Microelectronics netlist generator project

Abdelsalam ElTamawy, Ali Ghazal, Bassel halabi

900170376, 900171722, 900171095

1 User guide

Simply input the expression when prompted and a netlist compatible with LTspice will be produced. Use '&' to AND elements, '|' to OR elements and a '' to not elements. The program supports parenthesis.

Example:

```
Enter your logical expression!
 for AND use &  for OR use | for NOT use '
a&b
*this is the generated netlist for the expression a&b
Vdd -1 0 5v
M0 -5 96 -1 -1 mypmos
M1 -6 -5 -1 -1 mypmos
M2 -8 -6 -1 -1 mypmos
M3 -10 96 -1 -1 mypmos
M4 -11 -10 -1 -1 mypmos
M5 -2 -11 -8 -8 mypmos
M6 -16 -15 -1 -1 mypmos
M7 -21 -20 -1 -1 mypmos
M8 -6 -5 0 0 mynmos
M9 -11 -10 0 0 mynmos
M10 -1 96 -15 -15 mynmos
M11 -16 -15 0 0 mynmos
M12 -2 -16 0 0 mynmos
M13 -1 96 -20 -20 mynmos
M14 -21 -20 0 0 mynmos
M15 -2 -21 0 0 mynmos
V0 96 0 pulse(0 5 0 1n 1n 128m 256m )
V1 -5 0 pulse(0 5 0 1n 1n 64m 128m )
V2 -6 0 pulse(0 5 0 1n 1n 32m 64m )
V3 96 0 pulse(0 5 0 1n 1n 16m 32m )
V4 -10 0 pulse(0 5 0 1n 1n 8m 16m )
V5 -11 0 pulse(0 5 0 1n 1n 4m 8m )
V6 -15 0 pulse(0 5 0 1n 1n 2m 4m )
V7 -20 0 pulse(0 5 0 1n 1n 1m 2m )
*models discription
.model mynmos VDMOS( Rg=3 Vto=1.4 Rd=8m
+ Rs=6m Rb=10m Kp=70 Cgdmax=.5n Cgdmin=.12n
```

```
+ Cgs=.8n Cjo=.24n Is=24p ksubthres=.1
+ mfg=Siliconix Vds=12 Ron=20m Qg=12n

.model mypmos VDMOS(pchan Rg=3 Vto=-1.3 Rd=16m
+ Rs=12m Rb=20m Kp=30 Cgdmax=.6n Cgdmin=.14n
+ Cgs=.9n Cjo=.28n Is=28p ksubthres=.1
+ mfg=Siliconix Vds=-12 Ron=40m Qg=14n

.tran 256m
.backanno
.end
```

2 Parsing

In order to handle the input string efficiently, we thought we could change it from its infix form to prefix. For this purpose, stacks were used where we pushed and popped parts of the strings depending on the type of the character. Also, in handling the string, we were aware of the fact that pull up circuit should in terms of complemented input and pull down are the opposite. Therefore, for the pull up for example, if an input was not complemented, we complemented and if it was complemented we left as it is.

3 Netlist generator

The postfix expression generated by the infix parser is then fed into a function that pareses the postfix expression into a boolean binary expression tree, allowing us to easily represent and, more easily, access what we're looking for in the expression.

The netlist is built through recursively traversing the tree. The algorithm continues to traverse the tree until it reaches a node with children that point to null, then a MOSFET object is created and added to a vector storing a list of all MOSFETs. The recursive function then returns a pair of pointers to the drain and sink which in themselves are recursive structures. Recursive structures are used to created a linked list where when a node in the circuit points to another node (by node we are referring to a drain or sink of a MOSFET), the first node would effectively have the resulting value of the second and if the second points to a third node, then the first node can access the third node through the second node; resulting in all of them having the same value when printing the netlist. A pair of drain and sink pointers are returned from each child, causing a 2 pairs to be given to a parent since this is a binary tree. Depending on what the parent is, this pair of pairs will be manipulated. For instance, if the parent is an '&' then the the drain of the second pair is appended in the linked list to the sink of the first pair. Then this '&' element of the tree then returns the drain of the first pair and the sink of the first pair, thus creating an AND connection between its 2 children and returning a pair of pointers pointing to the start and end of this AND sub circuit to be used by its parent along with the other child of its parent. By the end of this function a pair of pointers pointing to the start and end of the entire circuit is returned to the root call of the function. This final value is given the start value of the input voltage node and the second, is given the value of the output node; this is the case if the returned is the start and end pointers of the pull up network. However, if the start and end pointers of the pull down network is returned then the start gets the value of the output node and the end becomes the ground node.

When printing the MOSFETs, each drain and sinks netlist is traversed until its next value points to null. Then we print the data value of the final cursor pointer; this would be the effective node this MOSFET is connected to.

4 LTspice netlist generation

After researching and exterminating with ltSpice, it turned out that the only addition we need aside from the MOSFETs connections is specifying the voltage input for each signal, the details of the used MOSFETS (both the NMOS and the PMOS), and finally the type of the simulation. For the voltage sources we used a pulse signal (0v and 5v) with a period and a frequency which depends on the number of inputs.

example:

```
V8 -21 0 pulse(0 5 0 1n 1n 2m 4m )
V9 -28 0 pulse(0 5 0 1n 1n 1m 2m )
```

For the MOSFET specifications, after researching we included the details of two matching MOSFETs created by Siliconix.

For the PMOS:

```
.model mypmos VDMOS(pchan Rg=3 Vto=-1.3 Rd=16m
+ Rs=12m Rb=20m Kp=30 Cgdmax=.6n Cgdmin=.14n
+ Cgs=.9n Cjo=.28n Is=28p ksubthres=.1
+ mfg=Siliconix Vds=-12 Ron=40m Qg=14n
```

For the NMOS:

```
.model mynmos VDMOS( Rg=3 Vto=1.4 Rd=8m
+ Rs=6m Rb=10m Kp=70 Cgdmax=.5n Cgdmin=.12n
+ Cgs=.8n Cjo=.24n Is=24p ksubthres=.1
+ mfg=Siliconix Vds=12 Ron=20m Qg=12n
```

Finally, for the testing we performed a transient simulation by using the command

```
.trans [max_time]
```

And here is a sample of simulations we ran for the generated NETLISTs

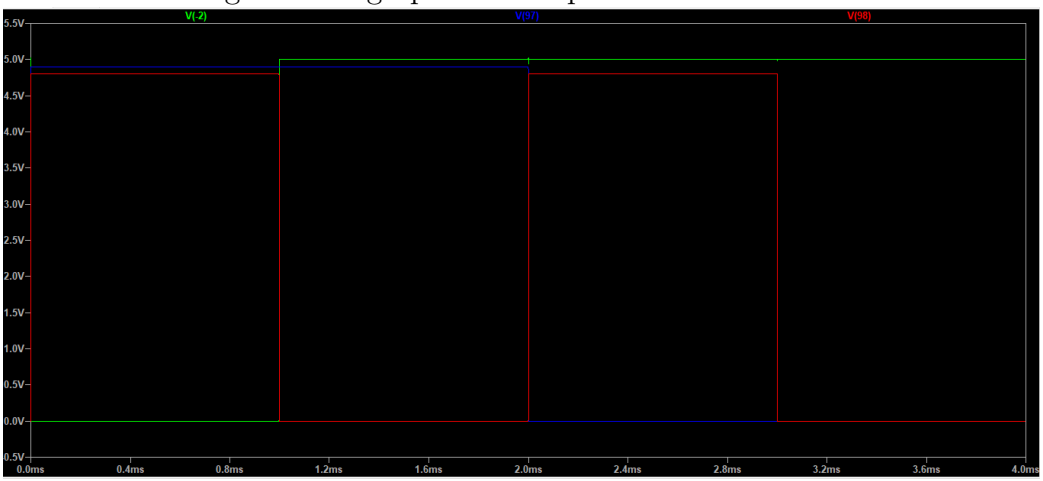
For Y = A'—B':

```
*this is the generated netlist for the expression a'|b'
Vdd -1 0 5v
M0 -2 97 -1 -1 mypmos
M1 -2 98 -1 -1 mypmos
M2 -2 97 -9 -9 mynmos
M3 -9 98 0 0 mynmos
V0 97 0 pulse(0 5 0 1n 1n 2m 4m )
V1 98 0 pulse(0 5 0 1n 1n 1m 2m )
*models discription
.model mynmos VDMOS( Rg=3 Vto=1.4 Rd=8m
+ Rs=6m Rb=10m Kp=70 Cgdmax=.5n Cgdmin=.12n
+ Cgs=.8n Cjo=.24n Is=24p ksubthres=.1
+ mfg=Siliconix Vds=12 Ron=20m Qg=12n

.model mypmos VDMOS(pchan Rg=3 Vto=-1.3 Rd=16m
+ Rs=12m Rb=20m Kp=30 Cgdmax=.6n Cgdmin=.14n
+ Cgs=.9n Cjo=.28n Is=28p ksubthres=.1
+ mfg=Siliconix Vds=-12 Ron=40m Qg=14n

.tran 4m
.backanno
.end
```

And here is the generated graph from ltSpice:



For Y = A'& B'—C':

```
Enter your logical expression!
 for AND use &  for OR use | for NOT use '
a'&b'|c'
*this is the generated netlist for the expression a'&b'|c'
Vdd -1 0 5v
M0 -5 97 -1 -1 mypmos
M1 -6 -5 -1 -1 mypmos
M2 -8 -6 -1 -1 mypmos
M3 -10 98 -1 -1 mypmos
M4 -11 -10 -1 -1 mypmos
M5 -2 -11 -8 -8 mypmos
M6 -15 99 -1 -1 mypmos
M7 -16 -15 -1 -1 mypmos
M8 -2 -16 -1 -1 mypmos
M9 -21 -20 -1 -1 mypmos
M10 -26 -25 -1 -1 mypmos
M11 -31 -30 -1 -1 mypmos
M12 -6 -5 0 0 mynmos
M13 -11 -10 0 0 mynmos
M14 -16 -15 0 0 mynmos
M15 -1 97 -20 -20 mynmos
M16 -21 -20 0 0 mynmos
M17 -2 -21 -23 -23 mynmos
M18 -1 98 -25 -25 mynmos
M19 -26 -25 0 0 mynmos
M20 -2 -26 -23 -23 mynmos
M21 -1 99 -30 -30 mynmos
M22 -31 -30 0 0 mynmos
M23 -23 -31 0 0 mynmos
V0 -30 0 pulse(0 5 0 1n 1n 2048m 4096m )
V1 -25 0 pulse(0 5 0 1n 1n 1024m 2048m )
V2 -20 0 pulse(0 5 0 1n 1n 512m 1024m )
V3 -16 0 pulse(0 5 0 1n 1n 256m 512m )
V4 -15 0 pulse(0 5 0 1n 1n 128m 256m )
V5 -11 0 pulse(0 5 0 1n 1n 64m 128m )
V6 -10 0 pulse(0 5 0 1n 1n 32m 64m )
V7 -6 0 pulse(0 5 0 1n 1n 16m 32m )
```

```
V8 -5 0 pulse(0 5 0 1n 1n 8m 16m )
V9 97 0 pulse(0 5 0 1n 1n 4m 8m )
V10 98 0 pulse(0 5 0 1n 1n 2m 4m )
V11 99 0 pulse(0 5 0 1n 1n 1m 2m )
*models discription
.model mynmos VDMOS( Rg=3 Vto=1.4 Rd=8m
+ Rs=6m Rb=10m Kp=70 Cgdmax=.5n Cgdmin=.12n
+ Cgs=.8n Cjo=.24n Is=24p ksubthres=.1
+ mfg=Siliconix Vds=12 Ron=20m Qg=12n

.model mypmos VDMOS(pchan Rg=3 Vto=-1.3 Rd=16m
+ Rs=12m Rb=20m Kp=30 Cgdmax=.6n Cgdmin=.14n
+ Cgs=.9n Cjo=.28n Is=28p ksubthres=.1
+ mfg=Siliconix Vds=-12 Ron=40m Qg=14n

.tran 4096m
.backanno
.end
```