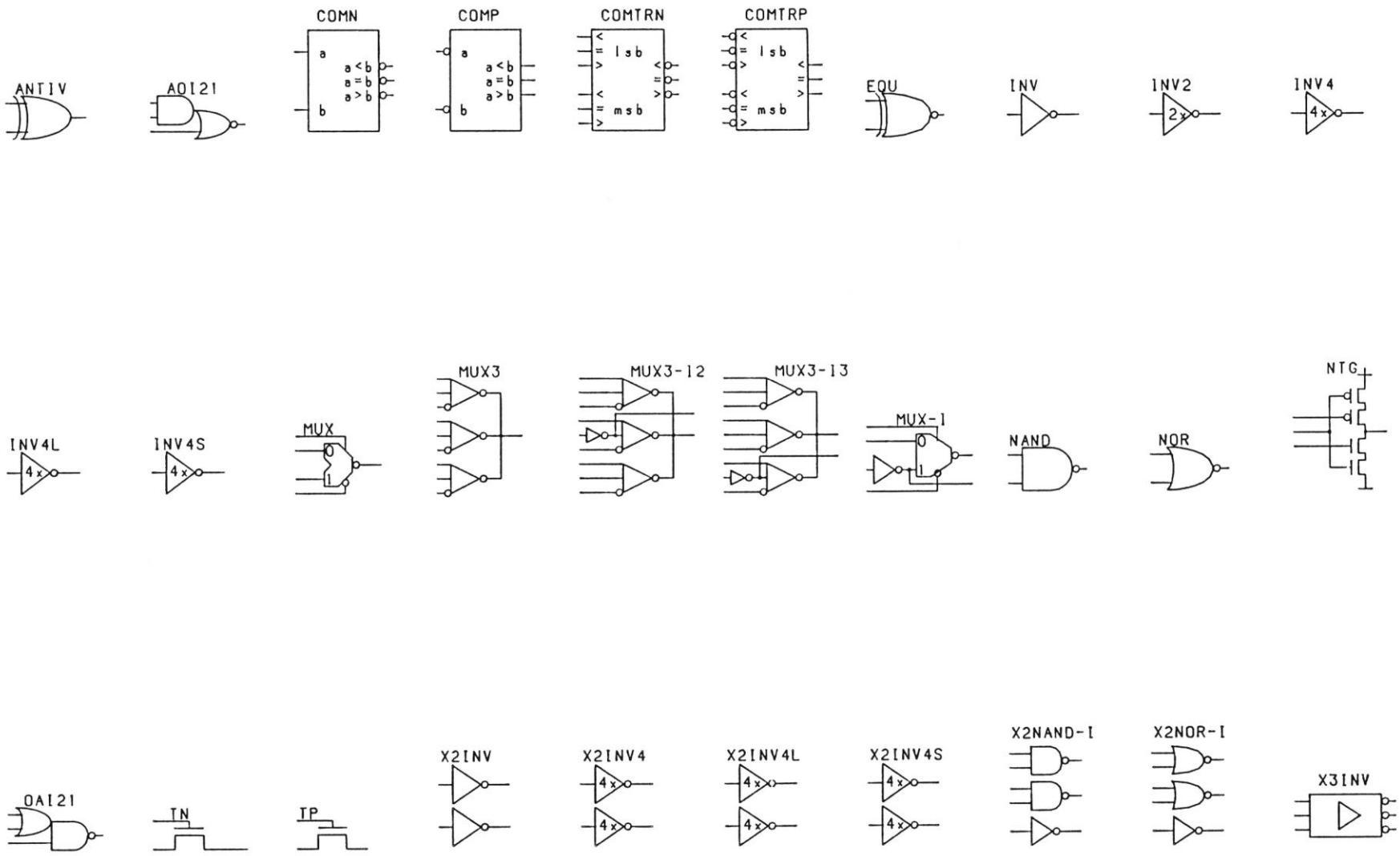
NETLIST FORMATING

```
struct    pot {char *name;
              int  f,x,y,z;
              };

struct    lin {int      eltyp, inp_nr, out_nr, bidir_nr;
              char     *eltyp_char;
              struct   pot inp[16];
              struct   pot out[7];
              struct   pot bid[6];
              struct   pot ele_name;
              } *line;
```

GATE DESCRIPTION FILE (*.CMD)
Standart Interchange Format (SIF)

```
blk CONVGATE
  ///
edn 7
typ INV
pad A PN-INPUT
pad B PN-INPUT
pad C PN-INPUT
pad G PN-OUTPT
pad H PN-OUTPT
pad I PN-OUTPT
pag 8500 11000 1
sub 25 16 0 1 INV
  ///
sub 25 27 0 1 INV
  ///
sub 25 38 0 1 INV
  ///
seg 12 21 25 21
seg 35 21 55 21
seg 12 32 25 32
seg 35 32 55 32
seg 12 43 25 43
seg 35 43 55 43
pcp 12 21 6 0 C  PN-INPU
pcp 55 21 4 0 I  PN-OUTP
pcp 12 32 6 0 B  PN-INPU
pcp 55 32 4 0 H  PN-OUTP
pcp 12 43 6 0 A  PN-INPU
pcp 55 43 4 0 G  PN-OUTP
sym 1400 1200 1
dis 3 11 4 0 1
dis 3 1 4 0 2
pin 14 4 0 0 I PN-OUTPT
pin 14 6 0 0 H PN-OUTPT
pin 14 8 0 0 G PN-OUTPT
pin 0 4 0 0 C PN-INPUT
pin 0 6 0 0 B PN-INPUT
pin 0 8 0 0 A PN-INPUT
cos lin 900 600 600 400
cos lin 600 800 900 600
cos lin 600 400 600 800
cos lin 200 300 200 900
cos lin 1200 900 1200 300
cos bbr 1200 400
cos bbr 1200 600
cos bbr 1200 800
cos lin 1300 400 1400 400
cos lin 1300 800 1400 800
cos lin 1300 600 1400 600
cos lin 0 400 200 400
cos lin 0 600 200 600
cos lin 0 800 200 800
cos lin 1200 300 200 300
cos lin 200 900 1200 900
```


ANTIV
AOI21
COMN
a
b
a < b
a = b
a > b
COMP
COMTRN
COMTRP
lsb
msb
EQU
INV
INV2
2x
INV4
4x
INV4L
INV4S
MUX
0
1
MUX3
MUX3-12
MUX3-13
MUX-1
NAND
NOR
NTG
d
OAI21
IN
IP
X2INV
X2INV4
X2INV4L
X2INV4S
X2NAND-I
X2NOR-I
X3INV

6. SUMMARY

SPEED IMPROVEMENT

(ripple-/tree- algorithm, speed in gate delays)

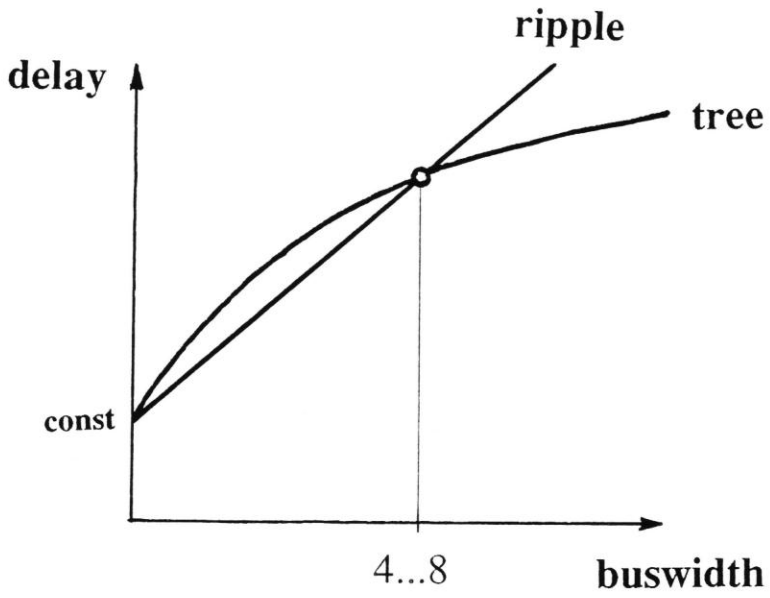
MODUL	BIT	GATE-DELAYS	NORM. AREA
incr/decr	32	64/11	1/3
parity flag	16	17/5	1/2
zero flag	16	17/5	1/2
comparator	64	65/7	1/2

DESIGN EFFICIENCY

(by_hand/generator; times in hours)

MODUL	BIT	DEVELOPMENT	TEST
incr/decr	32	200/.1	6/1
parity flag	16	40/.1	3/1
zero flag	16	40/.1	3/1
comparator	64	80/.1	3/1

DELAY COMPARISON RIPPLE/ TREE



ripple_delay = const + fact * buswidth

tree_delay = const + fact * (log $_{fanout}$ (buswidth) - 1)