

# **Capitolul 1**

## **Laboratorul 1**

### **Proiectarea circuitelor logice**

Scopul acestui laborator este sa proiectam libraria de porti logice cu care vom putea alcatai functii logice. Primul pas este sa proiectam si sa caracterizam cea mai simpla poarta logica: *inversorul*. Pasii detaliati care trebuie urmariti pentru a-l proiecta sunt descrisi in Anexa A.

In continuarea vom proiecta urmatoarele circuite:

1. NAND (Figure 1.1) cu testbech-ul in Figure 1.2;
2. NOR (Figure 1.3) cu testbech-ul in Figure 1.4;
3. Transmission Gate (Figure 1.5) cu testbech-ul in Figure 1.6;
4. Transmission Gate cu *Enable*(Figure 1.7) cu testbech-ul in Figure 1.8;
5. Three state inverter cu *Enable*(Figure 1.9) cu testbech-ul in Figure 1.10;
6. SR Latch (Figure 1.11) cu testbech-ul in Figure 1.12;

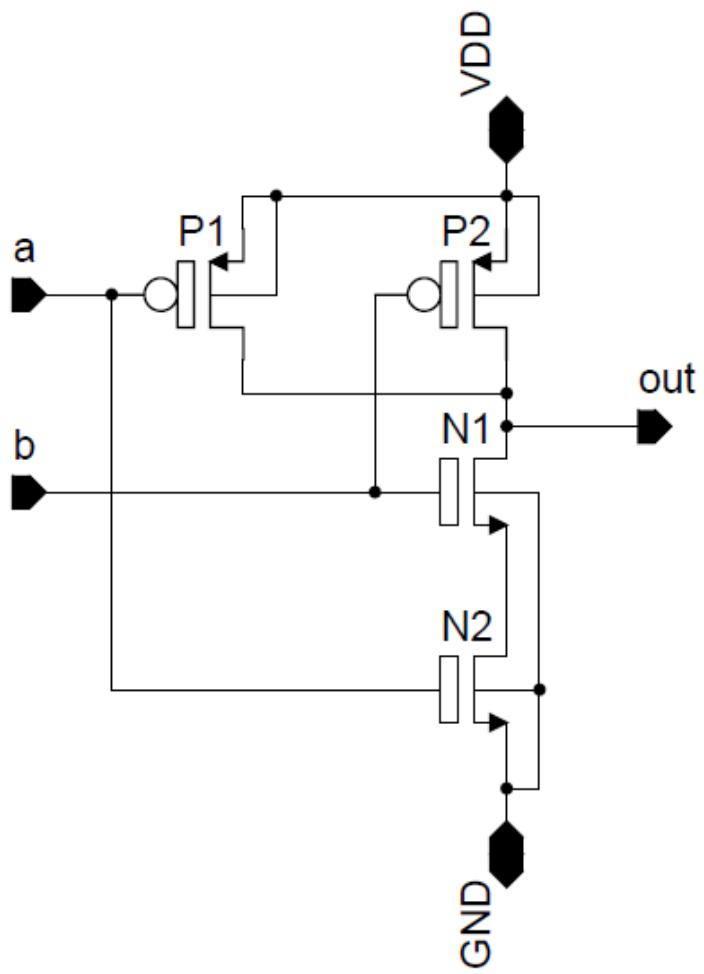


Figura 1.1: NAND Gate.

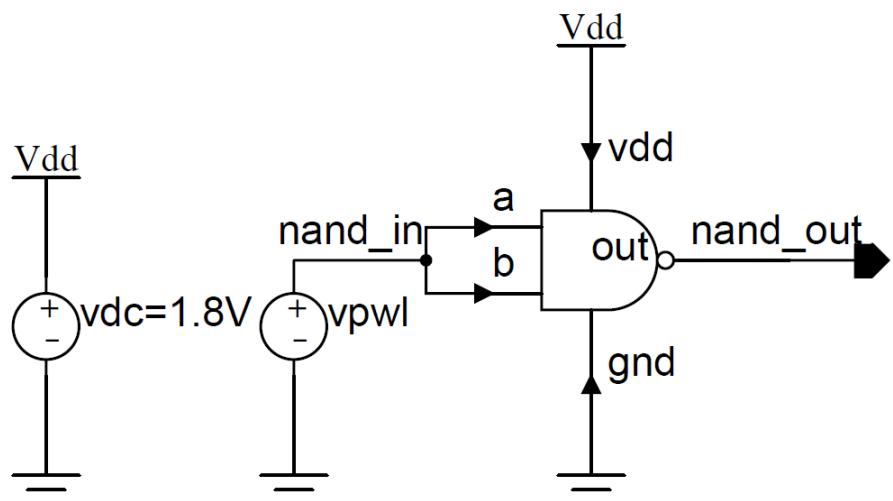


Figura 1.2: NAND Gate Testbench.

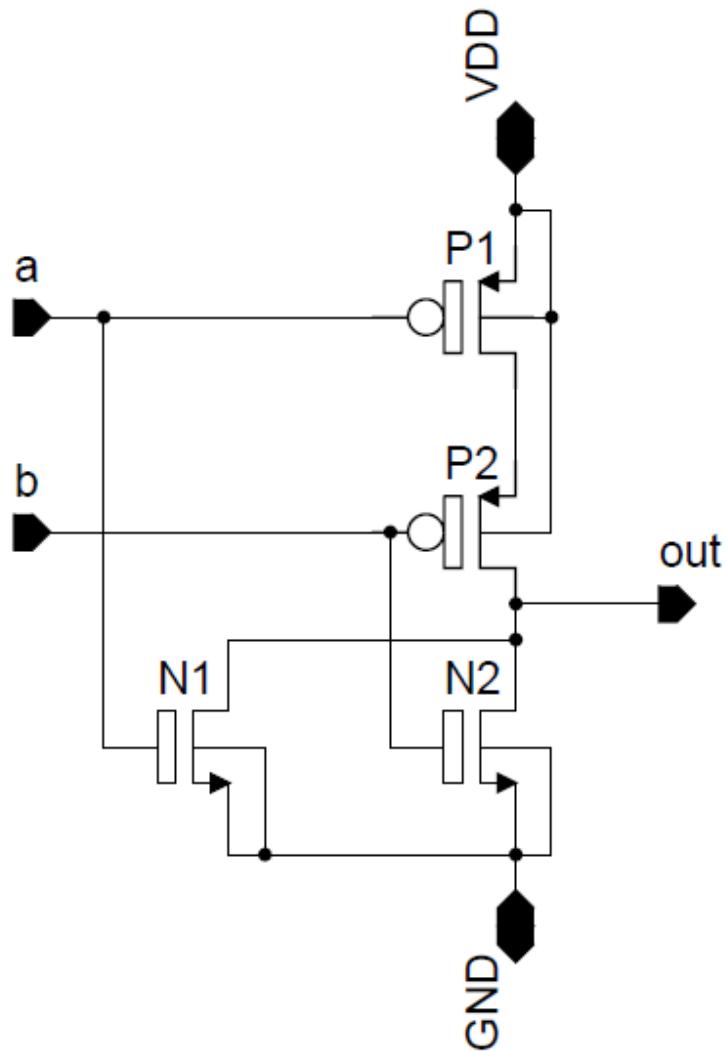


Figura 1.3: NOR Gate.

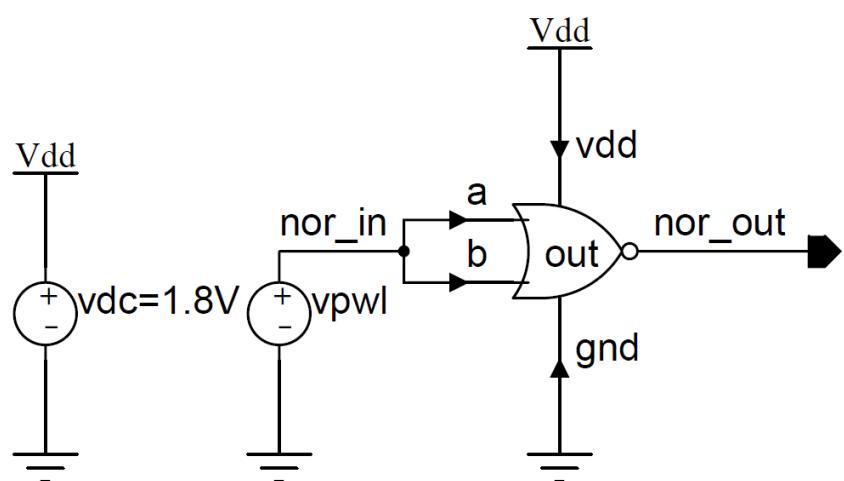


Figura 1.4: NOR Gate Testbench.

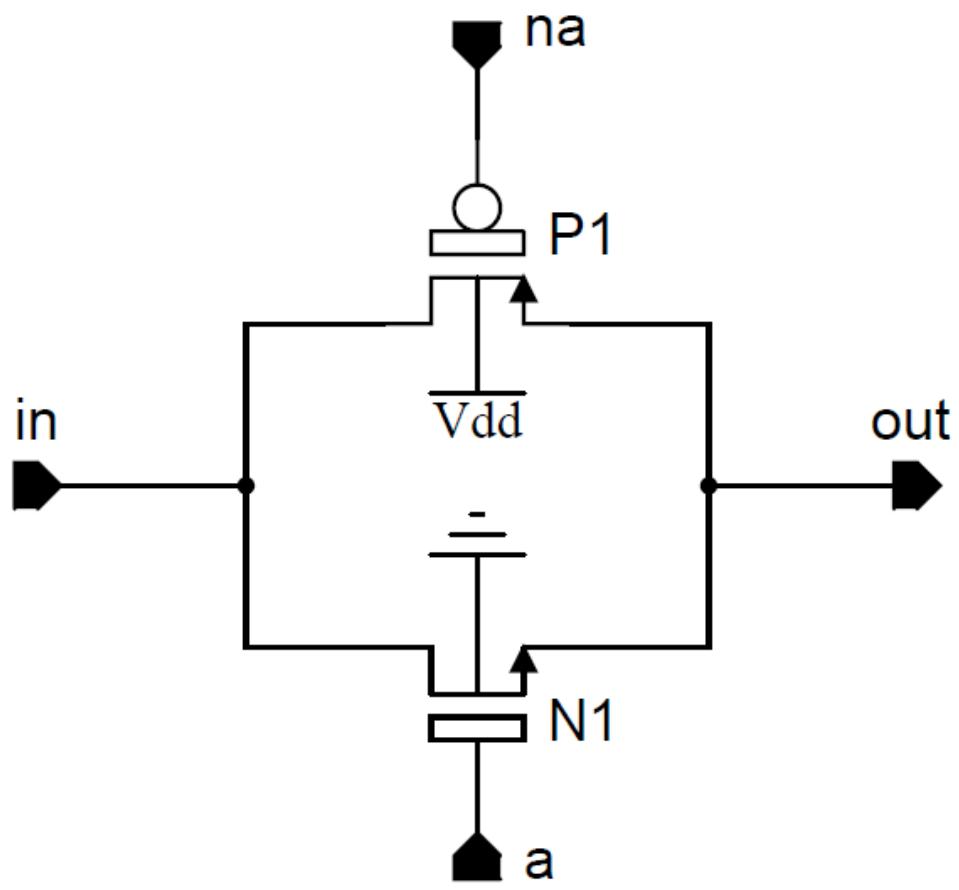


Figura 1.5: Transmission Gate (TG).

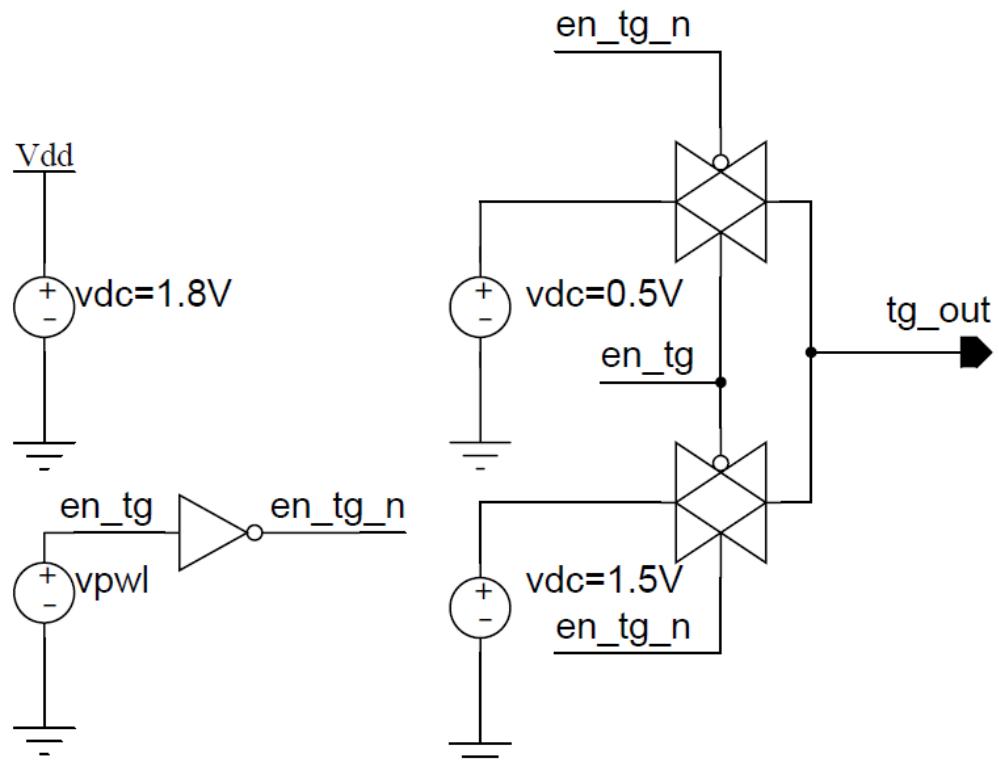


Figura 1.6: TG Testbench.

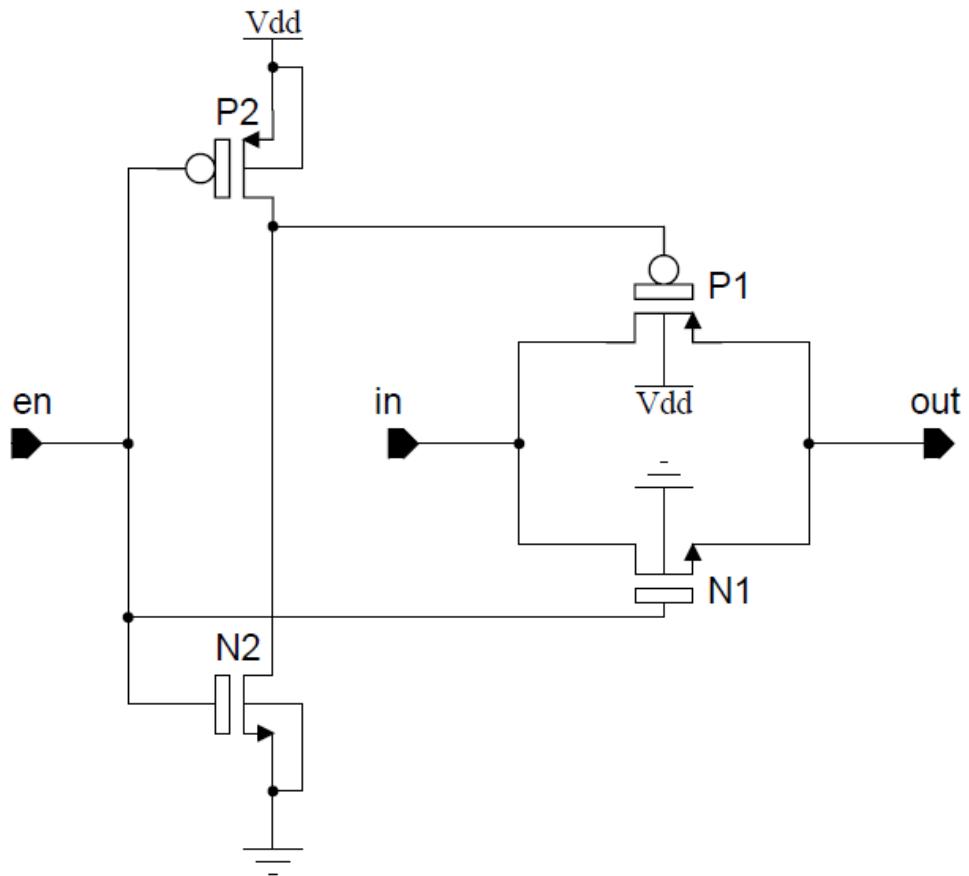


Figura 1.7: Transmission Gate cu Enable.

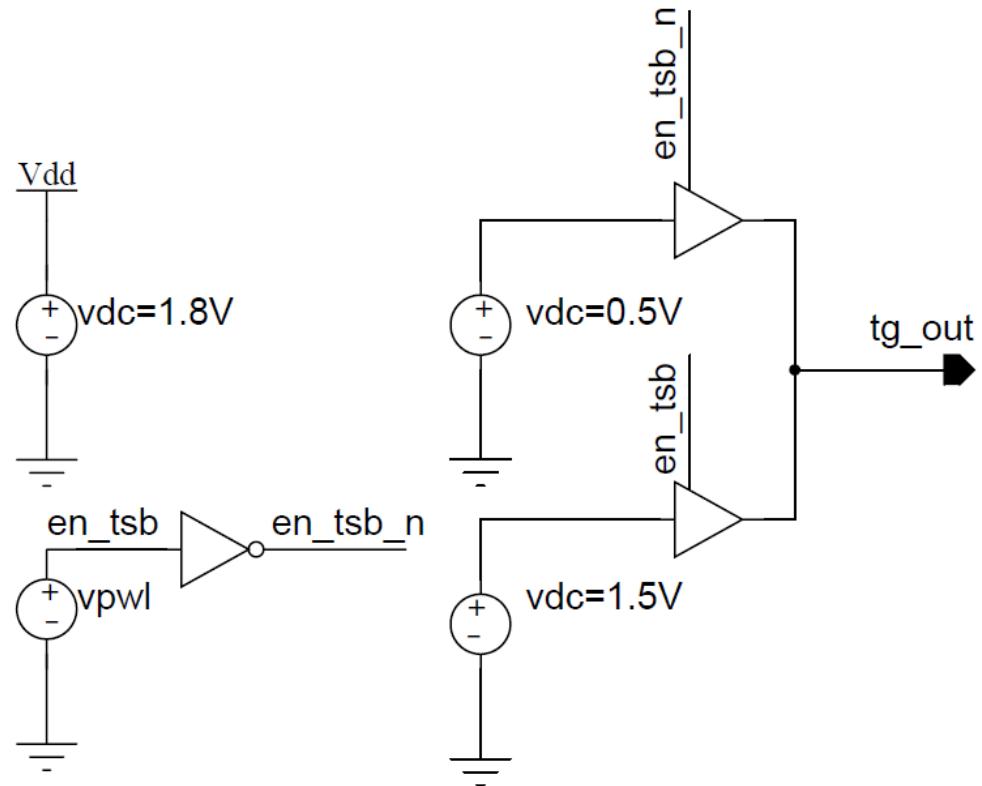


Figura 1.8: TG cu Enable Testbench.

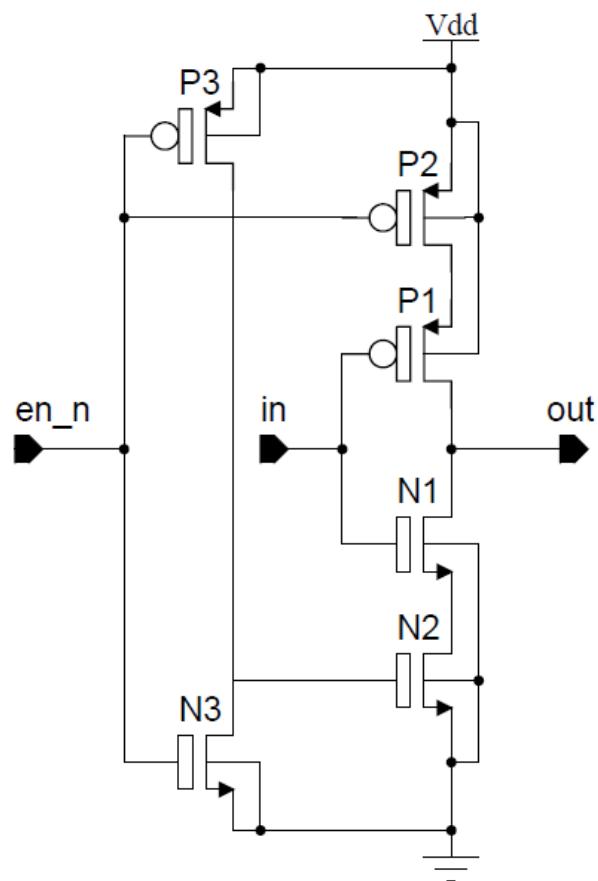


Figura 1.9: Three state inverter.

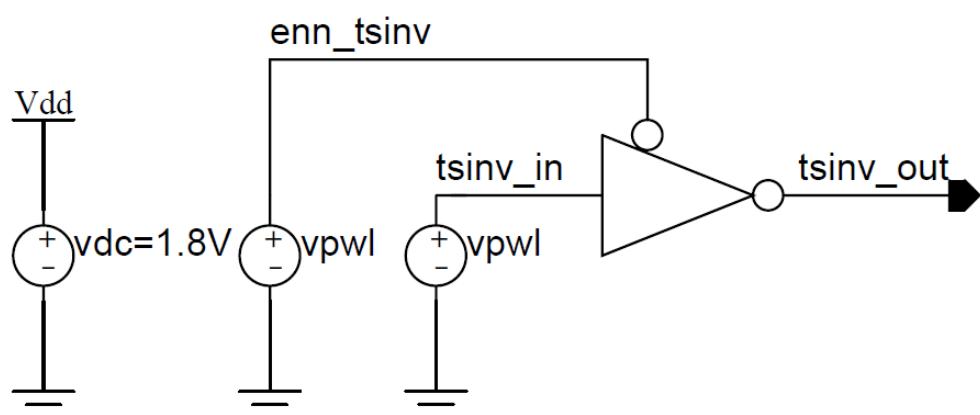


Figura 1.10: Three state inverter Testbench.

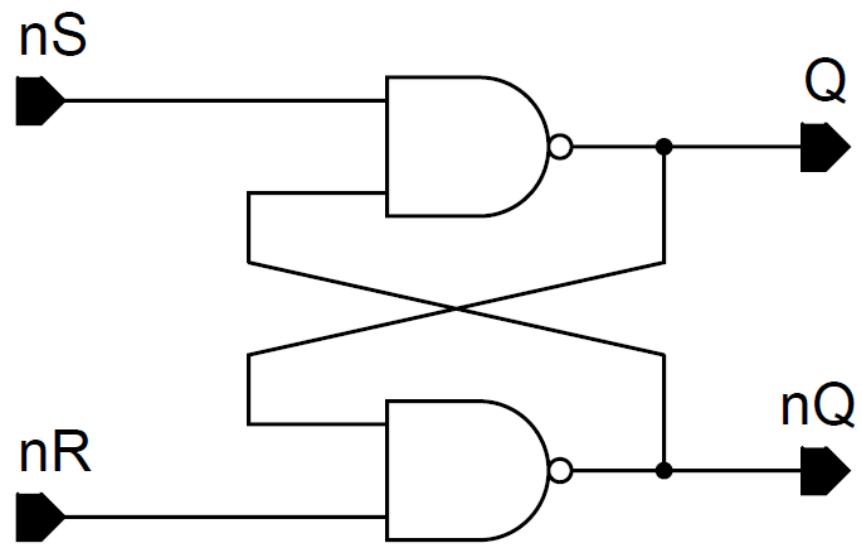


Figura 1.11: SR Latch.

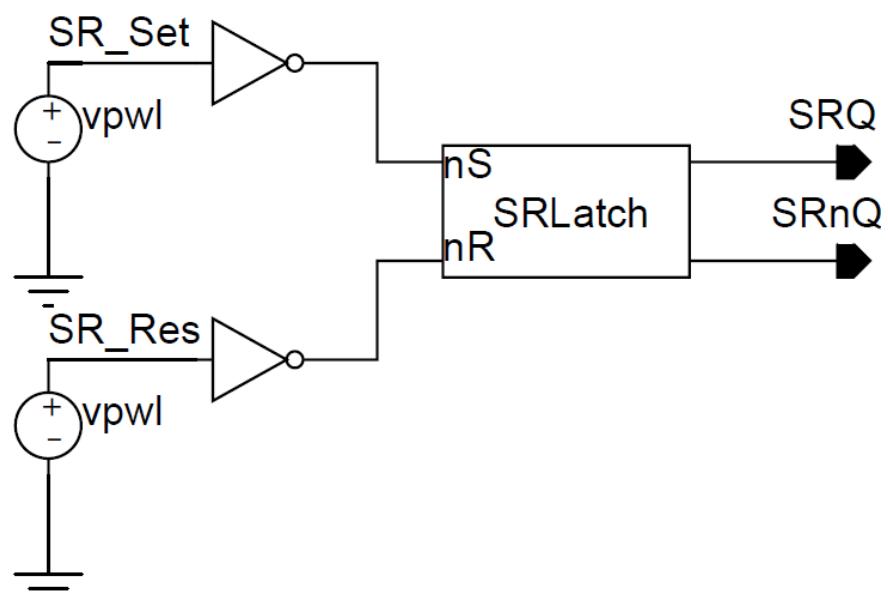


Figura 1.12: SR Latch Testbench.

## Capitolul 2

### Laboratorul 2

Scopul acestui laborator este să înțelegem funcționarea portilor logice proiectate în Chapter 1 și să îmbunătățim funcționarea acestora.

#### 2.1 Proiectarea inversorului cu $t_{pdr} = t_{pdf}$

Primul pas este să efectuam o analiză *transient* pentru inversorul proiectat în Chapter 1. Scopul acestei simulații este să calculăm timpul de propagare al inversorului - $t_{pd}$ , adică distanța în timp dintre semnalul de ieșire și semnalul de intrare atunci când valoarea acestora ajunge la  $VDD/2$  (vezi Figura 2.1).

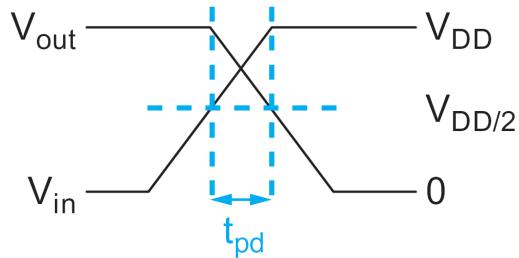


Figura 2.1: Inversor: Timpul de propagare -  $t_{pd}$ .

Observăm în Figura 2.2 diferența dintre  $t_{pd}$  atunci când ieșirea comută din  $VDD$  în  $GND$  ( $t_{pdf}$ ) și atunci când ieșirea comută din  $GND$  în  $VDD$  ( $t_{pdr}$ ). Această diferență este cauzată în mare parte de diferența mobilității purtătorilor de sarcină:  $\mu_n > \mu_p$ .

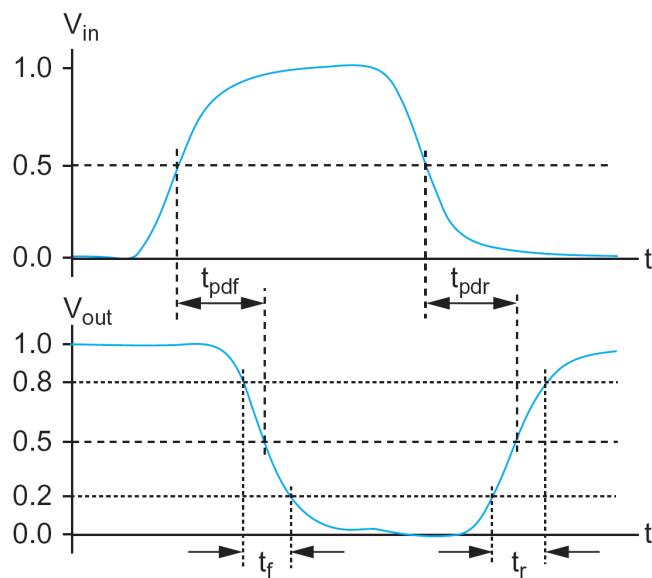


Figura 2.2: Inversor: Timpul de propagare atunci când ieșirea crește sau scade -  $t_{pdr}$  și  $t_{pdf}$ .

Tinând cont de modelul RC simplificat al tranzistorului MOS prezentat în Figura 2.3, modelați raportul  $W/L$  al tranzistorului nMOS și al celui pMOS dintr-un inversor pentru a obține  $t_{pdr} = t_{pdf}$ . Este de preferat să păstrati lungimea canalului ( $L$ ) la valoarea minimă pentru ambele dispozitive, i.e. 180 nm pentru  $gpdk180$ , modificând doar lățimea ( $W$ ).

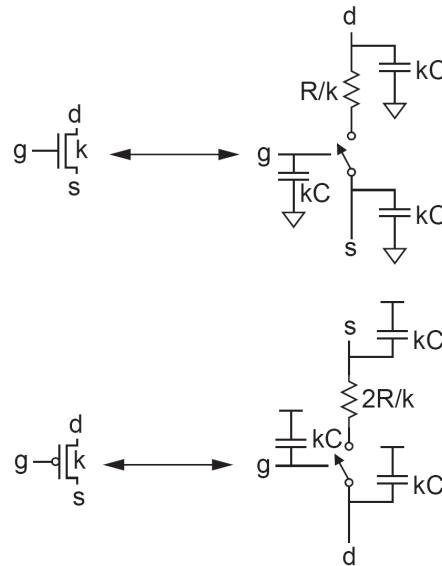


Figura 2.3: nMOS și pMOS: Model RC simplificat.

Pentru a identifica rapid valorile optime ale parametrilor  $W_n$  și  $W_p$  se poate rula o analiză transientă parametrică. Aceasta analiză se obține urmând urmatorii pași:

1. se modifică parametrul  $W$  al dispozitivelor *nmos* și *pmos* folosind tasta *q*. Pentru dispozitivul *nmos* vom folosi parametrul *m*, iar pentru dispozitivul *pmos* parametrul *n*. În Figura 2.4 se regăsește exemplul pentru dispozitivul *nmos*
2. în meniul *Analog Design Environment* adăugăm variabile nou create folosind meniul *Variables → Copy from Cellview* (vezi Figura 2.5).

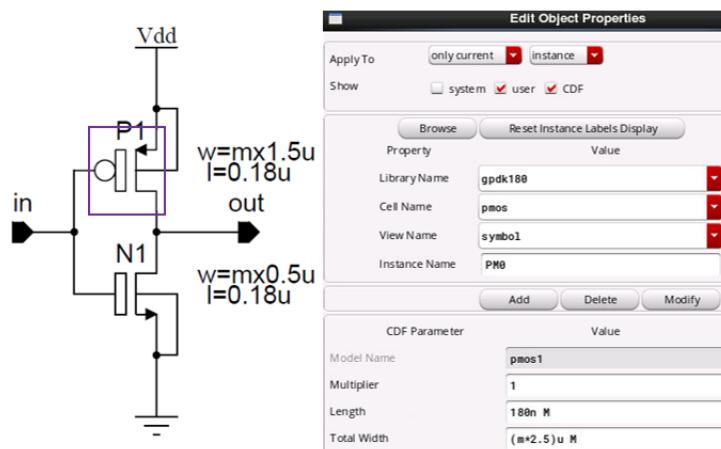


Figura 2.4: nMOS și pMOS: Modificare parametrului  $W$ .

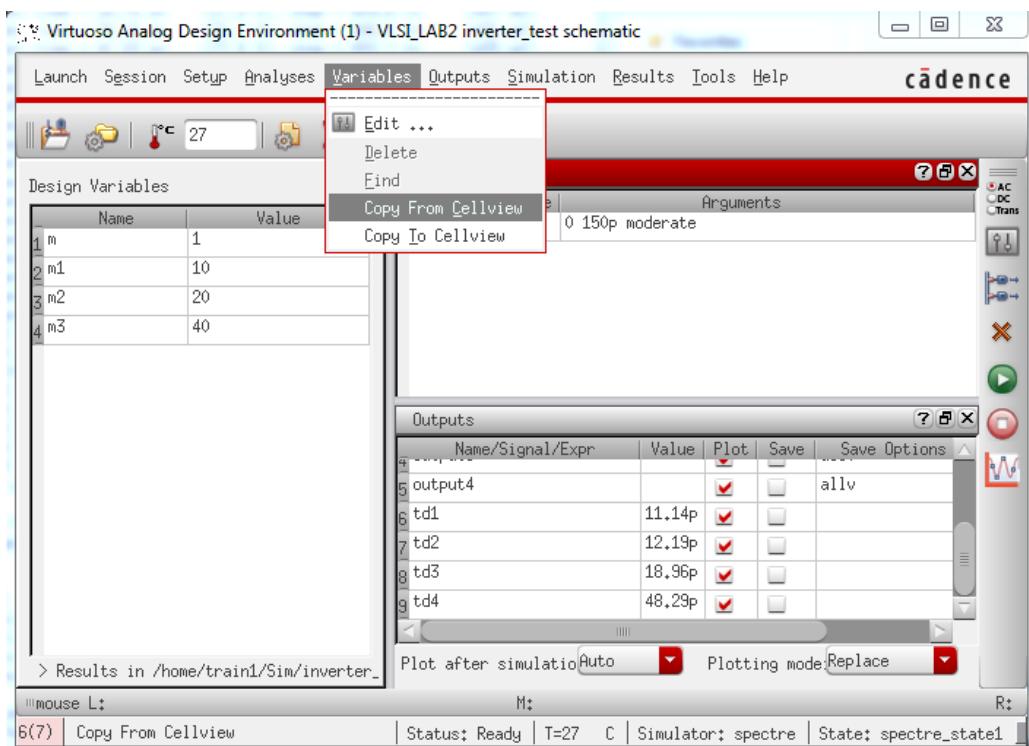


Figura 2.5: Inverter: Copierea variabilelor nou create din schema.

### 2.1.1 Introducere în folosirea calculatorului integrat în Cadence Virtuoso

Pentru calcularea automată a parametrilor  $t_{pdr}$  și  $t_{pdf}$  se poate folosi calculatorul integrat. După identificarea raportului  $W_p/W_n$  pentru care  $t_{pdr} = t_{pdf}$  putem începe simularea inversoarelor cu *drive strength* diferit. În scop didactic am considerat 4 configurații de inversor diferențite (4 scheme diferențite) în care am folosit 4 factori de multiplicare  $m - m_3$  pentru 4 celule diferențite de inversor, iar pentru *pMOS* acesta este înmulțit cu raportul optim  $W_p/W_n$  (vezi Figura 2.5). Schema celor 4 inversoare se găsește în Figura 2.6. Proprietățile sursei *vpwl* folosite în aceasta simulare se regăsesc în Figura 2.7. Notă: Nu uitati să alimentați circuitul la 1.8V.

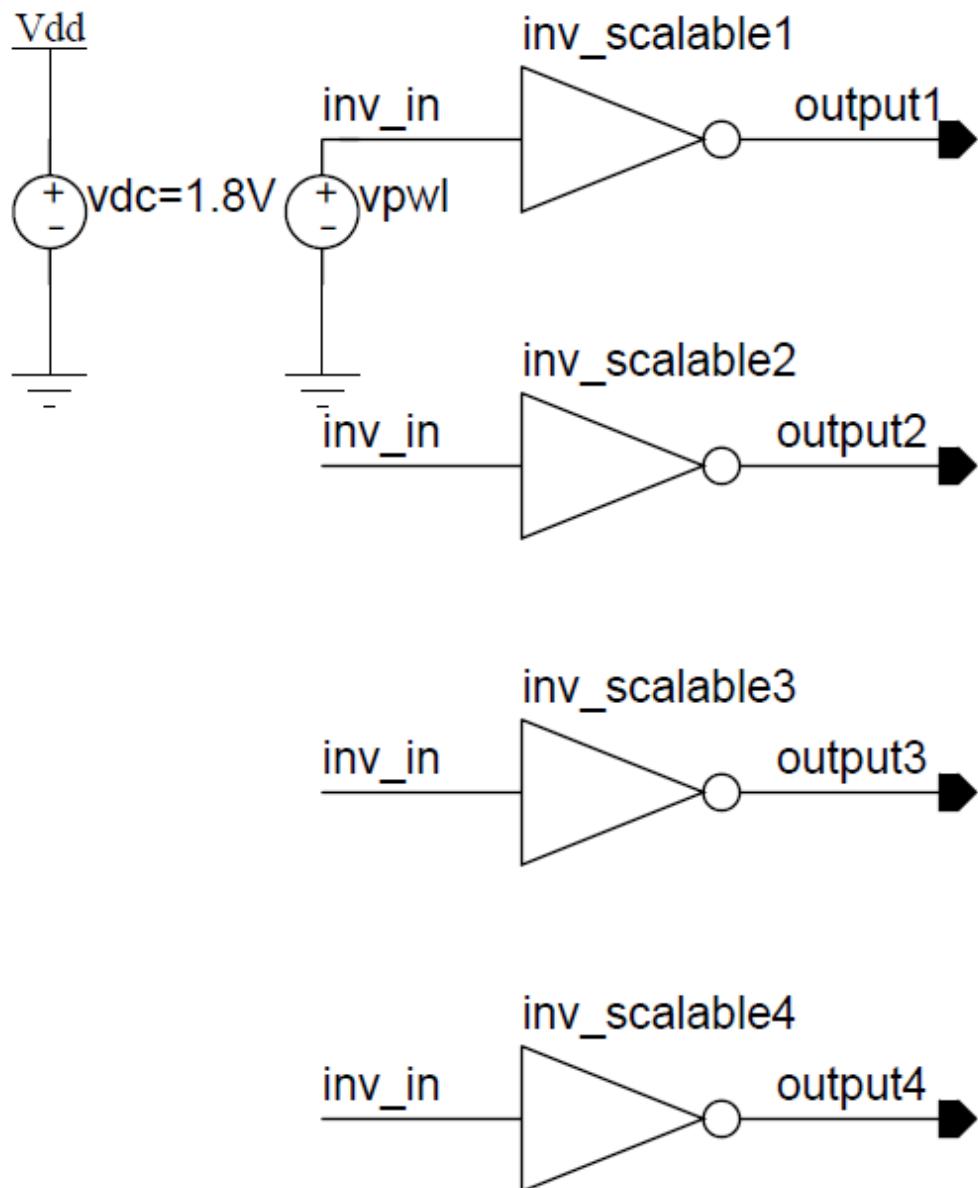


Figura 2.6: Inverter: Schema a 4 inversoare cu drive strength diferit, parametrii  $m$  si  $n$  diferenți.

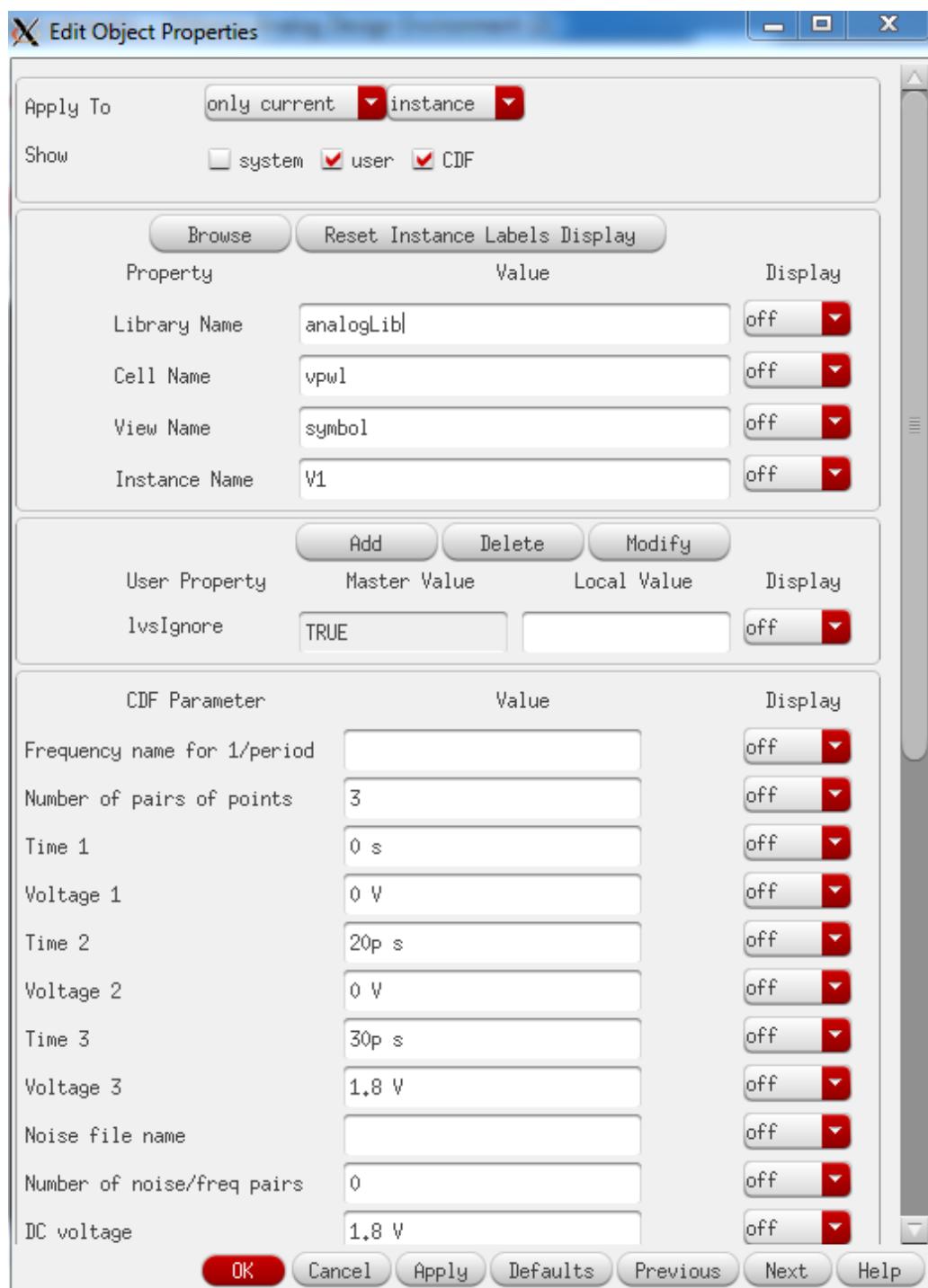


Figura 2.7: Inverter: Proprietatile sursei vpwl.

Pentru a masura automat  $t_{pd}$  vom folosi calculatorul integrat urmand secvența următoare de comenzi:

1. Pornim calculatorul din meniul *Tools · Calculator* (vezi Figure 2.8);
2. Deoarece ne dorim variația unei tensiuni în timp, folosim funcția *vt* (*v* deoarece ne interesează o tensiune, iar *t* deoarece semnalul este în timp), *Left Click* pe funcție, după care selectăm semnalul dorit din schema tot cu *Left Click*, iar la final apasam *Esc* (vezi Figure 2.9);
3. Urmatorul pas este să folosim funcția predefinită *cross* din submeniul *Function Panel* (vezi Figure 2.10) pentru a afla timpul la care semnalul comută din 0 logic în 1 logic;
4. Funcția *cross* are 3 parametrii (vezi Figure 2.11), numele semnalului căruia vrem să îl aplicăm aceasta funcție, e.g., *VT("/output1")* pentru semnalul *output1*, pragul la care vrem să măsurăm timpul, în cazul nostru *VDD/2*, adică 0.9V, și ultimul parametru este numărul frontului pe care facem această măsurare.
5. Dupa apasarea tastei *OK* functia va aparea in meniul principal al calculatorului (vezi Figure 2.12).
6. Functia finala pentru a calcula  $t_{pd}$  va fi compusa din 2 functii *cross* aplicate atat semnalului de la iesire cat si semnalului de la intrare (vezi Figure 2.13).
7. In final putem folosi acest calcul pentru toate cele 4 configurații de inversoare folosind meniul *Outputs · Setup* (vezi Figure 2.14), *Get Expression* pe care o să o numim *td1* (vezi Figure 2.15). Ulterior, modificând semnalul de iesire *output1* din funcția  $t_{pd}$  cu *output2*, *output3*, *output4*, și numele funcției noi create cu *tpd2*, *tpd3*, *tpd4*, putem predefini funcțiile care calculează  $t_{pd}$  pentru toate cele 4 inversoare, afișându-ne la sfârșitul simulației valoarea  $t_{pd}$  exact ca în Figure 2.14.

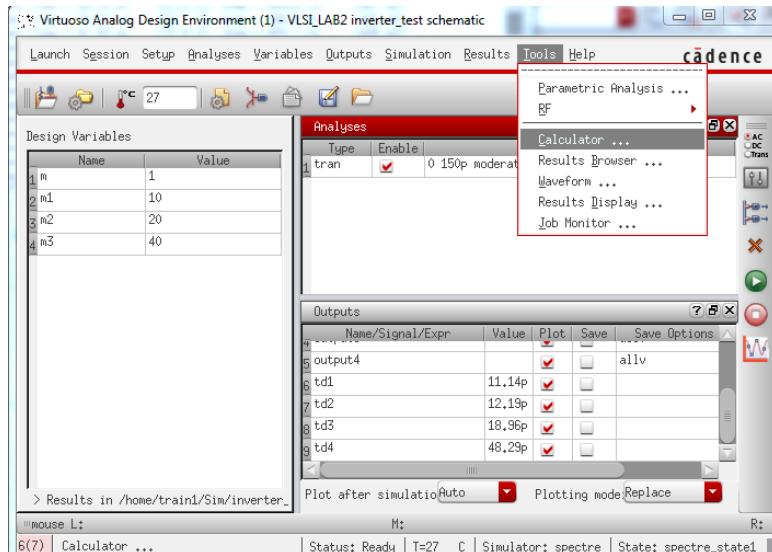


Figura 2.8: ADE-L: Pornirea calculatorului.

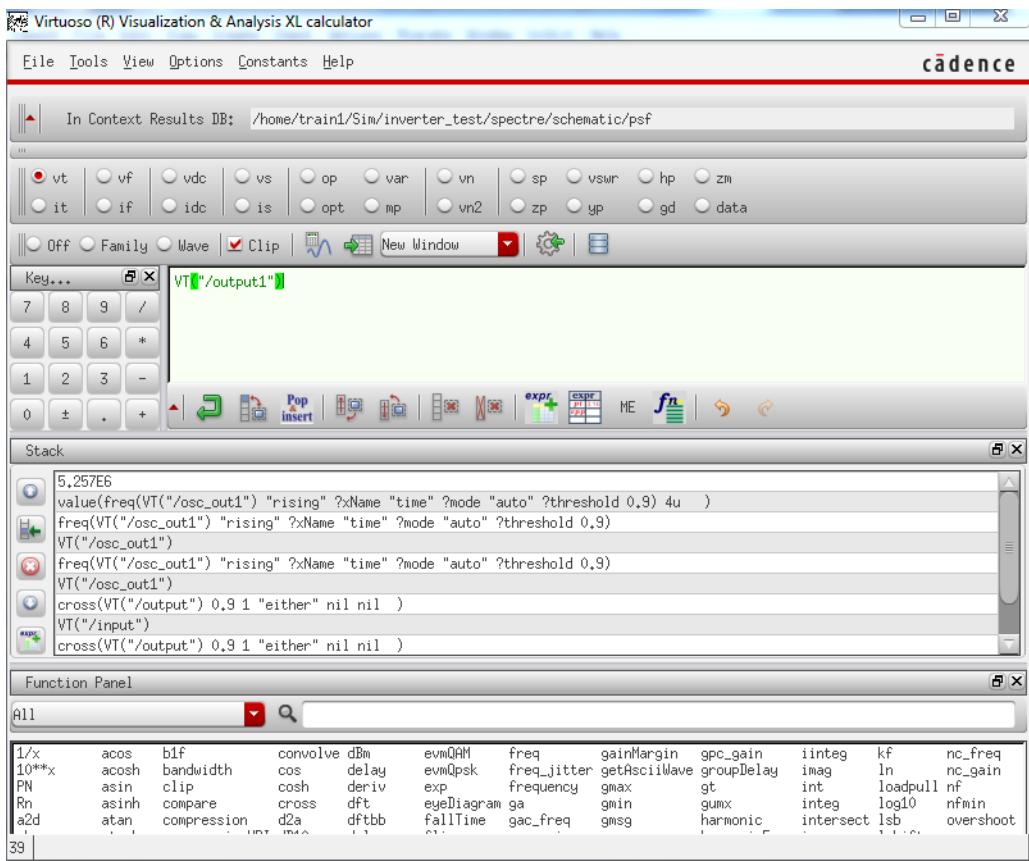


Figura 2.9: ADE-L: Selectarea semnalului.



Figura 2.10: ADE-L Calculator: Functia cross.



Figura 2.11: ADE-L Calculator: Parametrii functiei cross.

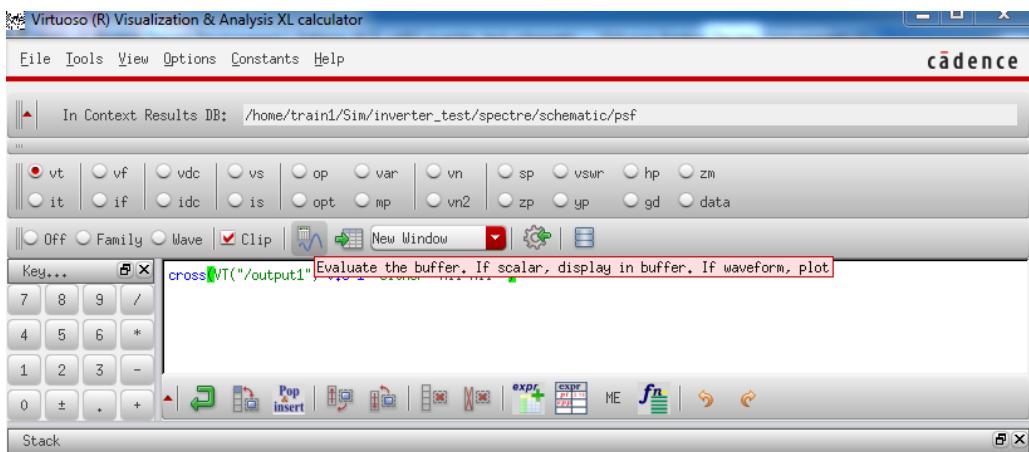


Figura 2.12: ADE-L Calculator: Aplicarea functiei cross pentru semnalul *output1*.

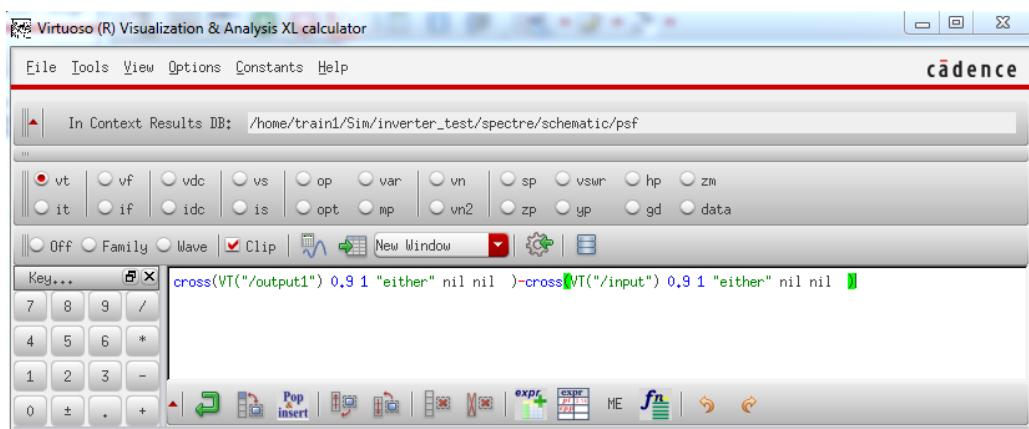


Figura 2.13: ADE-L Calculator: Aplicarea functiei cross pentru  $t_{pd}$ .

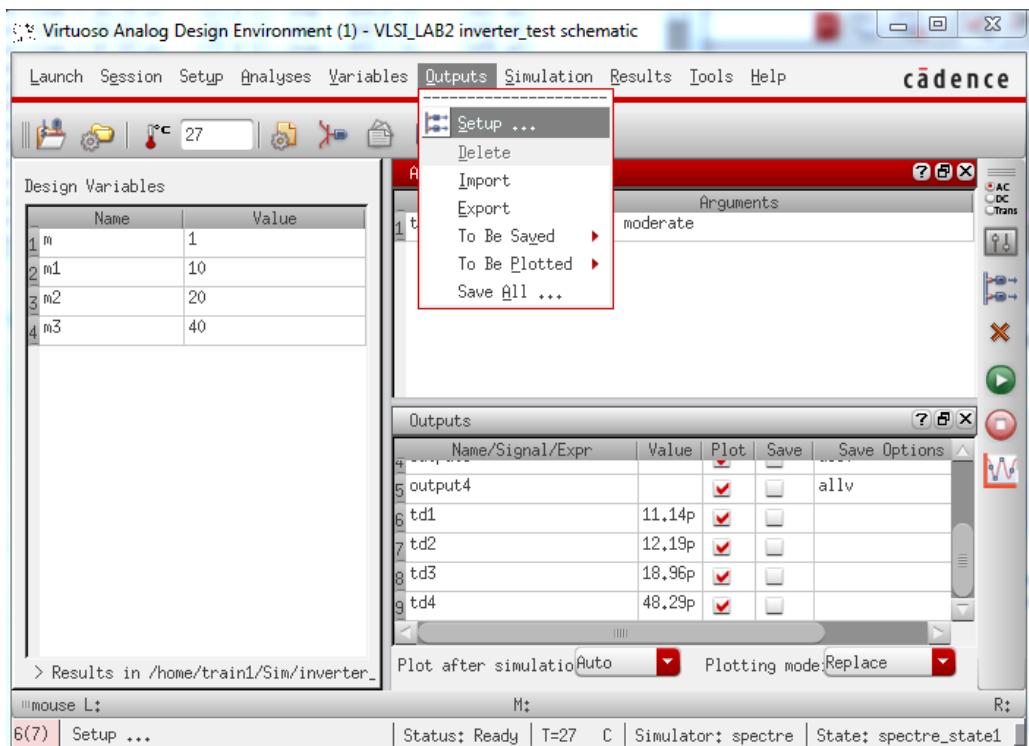


Figura 2.14: ADE-L: Predefinirea functiei cross pentru calculul  $t_{pd}$  multiplu.

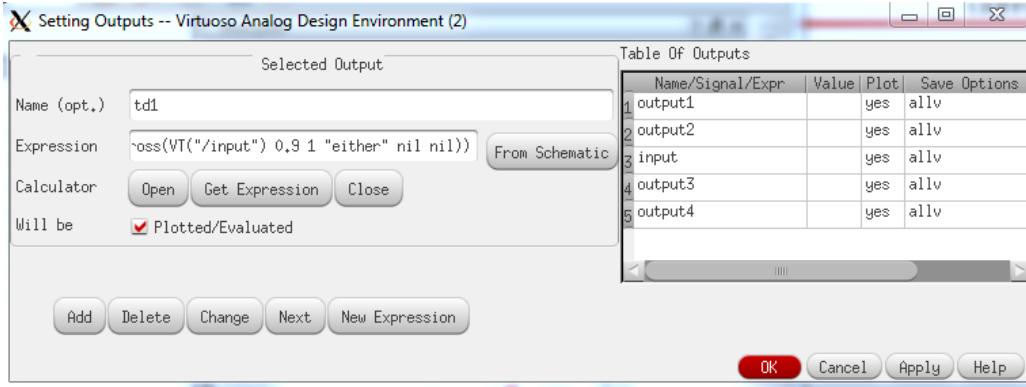


Figura 2.15: ADE-L: Predefinirea functiei cross pentru calculul  $t_{pd}$  multiplu.

## 2.2 Proiectarea si evaluarea unei porti NOR cu 2 intrari

In această secțiune vom proiecta poarta NOR din Figura 2.16 avand lățimea tranzistorelor,  $W$ , aleasă în aşa fel încât să obținem rezistențele echivalente pentru trazițiile cele mai defavorabile, i.e., worst case rise/fall -  $t_{pdr}/t_{pdf}$  -, egale cu rezistențele unui inversor unitate și egale între ele. Se menționează că pentru  $\mu_n = 2 \cdot \mu_p$ , inversorul unitate are raportul lățimilor dat de urmatoarea ecuație:

$$\frac{W_p}{W_n} = \frac{8 \cdot \lambda}{4 \cdot \lambda} = \frac{720 \text{ nm}}{360 \text{ nm}}$$

Va fi necesar să **adaptați** ecuația la **raportul mobilităților** din tehnologia utilizată în laborator, i.e., *gpdk180*.

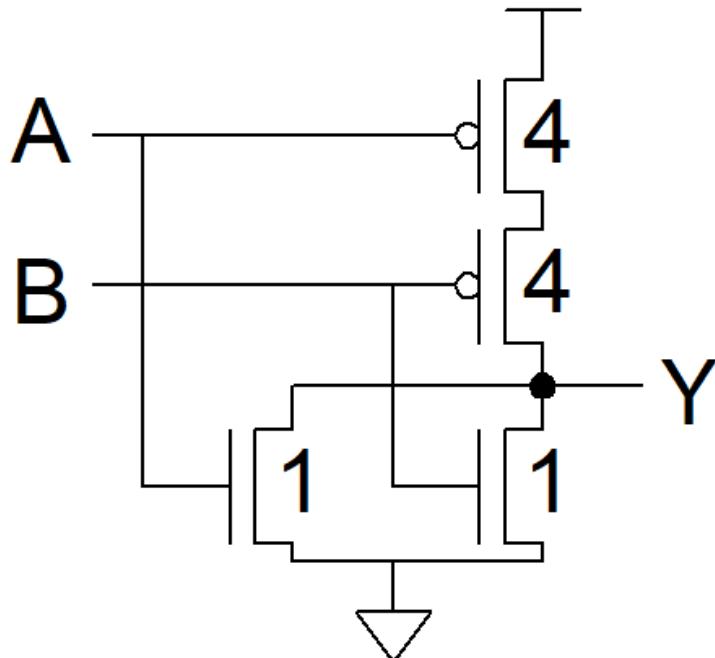


Figura 2.16: Poarta NOR cu 2 intrari pentru tehnologia ideală în care  $\mu_n = 2 \cdot \mu_p$ ,  $k_n=1$  și  $k_p=4$

## 2.3 Proiectarea si evaluarea inversorului cu efect de substrat

Efectul de substrat reprezintă modificarea tensiunii de prag a unui tranzistor *nMOS* ( $V_{TN}$ ) aplicând o diferență de potențial între sursa și substratul acestuia ( $V_{SB}$ ). Relația (2.1) prezintă dependența  $V_{TN}$  în funcție de  $V_{SB}$ .

$$V_{TN} = V_{T0} + \gamma \cdot (\sqrt{|V_{SB} - 2 \cdot \phi_F|} - \sqrt{|2 \cdot \phi_F|}) \quad (2.1)$$

Acest fenomen se poate observa proiectând inițial o schemă nouă a inversorului *CMOS* exact ca în Figura 2.17 urmărind următorii pași, astfel:

1. Se adaugă două surse de tensiune *DC* pentru bulk-ul fiecarui tranzistor cu valoarea tensiunii  $DC = v_{bg}$ ,
2. Conexiunile bulk-urilor se redenumesc  $v_{bn}$  (nMOS) și  $v_{bp}$  (pMOS).

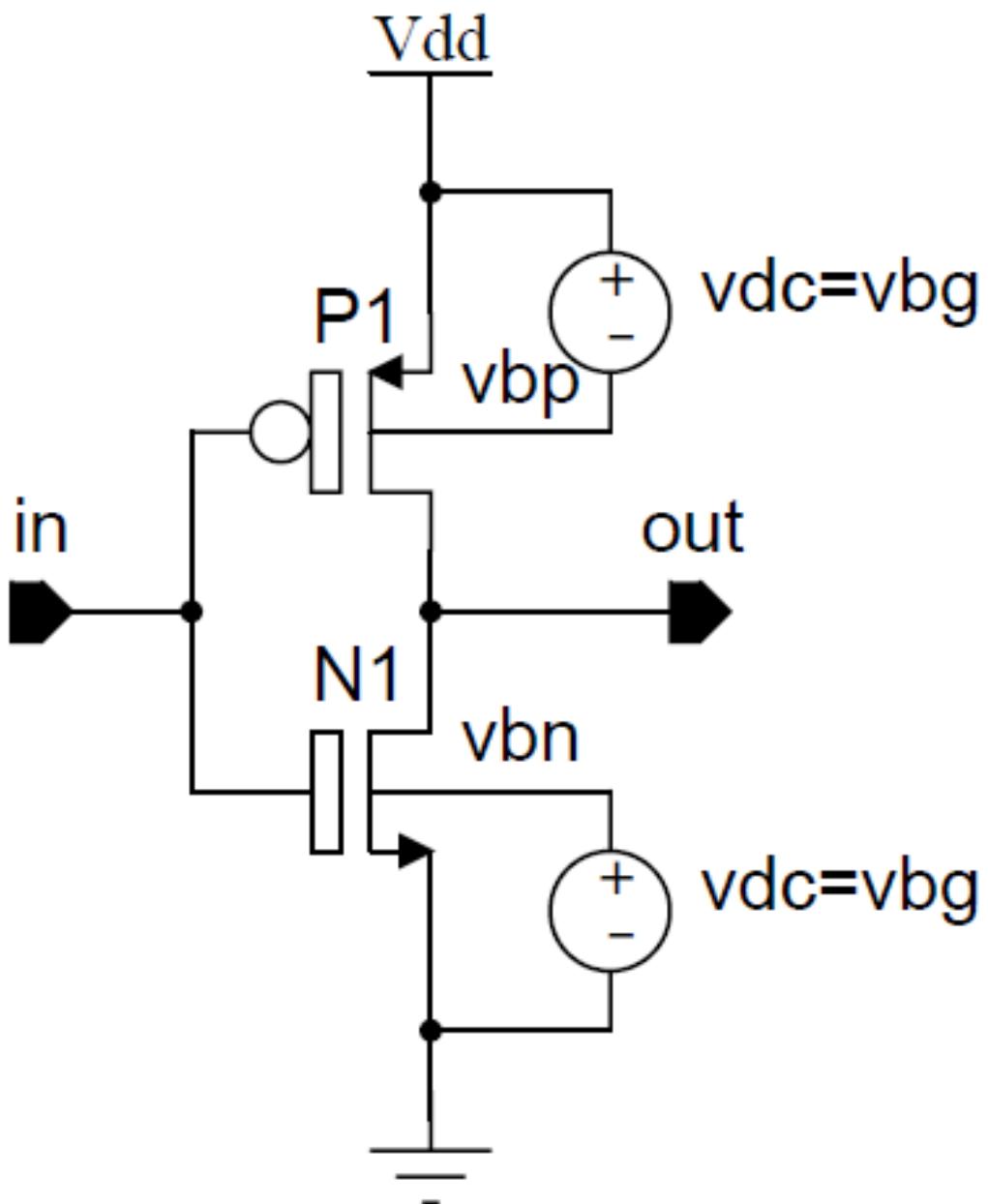


Figura 2.17: Inversorul CMOS: Simularea efectului de substrat.

Următorul pas este să desenăm o nouă schemă de simulare care inglobează atât simbolul noului inversor, cât și simbolul inversorului de referință, adică cel care nu are implementat efectul de substrat (vezi *testbench*-ul din Figura 2.18).

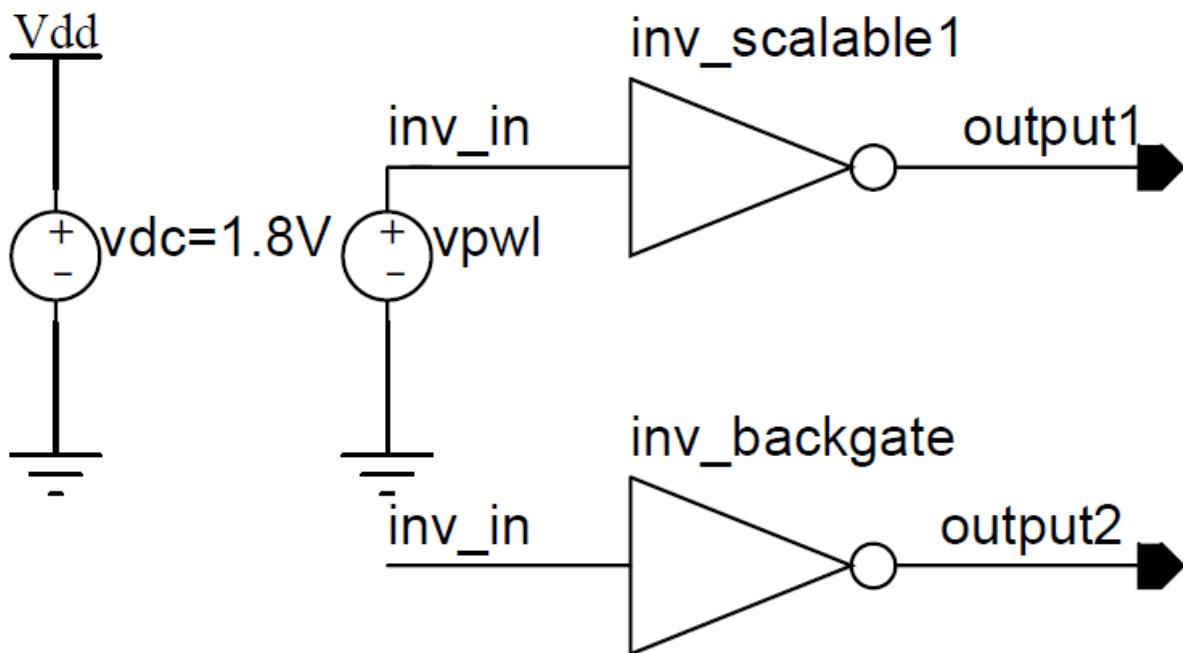


Figura 2.18: Inversorul CMOS: Testbench-ul pentru efectul de substrat.

În final se utilizează o analiză *transient* în care se urmărește modificarea pragului de comutație al inversorului cu și fără efectul de substrat. Valoarea pragului de comutație se poate identifica pe formele de undă ale semnalelor de intrare și ieșire sau se poate calcula folosind formula din Figura 2.19 pentru ambele semnale de la ieșire, *output1* și *output2*. Rezultatele simulării se pot vedea în Figura 2.20

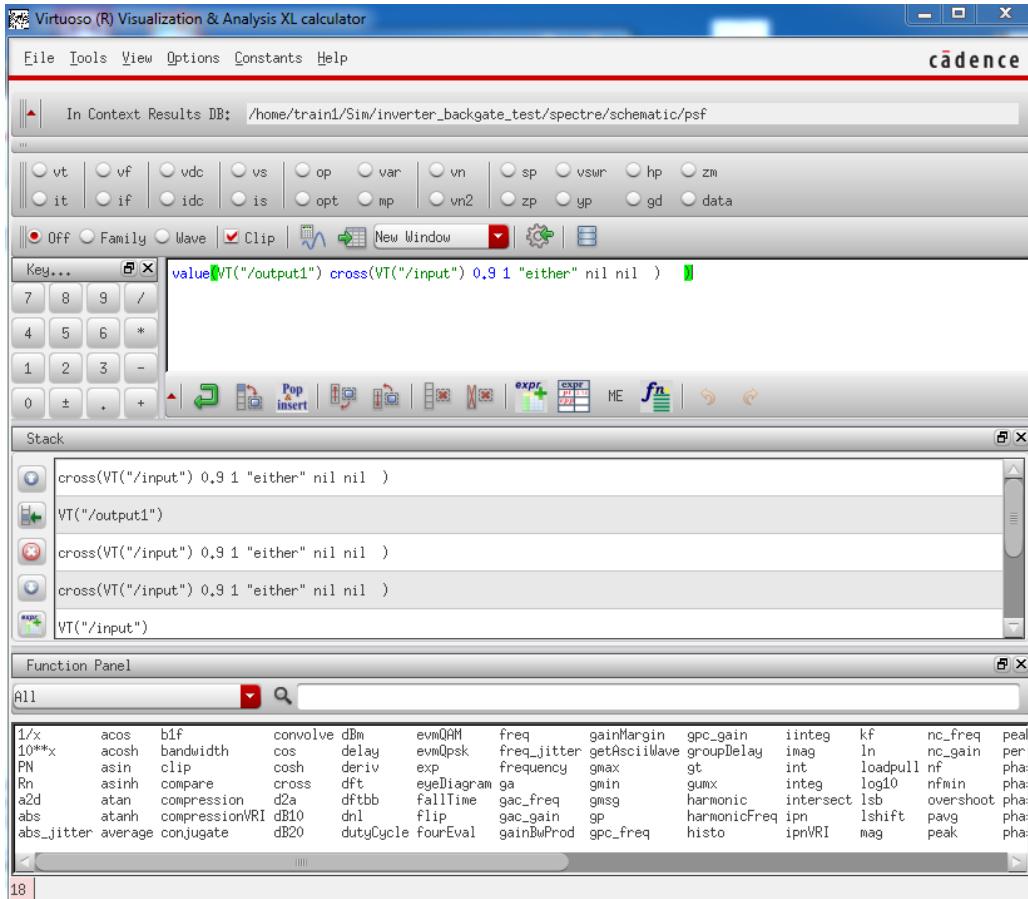


Figura 2.19: Inversorul CMOS: Testbench-ul pentru efectul de substrat.

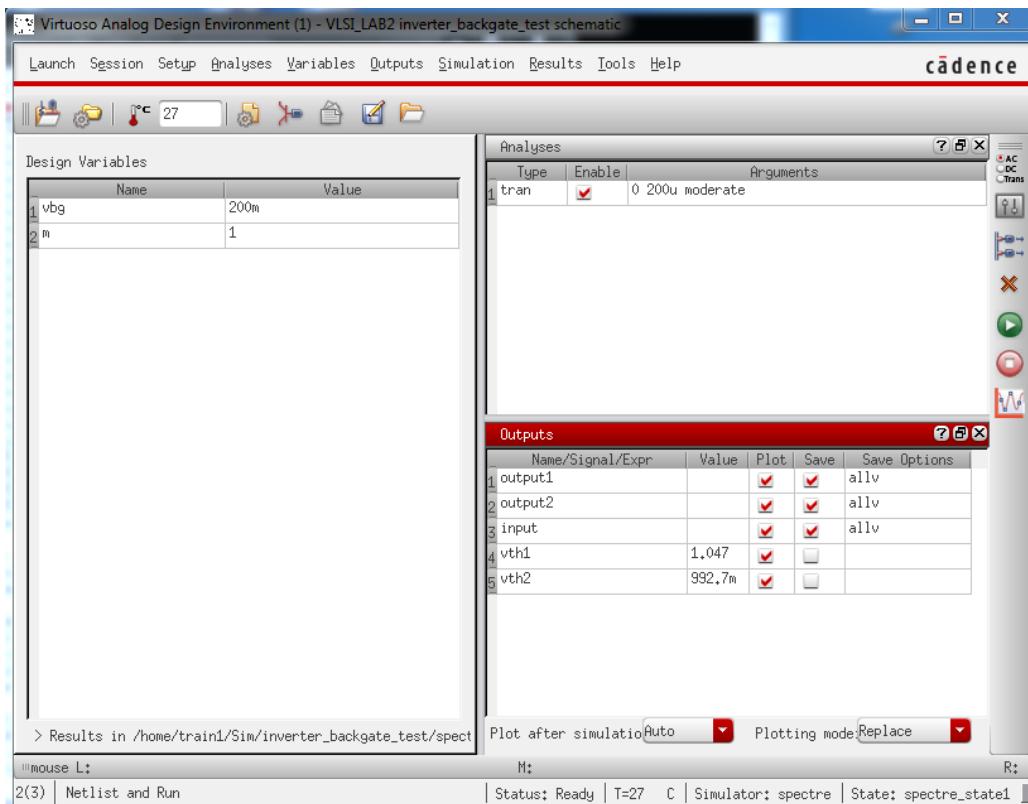


Figura 2.20: Inversorul CMOS: Calculul pragului unui inversor cu si fara efect de substrat.

## 2.4 Proiectarea si evaluarea oscilatorului de tip inel *Ring Oscillator*

Un oscilator in inel este compus dintr-un număr impar de inversoare conectate in inel. Iesirea oscilatorului balează între două nivele de tensiune, nivelul de 0 logic și nivelul 1 logic al inversorului. Iesirea ultimului inversor fiind conectată la intrarea primului inversor.

Oscilatorul inel face parte din clasa oscilatoarelor *time delay*. Acestea sunt compuse dintr-un amplificator inversor la iesirea căruia î se adaugă un element de intârziere (intârziere *RC*). Amplificatorul trebuie să aibă un castig mai mare ca 1 la frecventa la care dorim să functioneze. Atât timp cât câștigul amplificatorului este supraunitar și cu semn schimbat, ieșirea își va schimba polaritatea în comparație cu intrarea. Datorită amplificării, valoarea tensiunii de la ieșire se va schimba proporțional cu schimbarea semnalului de la intrare înmulțit cu amplificarea inversorului. Rezultatul amplificărilor succesive din lanțul de inversoare va duce la generarea unui semnal dreptunghiular.

Frecvența de oscilație al unui astfel de oscilator este data de Ecuatie 2.2.

$$f = \frac{1}{2 \cdot t \cdot n} \quad (2.2)$$

unde  $t$  este intârzierea unui stagiu de inversare și  $n$  este numărul total al stagilor de inversare.

La acest laborator vom proiecta un oscilator in inel denumit *current starving ring oscillator*. Schema unui astfel de oscilator este prezentata in Figure 2.21.

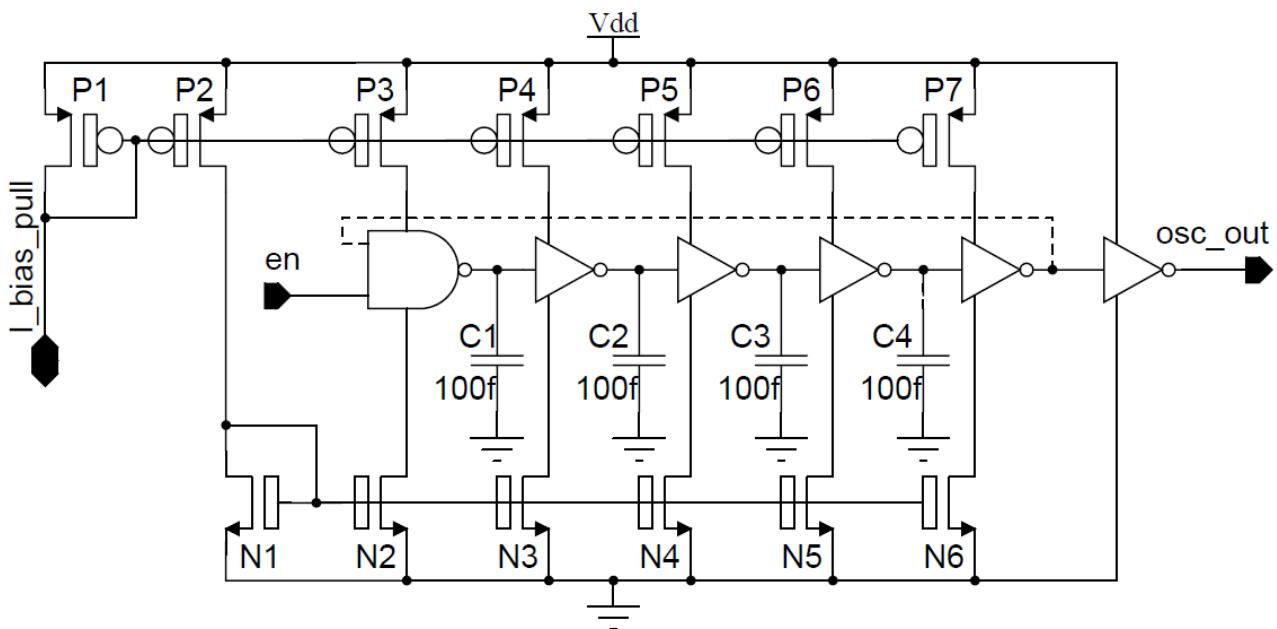


Figura 2.21: Oscilatorul in inel de tipul *current starving*.

Pentru proiectarea acestui oscilator vom refolosi inversorului proiectat in timpul primului laborator, in plus adaugând și capacitoare. Principala proprietate a acestui tip de oscilatoare este capabilitatea modificării frecvenței de oscilație cu ajutorul curentului de polarizare  $I_{BIAS}$ . Capacitorul pe care il vom folosi pentru proiectarea acestui oscilator este *nmoscap* și il gasiti in libraria *gpdk180* (vezi Figura 2.22). Valoarea capacității o putem modifica din parametrul *Capacitance* al acestuia. O alternativă este să modificam aria capacitorului (vezi Figura 2.23)

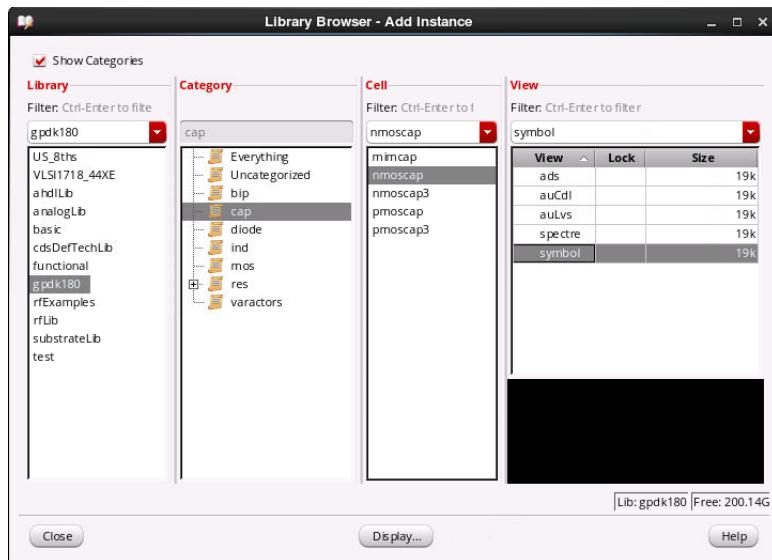


Figura 2.22: Selectia capacitorului din libraria gpdk180.

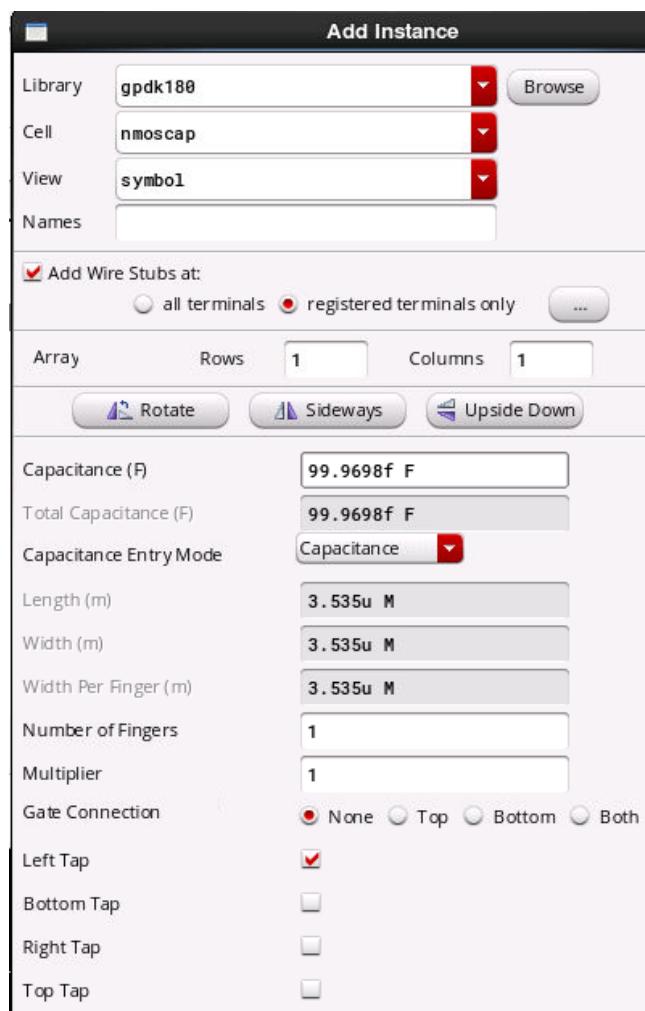


Figura 2.23: Dimensionarea capacitorului.

Odată finalizată schema, putem testa oscilatorul folosind o nouă schemă ca *testbench* (vezi Figure 2.24). Pentru a putea observa diferența dintre frecvențele de oscilație dateă de curenții de polarizare, vom simula 2 oscilatoare în același testbench în care oscilatoarele vor fi polarizate folosind curenti diferiti. Pentru polarizarea circuitului vom folosi surse de curent *DC* ideale din libraria *analogLib* cu valorile *ipull1* și *ipull2* (vezi Figure 2.25 și Figure 2.26).

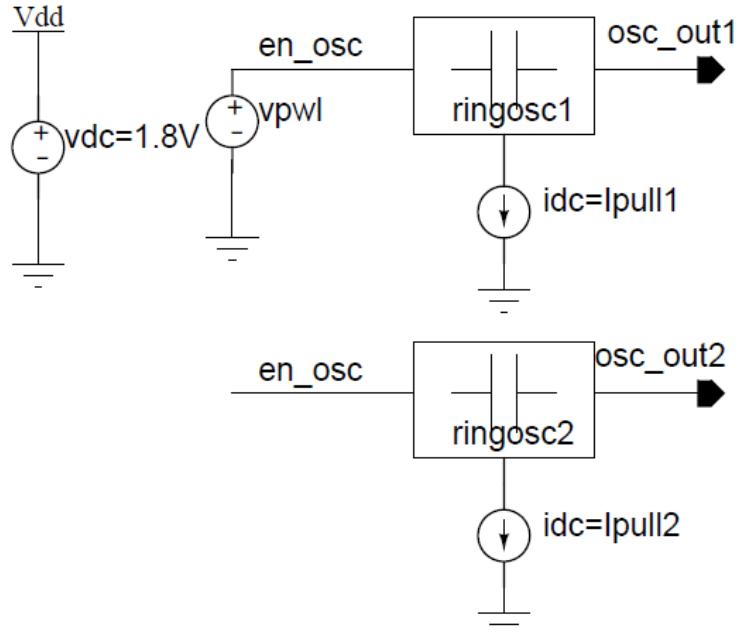


Figura 2.24: Dimensionarea capacitorului.



Figura 2.25: Selectia surse de curent DC din analogLib.

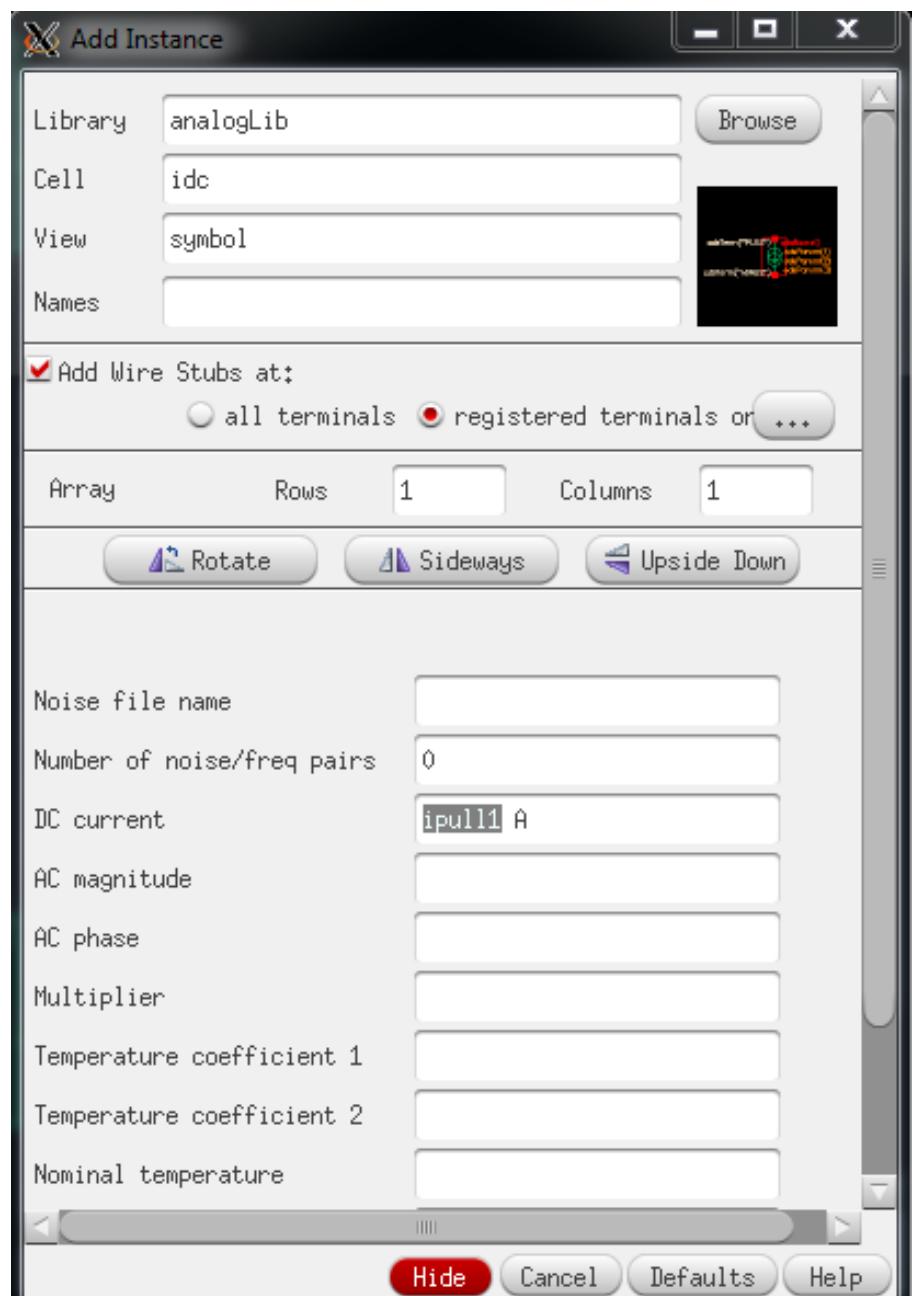


Figura 2.26: Proprietatile sursei de curent DC din analogLib.

Urmatorul pas este să copiem variabilele din schema în fereastra *Analog Design Environment* sub meniu *Variables* · *Copy From Cellview* (vezi Figura 2.27). Vom folosi pentru valoarea curentului  $ipull1$   $5\mu A$  și pentru curentul  $ipull1$   $10\mu A$  într-o simulare tranzitorie de durată  $5\mu s$  (vezi Figura 2.28).

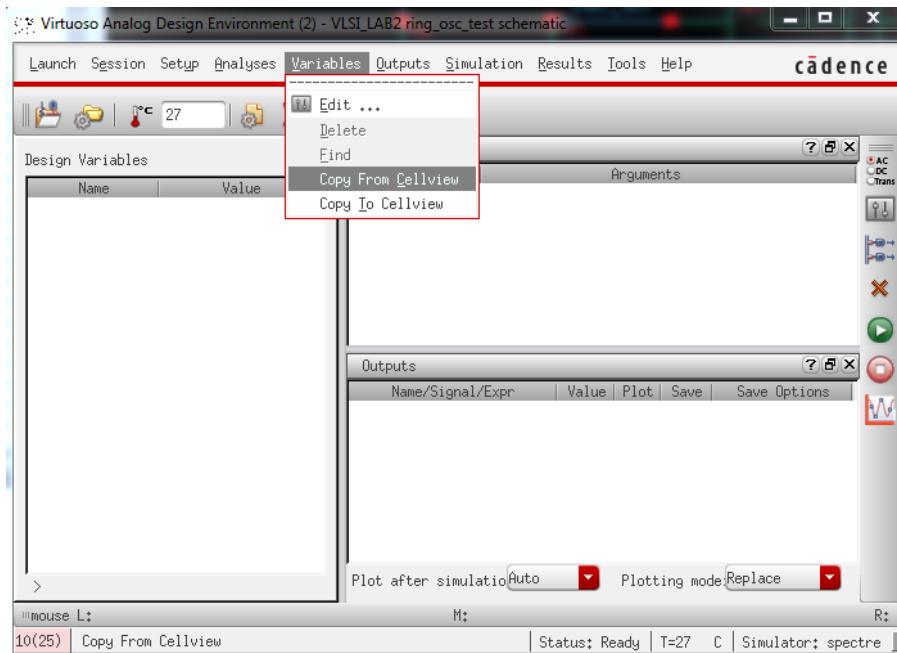


Figura 2.27: ADE-L: Copierea variabilelor din schema in meniu de simulare.

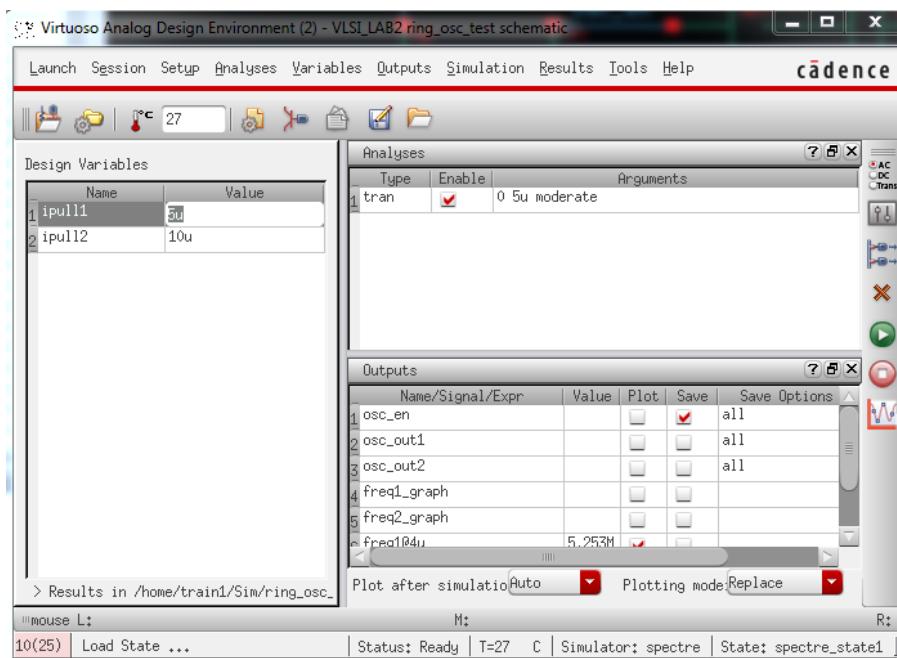


Figura 2.28: Parametrii ADE-L.

Calculul frecvenței se poate face măsurând-o din formele de undă a semnalului de ieșire sau cu ajutorul *Calculatorului* din meniu *Tools* · *Calculator* (vezi Figura 2.29). În cazul în care folosim *Calculatorul*, vom selecta semnalul dorit variabil în timp folosind butonul *vt*, selectându-l pe schema, după care butonul de la tastatura *Esc*. Odată apărut în meniu principal al *Calculatorului* ca în Figura 2.30 putem folosi funcția predefinită *freq* pentru a identifica variația frecvenței semnalului în timp (vezi Figura 2.31) având ca parametrii semnalul selectat initial, frontul crescător al semnalului și pragul de  $VDD/2$ , adică 0.9V (vezi Figura 2.32 și Figura 2.33). Exact ca la calculul  $t_{pd}$ , funcția se poate vizualiza în meniu principal al *Calculatorului* după apăsarea butonului *OK* (vezi Figura 2.34) după care se poate copia în meniu *ADE* folosind meniu *Outputs* · *Setup* și se pot genera 2 funcții *freq* pentru fiecare din cele 2 ieșiri ale oscilatoarelor (vezi Figura 2.35 și Figura 2.36). Pentru a identifica valoarea frecvenței la un anumit moment de timp, se va folosi funcția *value* și punctul la care să interpoleze valoarea, în cazul nostru  $4\mu s$  (vezi Figura 2.37).

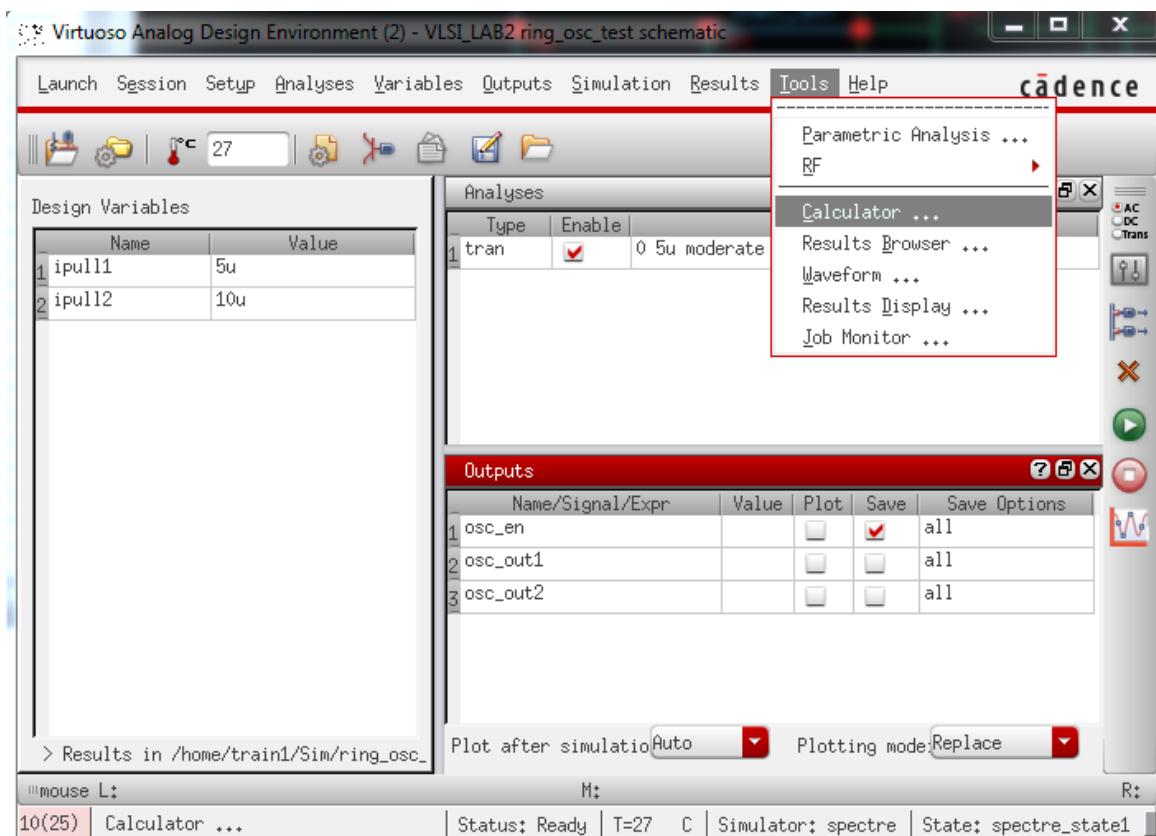


Figura 2.29: Pornirea calculatorului în ADE-L.

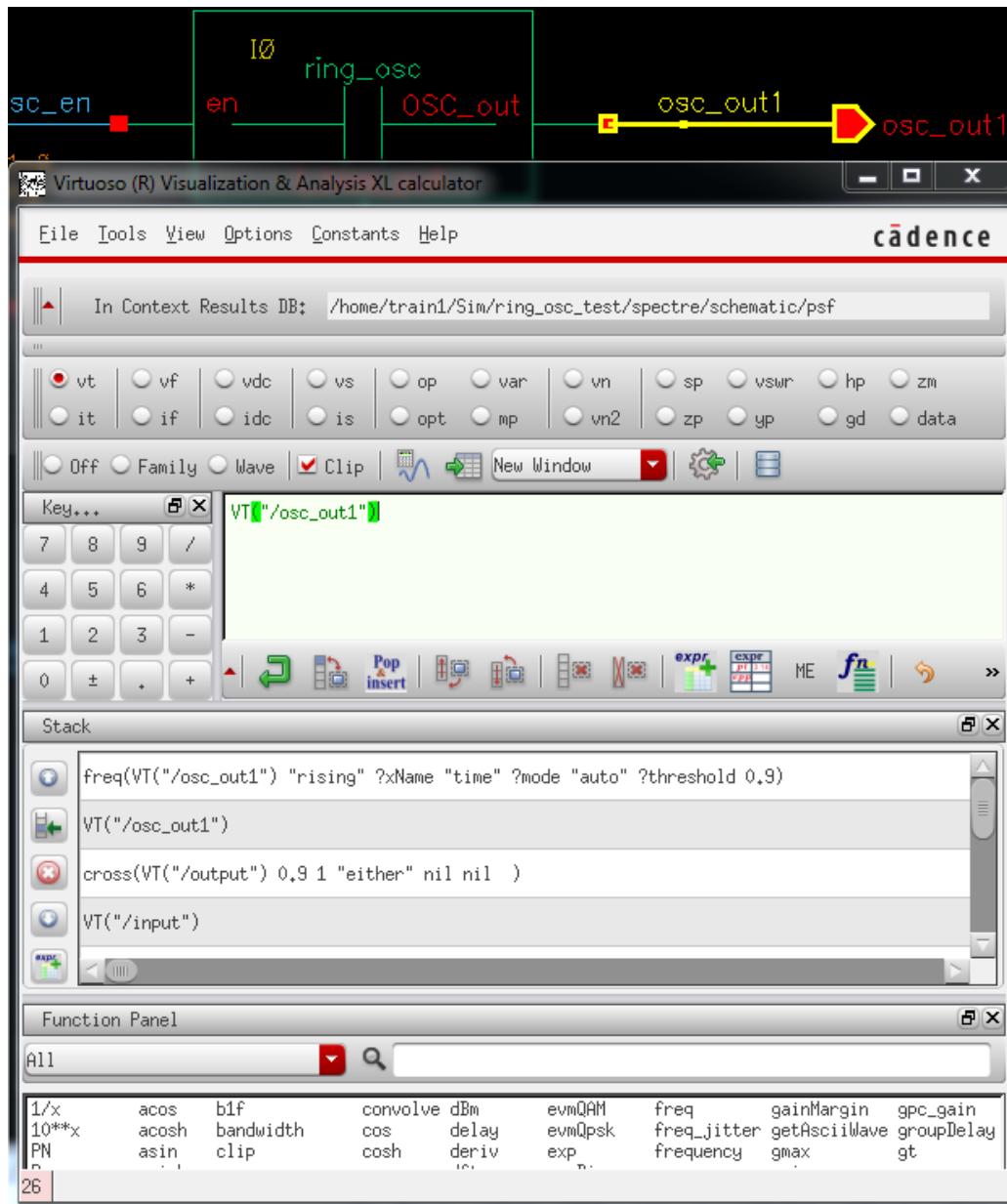


Figura 2.30: Calculatorul ADE-L.



Figura 2.31: Calculatorul ADE-L: functia *freq*.

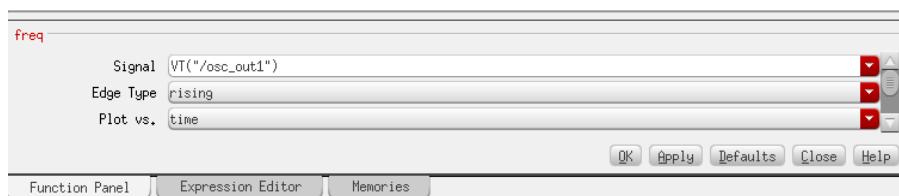


Figura 2.32: Calculatorul ADE-L: functia *freq*.



Figura 2.33: Calculatorul ADE-L: functia *freq*.

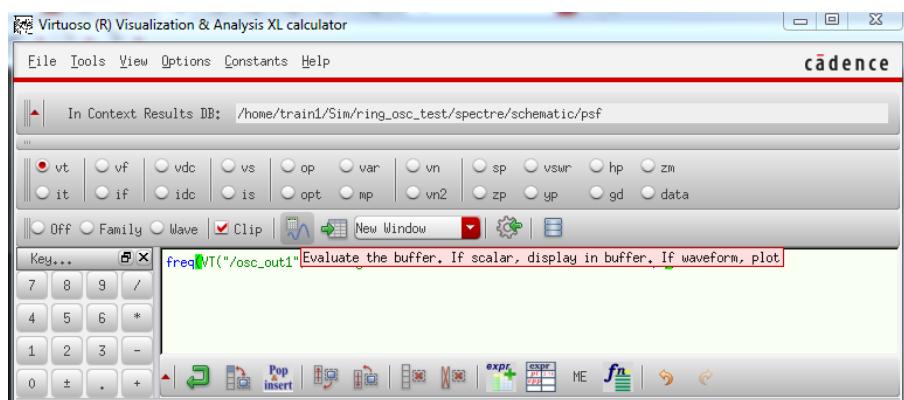


Figura 2.34: Calculatorul ADE-L: functia *freq*.

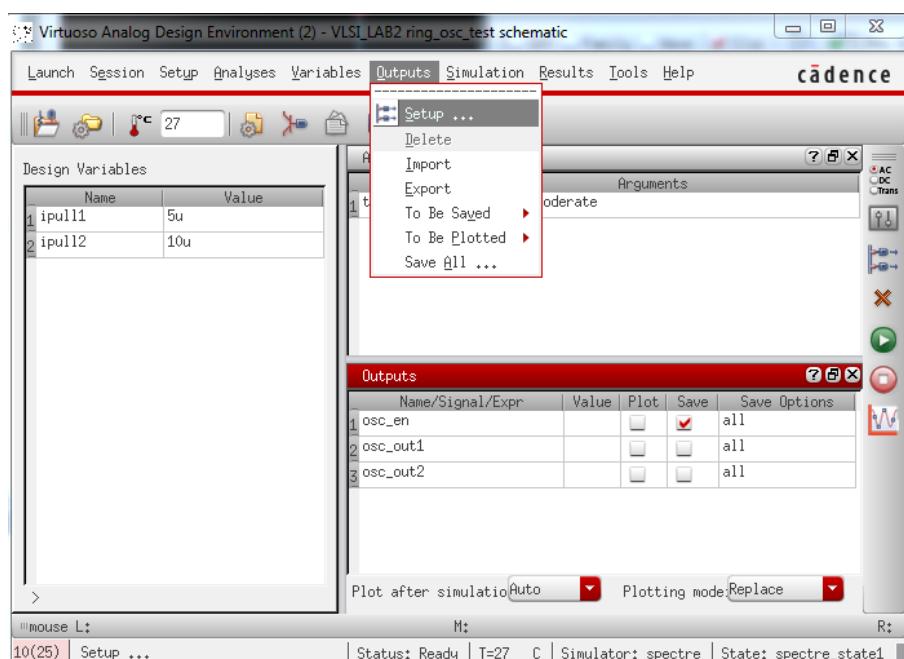


Figura 2.35: ADE-L: Calculul frecventei oscilatorului inel.

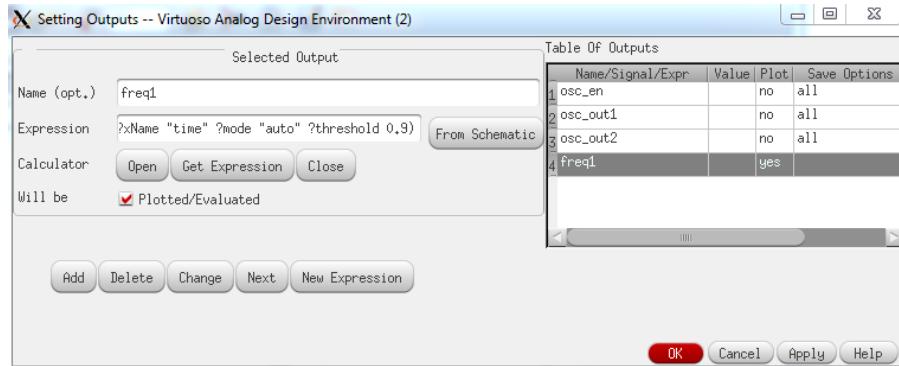


Figura 2.36: ADE-L: Calculul frecvenței oscillatorului inel.

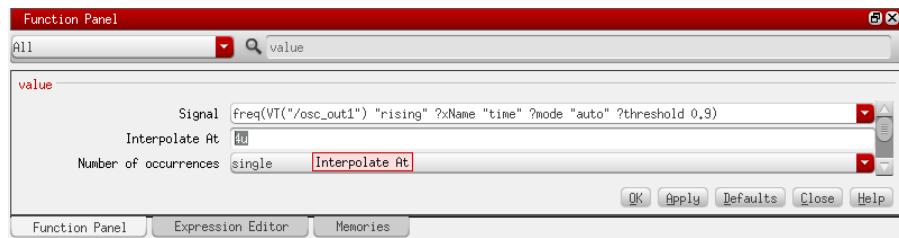


Figura 2.37: ADE-L: Calculul frecvenței oscillatorului inel.

## 2.5 Proiectarea și evaluarea unui divizor de ceas și a unui registru de deplasare

Proiectarea divizorului de ceas pleacă de la schema unui bistabil  $D$ (DFF) prezentată în Figura 2.38. Divizorul de ceas 8x se proiectează cascadând trei celule DFF folosind conexiunile ca în schema din Figură 2.39. Testbench-ul se regăsește în Figură 2.40. Tot cu ajutorul DFF se poate desena și un registru de deplasare (vezi Figură 2.41).

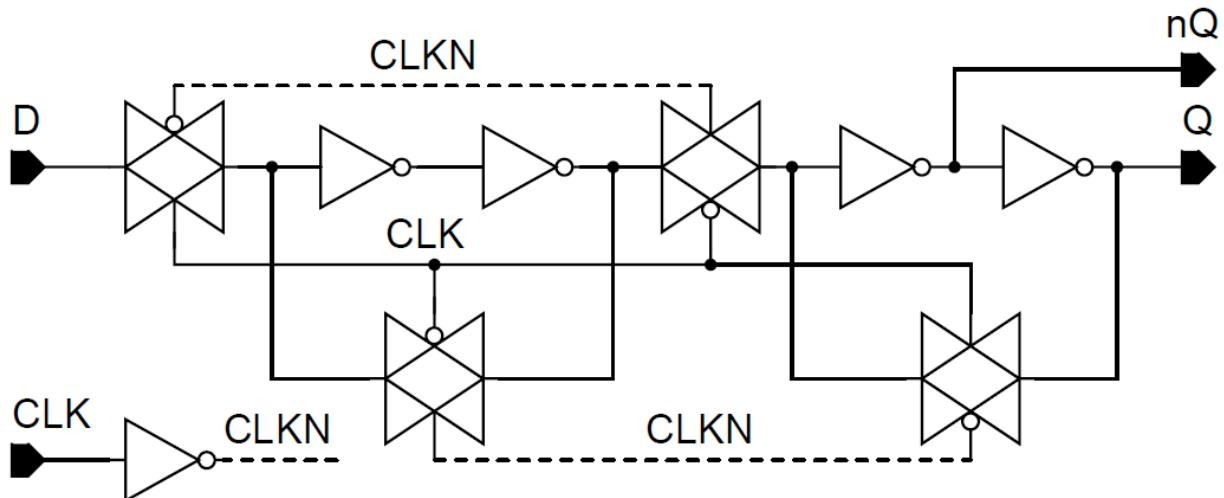


Figura 2.38: Schema unui bistabil  $D$ .

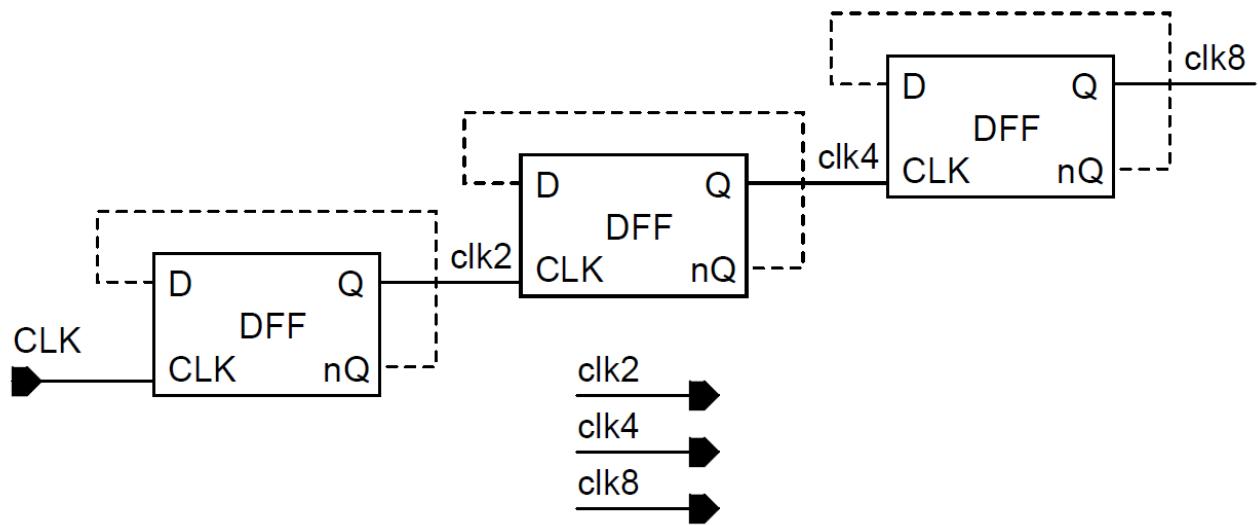


Figura 2.39: Schema unui 8x *Clock Devider*.

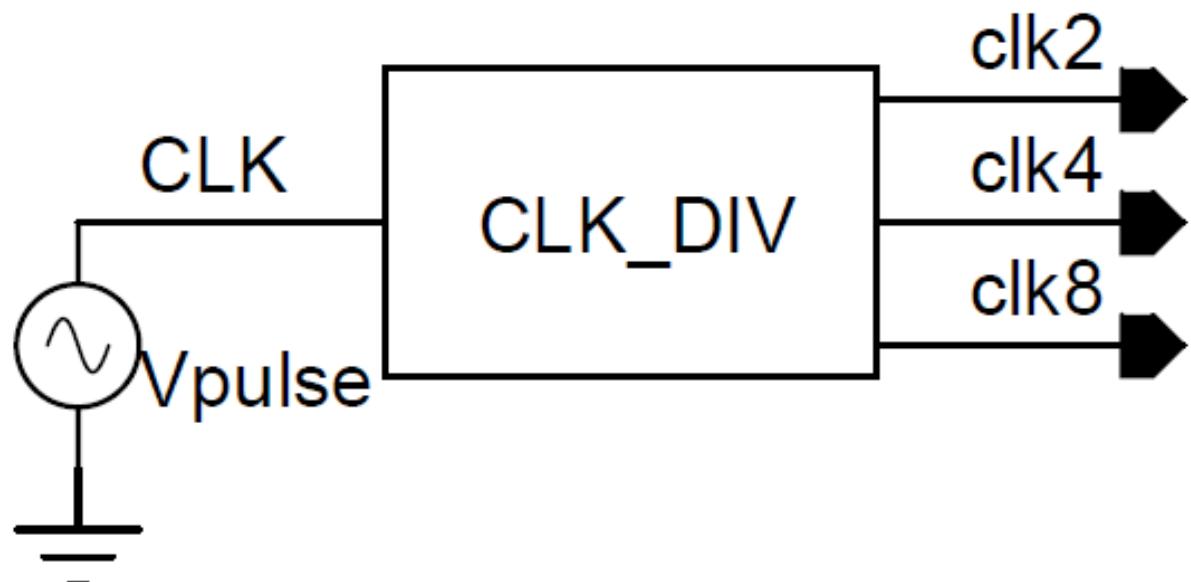


Figura 2.40: Testbench-ul unui 8x *Clock Devider*.

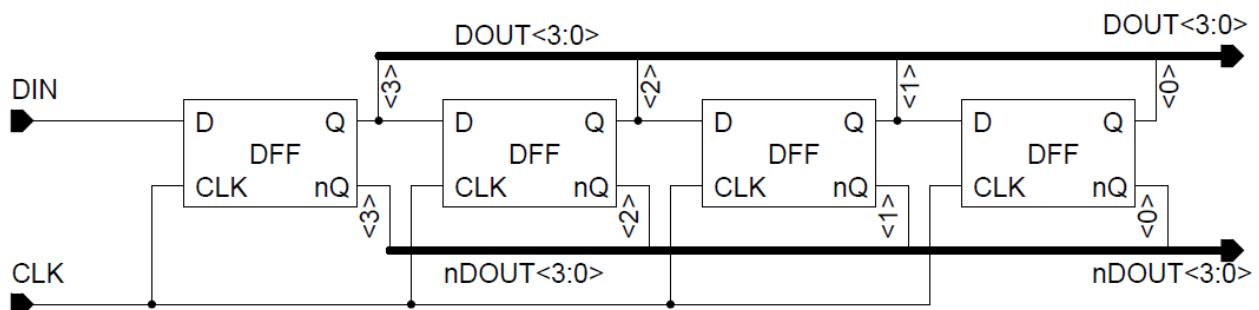


Figura 2.41: Schema unui *Shift Register*.

# Capitolul 3

## Laboratorul 3

Scopul acestui laborator este:

1. Să înțelegem funcționarea celulei de memorie Static Random-Access Memory (*SRAM*) și să implementăm un exemplu cu numai 6 tranzistoare (6T),
2. Să proiectăm decodeoare  $n : 2n$  pentru accesarea liniilor si coloanelor unei matrici de memorie, și
3. Să proiectăm o matrice SRAM de dimensiune  $4 \times 4$  avand la bază celula 6T SRAM.

### 3.1 Proiectarea celulei 6T *SRAM*

Celula 6T *SRAM* prezentată în Figura 3.1 este o celulă de memorie *volatile* (informația stocată de aceasta se pierde dacă nu este alimentată) care stochează informația asemănător unui *latch*. *SRAM*-ul se deosebește de memoria Dynamic Random-Access Memory (*DRAM*) prin viteza superioară și prin renunțarea la *refresh*-ul periodic. Orice imbunătățire are și un cost, acesta fiind aria marită datorita celor 6T în comparație cu celula *DRAM* care este formata doar dintr-un tranzistor și un condensator.

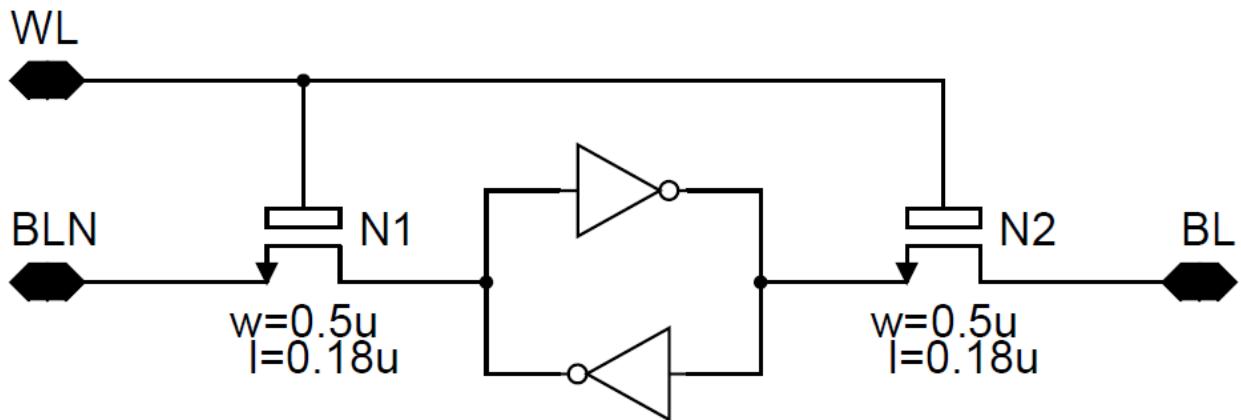


Figura 3.1: Schema unei celule de memorie 6T *SRAM*.

Cu ajutorul inversoarelor proiectate în Laboratorul 1 vom desena celula de memorie 6T *SRAM* exact ca în Figura 3.1. Inversoarele folosite la proiectarea acestei celule se conectează în configurație "cross"-coupled" (iesirea uneia la intrarea celuilalt). Aceasta celula de memorie poate stoca doar 2 valori: 0 logic (*gnd!*) și 1 logic (*vdd!*). Pentru accesarea valorii memorate în timpul citirii sau scrierii se folosesc tranzistorii *N1* și *N2* pe care de acum înainte ii vom numi *access transistors*.

Dupa finalizarea schemei se va genera un simbol care va fi asociat acestei celule. Deoarece vom folosi aceste celule de memorie într-o matrice, vom genera un simbol asemănător cu cel din Figura 3.2. Dupa cum se observă în figură, porturile *bitline - n* (*BLN*) și *bitline* (*BL*) sunt

accesibile atât din partea superioară cât și cea inferioară a simbolului, iar port-ul *wordline* este accesibil atât în dreapta și și în stanga simbolului.

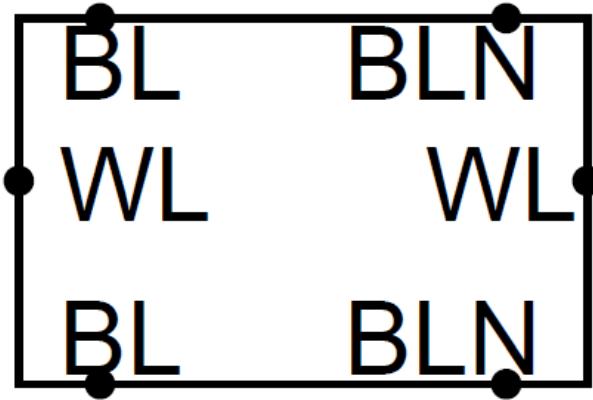


Figura 3.2: Simbolul unei celule de memorie 6T SRAM.

**Notă:** În cazul în care inversoarele desenate pe durata Laboratorul 1 au tensiuni dedicate de alimentare, la acest nivel ierarhic se vor folosi porturile globale *Gnd* și *Vdd*.

*Testbench*-ul celulei de memorie 6T SRAM se regăseste în Figura 3.3. Acesta conține, alături de stimuli, celula de memorie a carui simbol l-am generat mai la pasul anterior, simbolul inversorului și al unui *tristate buffer*, ambele generate în timpul primului laborator.

Ca stimuli de intrare se va folosi pentru *DATAin* sursa *vpwl* cu numele *V2* descrisă în Figura 3.4, pentru *WRITE – READn* sursa *vpwl* cu numele *V3* descrisă în Figura 3.5, iar pentru *WL* sursa *vpwl* cu numele *V1* descrisă în Figura 3.6.

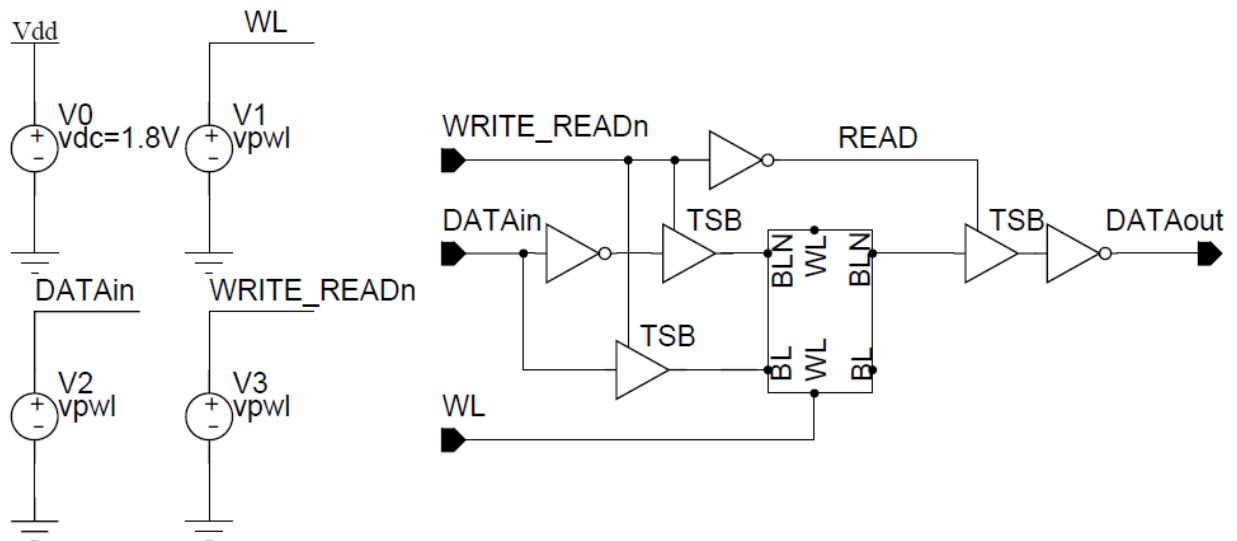


Figura 3.3: Testbench-ul celulei de memorie 6T SRAM.

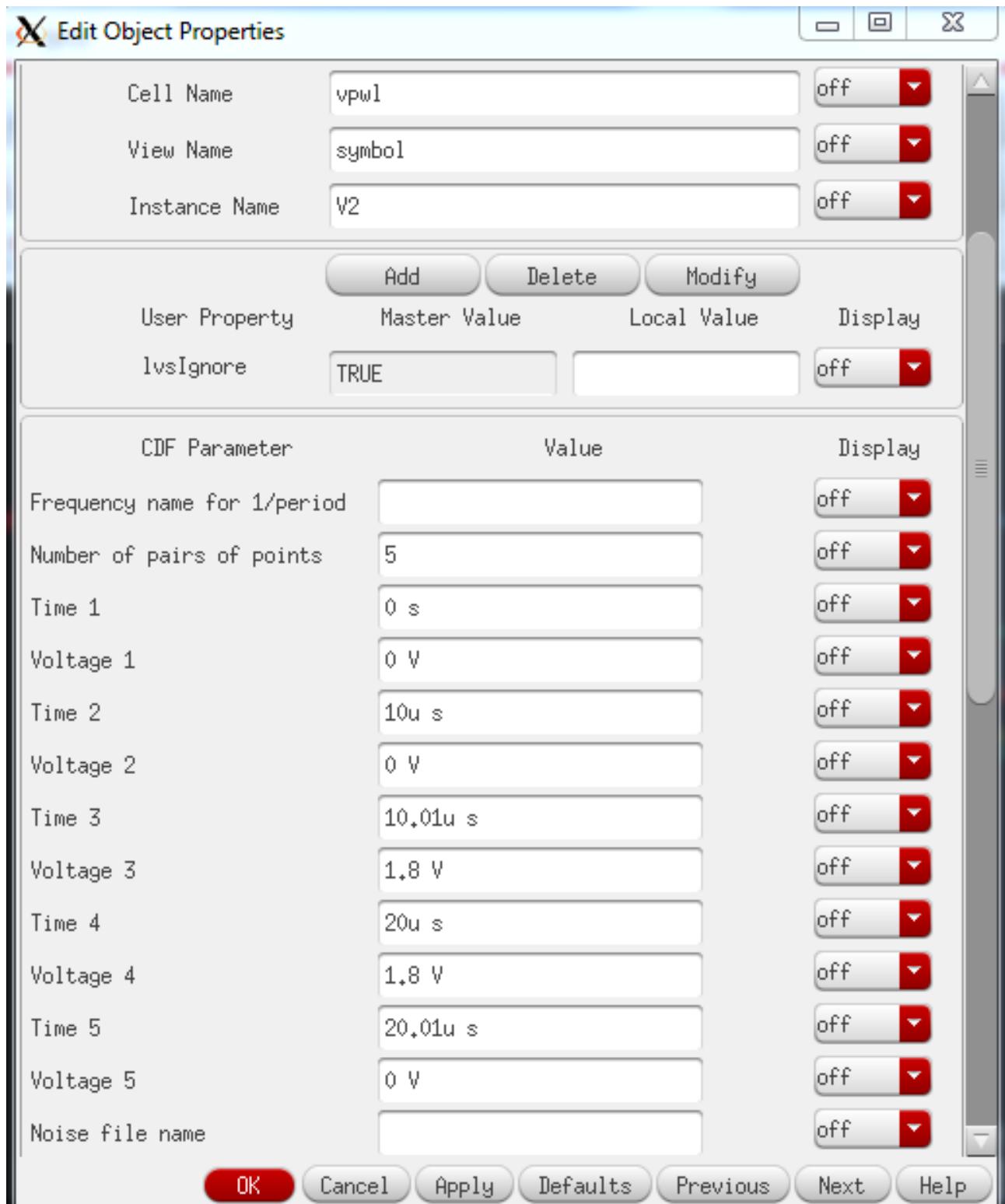


Figura 3.4: Proprietatile sursei vpwl V2.

**Edit Object Properties**

Voltage 1	0 V	off ▾
Time 2	4u s	off ▾
Voltage 2	0 V	off ▾
Time 3	4.01u s	off ▾
Voltage 3	1.8 V	off ▾
Time 4	6u s	off ▾
Voltage 4	1.8 V	off ▾
Time 5	6.01u s	off ▾
Voltage 5	0 V	off ▾
Time 6	14u s	off ▾
Voltage 6	0 V	off ▾
Time 7	14.01u s	off ▾
Voltage 7	1.8 V	off ▾
Time 8	16u s	off ▾
Voltage 8	1.8 V	off ▾
Time 9	16.01u s	off ▾
Voltage 9	0 V	off ▾
Time 10	24u s	off ▾
Voltage 10	0 V	off ▾
Time 11	24.01u s	off ▾
Voltage 11	1.8 V	off ▾
Time 12	26u s	off ▾
Voltage 12	1.8 V	off ▾
Time 13	26.01u s	off ▾
Voltage 13	0 V	off ▾
Noise file name		off ▾

OK Cancel Apply Defaults Previous Next Help

Figura 3.5: Proprietatile sursei vpwl V3.

Number of pairs of points	23	off ▾
Time 1	0 s	off ▾
Voltage 1	0 V	off ▾
Time 2	4.2u s	off ▾
Voltage 2	0 V	off ▾
Time 3	4.21u s	off ▾
Voltage 3	1.8 V	off ▾
Time 4	5.8u s	off ▾
Voltage 4	1.8 V	off ▾
Time 5	5.81u s	off ▾
Voltage 5	0 V	off ▾
Time 6	8u s	off ▾
Voltage 6	0 V	off ▾
Time 7	8.01u s	off ▾
Voltage 7	1.8 V	off ▾
Time 8	12u s	off ▾
Voltage 8	1.8 V	off ▾
Time 9	12.01u s	off ▾
Voltage 9	0 V	off ▾
Time 10	14.2u s	off ▾
Voltage 10	0 V	off ▾
Time 11	14.21u s	off ▾
Voltage 11	1.8 V	off ▾
Time 12	15.8u s	off ▾
Voltage 12	1.8 V	off ▾
Time 13	15.81u s	off ▾
Voltage 13	0 V	off ▾
Time 14	18u s	off ▾
Voltage 14	0 V	off ▾
Time 15	18.01u s	off ▾
Voltage 15	1.8 V	off ▾

Figura 3.6: Proprietatile sursei vpwl V1.

### 3.2 Proiectarea decodoarelor $n : 2^n$ pentru wordline și pentru bitline

Decodoarele  $n : 2^n$  sunt formate din porți AND având  $n$  intrări, una pentru fiecare rand din memorie. De obicei aceste porți AND sunt realizate cu ajutorul portilor  $NAND/NOR$  și al inversoarelor. Necesitatea lor este evidentă: decodarea pozitiei randului din array-ul de celule. Reducerea masiva a ariei destinate acestei interconectari. Un exemplu al unui decodor  $2 : 2^2$  se regaseste în Figura 3.7. Practic cu 2 biti de intrare putem decoda 4 wordline-uri. După finalizarea schemei, se va genera un simbol care va fi asociat acestei celule asemanator cu celu din Figura 3.2.

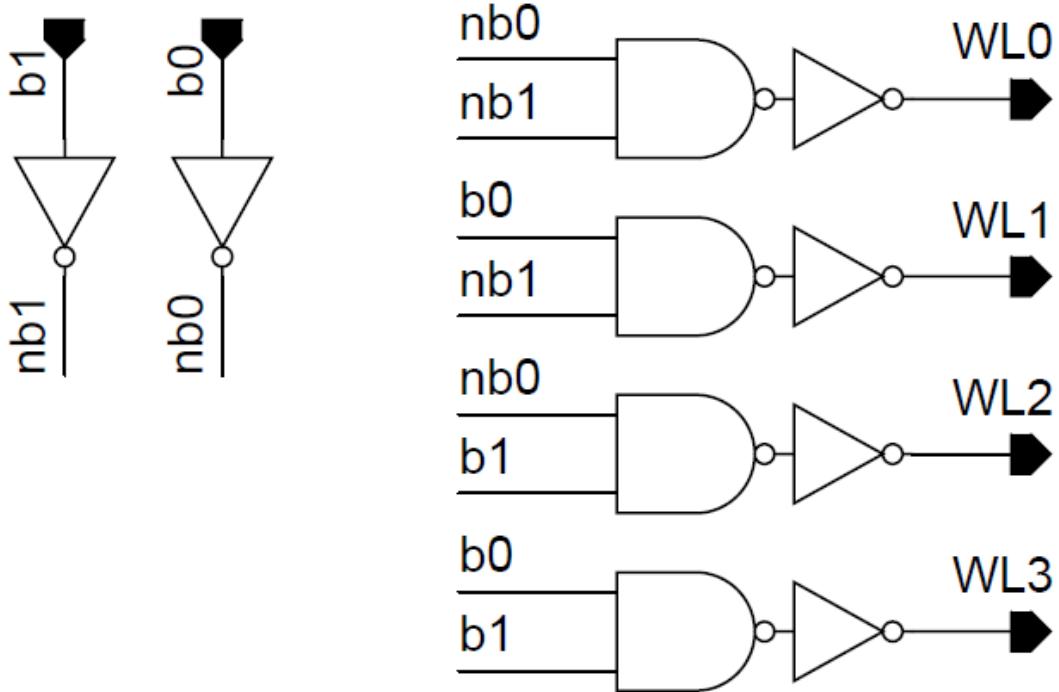


Figura 3.7: Schema decodorului  $2 : 2^2$ .

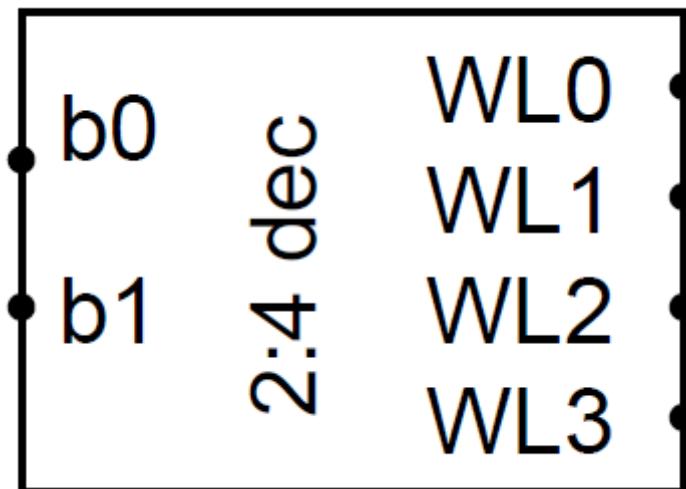


Figura 3.8: Simbolul decodorului  $2 : 2^2$ .

### 3.3 Scrierea si citirea datelor intr-o celula 6T SRAM

Deoarece la o celula *6T SRAM* scrierea se face differential, pentru *bitline*-uri este necesară existența datelor la intrarea negată, operație ce se realizează cu circuitul din Figura 3.9.

De asemenea, repetorul *tristate* (*TSB*) realizează izolarea *bitline*-urilor pe durata operației de citire.

Dupa finalizarea schemei se va genera un simbol care va fi asociat celulei proiectate. Deoarece aceste simboluri vor fi folosite ca și cap-de-rând pentru matricea celulelor de memorie, vom proiecta un simbolul exact ca cel din Figura 3.10. Dupa cum se observă în această figură, porturile *WRITE* sunt accesibile atât în partea dreaptă cât și în partea stângă a simbolului.

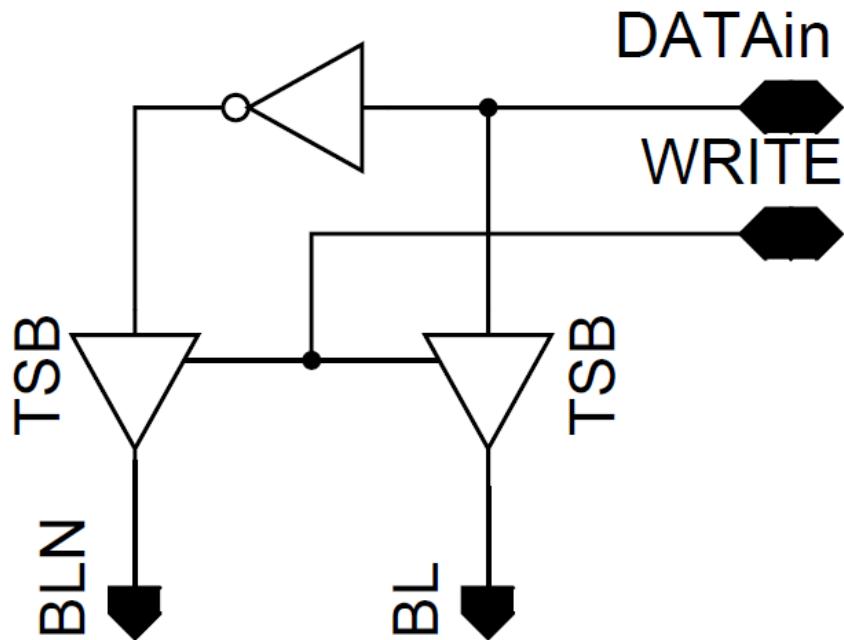


Figura 3.9: Semnalul util *DATAin* și conditionarea lui de semnalul *WRITE*.

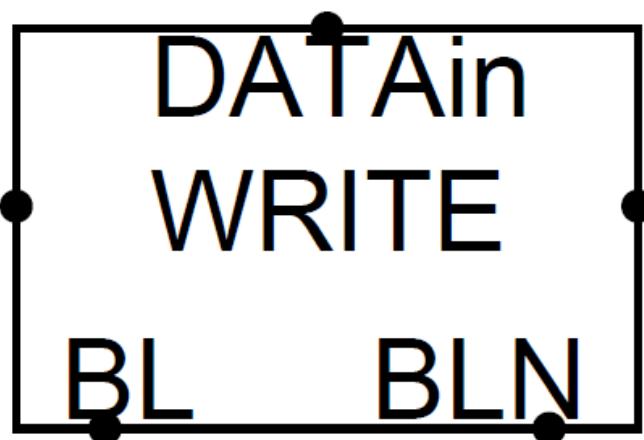


Figura 3.10: Simbolul blocului *DATAin*.

Pentru citirea informației de la fiecare *bitline*, vom folosi schema din Figura 3.11. Citirea celulei 6T SRAM se face folosind semnalul de control *READ* având funcția *enable* pentru *TSB*-ul figură.

După finalizarea schemei se va genera un simbol care va fi asociat acestei celule. Deoarece aceste celule vor fi folosite ca și cap-de-rand pentru matricea celulelor de memorie, vom genera un simbol asemanător cu cel din Figura 3.12. După cum se observă în această figură, portul *READ* este accesibil atât în partea dreaptă cât și în partea stângă a simbolului.

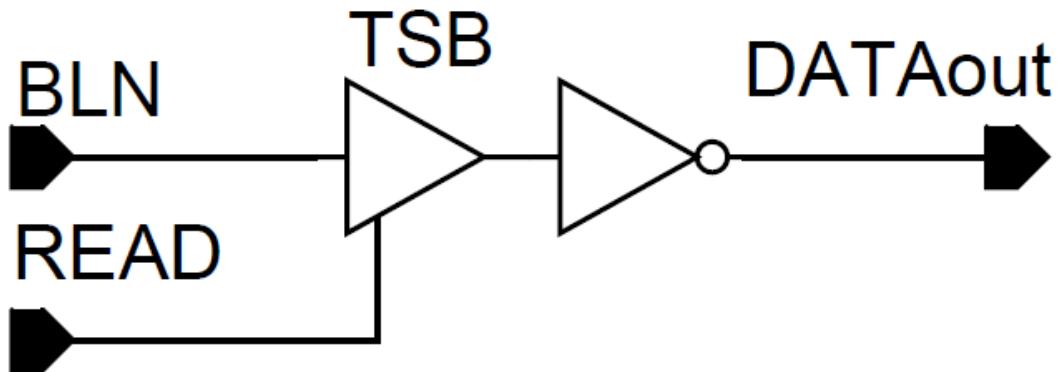


Figura 3.11: Citirea informatiei de pe un *bitline* conditionat de semnalul *READ*.

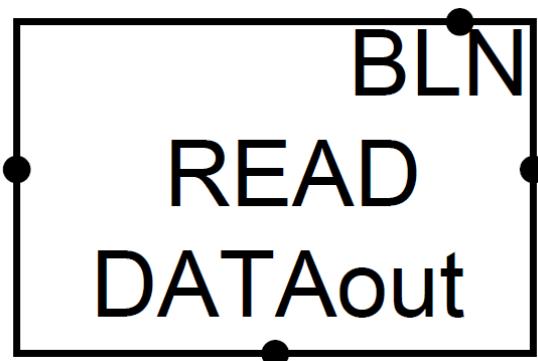


Figura 3.12: Simbolul blocului *DATaout*.

### 3.4 Studiu de caz: Scrierea si citirea datelor intr-o matrice SRAM 4x4

Pentru a intinge mai bine operația de scriere și de citire a unui celule SRAM, vom studia funcționarea unei matrici  $4 \times 4$  de celule 6T SRAM. *Array*-ul dispune de 4 randuri, fiecare rand reprezintă un cuvant de 4 biti. Figura 3.13 prezintă *testbench*-ul unei astfel de structuri. Pentru acest studiu de caz s-au folosit 16 instante *SRAM 6T*, 4 instante *DATAin*, 4 instante *DATaout*, o instantă  $2 : 2^2$  *row decoder*, 2 inversoare împreună cu stimulii de intrare și de control. Pentru semnalele *DATAIN*  $< 3 : 0 >$  și *DATAOUT*  $< 3 : 0 >$  se va folosi o conexiune de tip *bus*. Pentru aceasta conexiune în fereastra *Schematic Editor* accesăm meniul *Create -> Wire (wide)* sau apasam direct butonul din Figura 3.14.

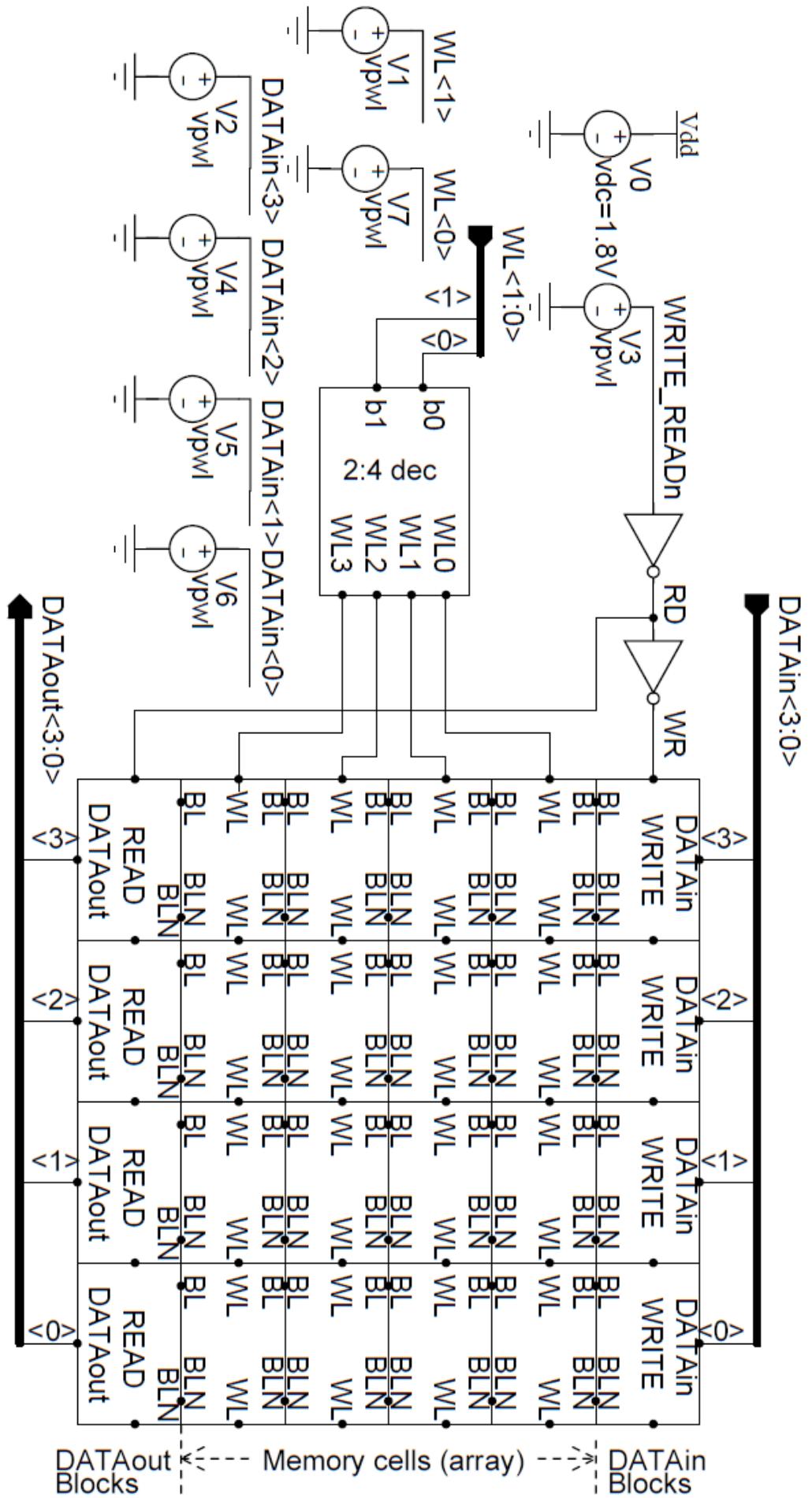


Figura 3.13: Testbench-ul studiului de caz: matrice SRAM 4x4.

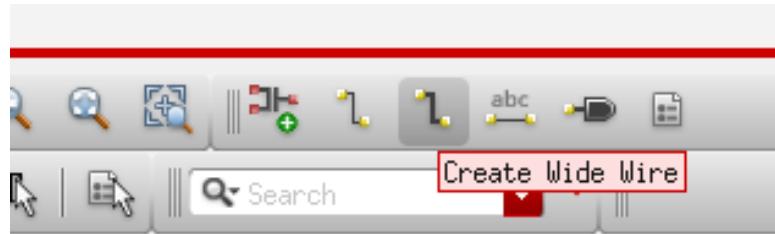


Figura 3.14: Butonul *wide wire*.

Următorul pas este generăm pinii de intrare si de ieșire  $DATAIN < 3 : 0 >$  si  $DATAOUT < 3 : 0 >$ . (vezi Figura 3.15 si Figura 3.16).

**Important:** pentru a conecta  $DATAin < 3 >$  la prima coloana,  $DATAin < 2 >$  la a doua coloana, și aşa mai departe, trebuie sa agăugăm o etichetă (*label*) pe conexiunile desenate cu *Wire (Narrow)* astfel:  $< 3 >$  pentru  $DATAin < 3 >$ ,  $< 2 >$  pentru  $DATAin < 2 >$ , și aşa mai departe. Pentru vectorul  $DATAout$  se aplică același algoritm.

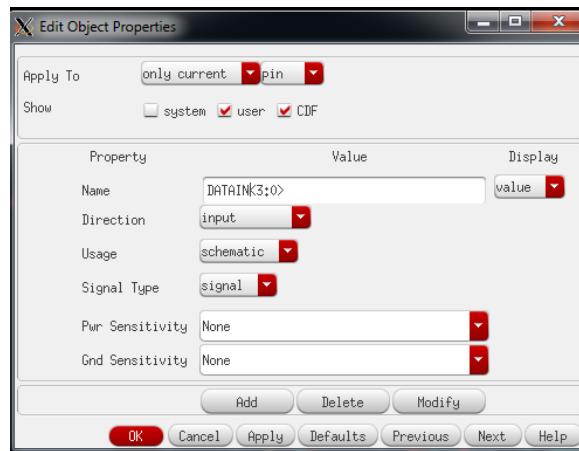


Figura 3.15: Generarea portului de intrare pe 4 biti  $DATAIN < 3 : 0 >$ .



Figura 3.16: Generarea portului de ieșire pe 4 biti  $DATAOUT < 3 : 0 >$ .

In final, stimulii pentru  $WL < 0 >$  se regăsesc în Figura 3.17, stimulii pentru  $WL < 1 >$  se regăsesc în Figura 3.18, pentru  $DATAIN < 3 >$  se regăsesc în Figura 3.19, stimulii pentru  $DATAIN < 2 >$  se regăsesc în Figura 3.20, pentru  $DATAIN < 1 >$  se regăsesc în Figura 3.21, stimulii pentru  $DATAIN < 0 >$  se regăsesc în Figura 3.22. stimulii pentru  $WRITE - READ$  se regăsesc în Figura 3.23.

Formele de undă pentru semnalele de mai sus se regăsesc în Figure 3.24.

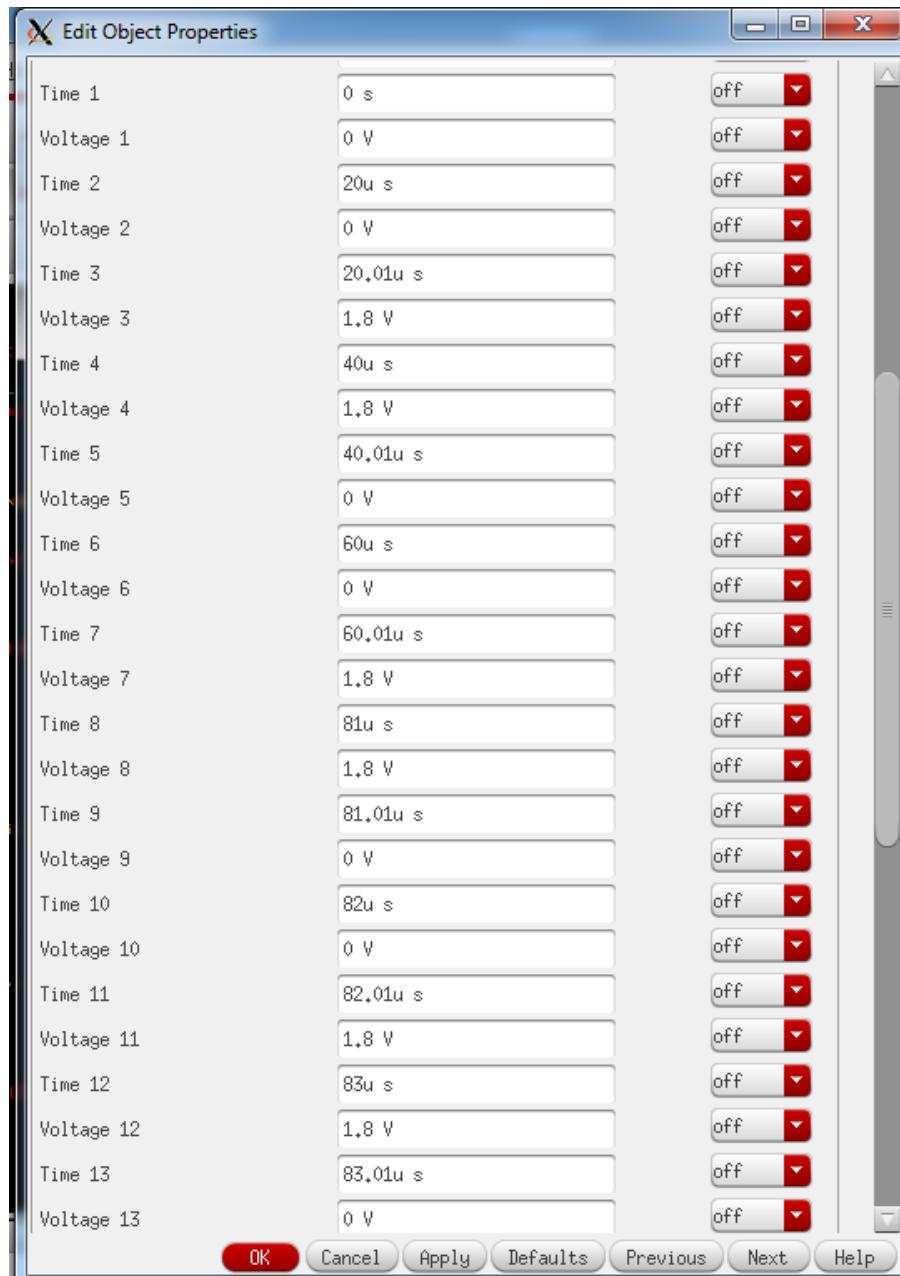


Figura 3.17: Proprietatile sursei vpwl V7 pentru semnalul  $WL < 0 >$ .

Number of pairs of points	5	off ▾
Time 1	0 s	off ▾
Voltage 1	0 V	off ▾
Time 2	40u s	off ▾
Voltage 2	0 V	off ▾
Time 3	40.01u s	off ▾
Voltage 3	1.8 V	off ▾
Time 4	82u s	off ▾
Voltage 4	1.8 V	off ▾
Time 5	82.01u s	off ▾
Voltage 5	0 V	off ▾

Figura 3.18: Proprietatile sursei vpwl V1 pentru semnalul  $WL < 1 >$ .

Number of pairs of points	7	off ▾
Time 1	0 s	off ▾
Voltage 1	0 V	off ▾
Time 2	30u s	off ▾
Voltage 2	0 V	off ▾
Time 3	30.01u s	off ▾
Voltage 3	1.8 V	off ▾
Time 4	50u s	off ▾
Voltage 4	1.8 V	off ▾
Time 5	50.01u s	off ▾
Voltage 5	0 V	off ▾
Time 6	70u s	off ▾
Voltage 6	0 V	off ▾
Time 7	70.01u s	off ▾
Voltage 7	1.8 V	off ▾

Figura 3.19: Proprietatile sursei vpwl V2 pentru semnalul  $DATAIN < 3 >$ .

Number of pairs of points	3	off ▾
Time 1	0 s	off ▾
Voltage 1	0 V	off ▾
Time 2	50u s	off ▾
Voltage 2	0 V	off ▾
Time 3	50.01u s	off ▾
Voltage 3	1.8 V	off ▾

Figura 3.20: Proprietatile sursei vpwl V4 pentru semnalul *DATAIN < 2 >*.

Number of pairs of points	7	off ▾
Time 1	0 s	off ▾
Voltage 1	0 V	off ▾
Time 2	30u s	off ▾
Voltage 2	0 V	off ▾
Time 3	30.01u s	off ▾
Voltage 3	1.8 V	off ▾
Time 4	50u s	off ▾
Voltage 4	1.8 V	off ▾
Time 5	50.01u s	off ▾
Voltage 5	0 V	off ▾
Time 6	70u s	off ▾
Voltage 6	0 V	off ▾
Time 7	70.01u s	off ▾
Voltage 7	1.8 V	off ▾

Figura 3.21: Proprietatile sursei vpwl V5 pentru semnalul *DATAIN < 1 >*.

Number of pairs of points	3	off ▾
Time 1	0 s	off ▾
Voltage 1	0 V	off ▾
Time 2	50u s	off ▾
Voltage 2	0 V	off ▾
Time 3	50.01u s	off ▾
Voltage 3	1.8 V	off ▾

Figura 3.22: Proprietatile sursei vpwl V6 pentru semnalul  $DATAIN < 0 >$ .

Number of pairs of points	5	off ▾
Time 1	0 s	off ▾
Voltage 1	0 V	off ▾
Time 2	10u s	off ▾
Voltage 2	0 V	off ▾
Time 3	10.01u s	off ▾
Voltage 3	1.8 V	off ▾
Time 4	80u s	off ▾
Voltage 4	1.8 V	off ▾
Time 5	80.01u s	off ▾
Voltage 5	0 V	off ▾

Figura 3.23: Proprietatile sursei vpwl V3 pentru semnalul  $WRITE - READn$ .

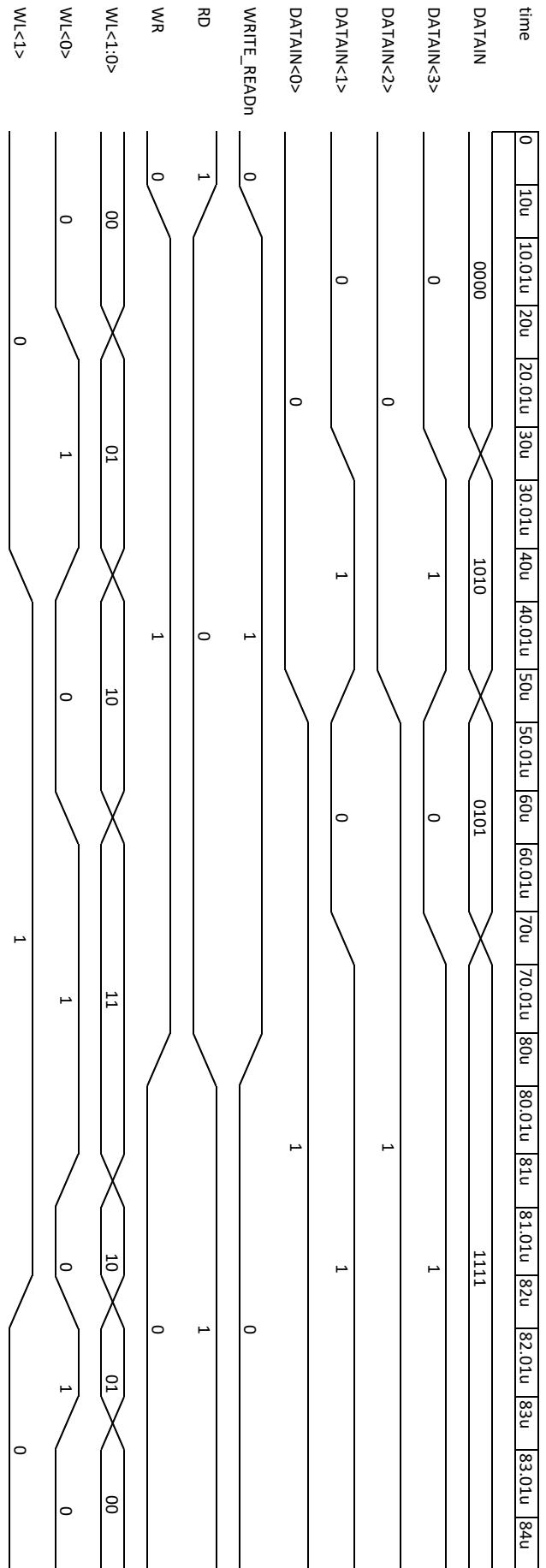


Figura 3.24: Formele de unde pentru  $DATAIN < 3 : 0 >$ ,  $WL < 1 : 0 >$  si  $WRITE - READn$ .

# Anexă A

## Anexă Descrierea Programului Cadence Virtuoso 6.1

### A.1 Introducere

Cadence Virtuoso este un program specializat pentru simularea funcționării circuitelor integrate electronice, digitale și analogice. Elementele uzuale de circuit de tipul dispozitivelor pasive (rezistențe, condensatoare, surse de curent și de tensiune) sau active (diode, tranzistoare bipolare și MOS, amplificatoare operaționale) sunt disponibile în biblioteca de componente, având asociați un număr de parametri definiti într-un model.

Va fi descrisa, în continuare, realizarea unui circuit integrat electronic, componentele utilizate, și parametrii acestora, analizele disponibile și modul de vizualizare a rezultatelor grafice.

### A.2 Pornirea programului Cadence Virtuoso

Primul pas este să vă introduceți credențialele în contul de *LINUX*.

- user: student
- pass: student

Următorul pas este să deschideți o fereastră terminal.

**Observatie:** Nu porniți niciodata programul *Cadence Virtuoso* în directorul vostru radacină.

Este indicat să creați un director de lucru, e.g., grupa și seria - 44xE cu x de la 1 la 4.

```
1 >> mkdir 44xE
2 >> cd 44xE
```

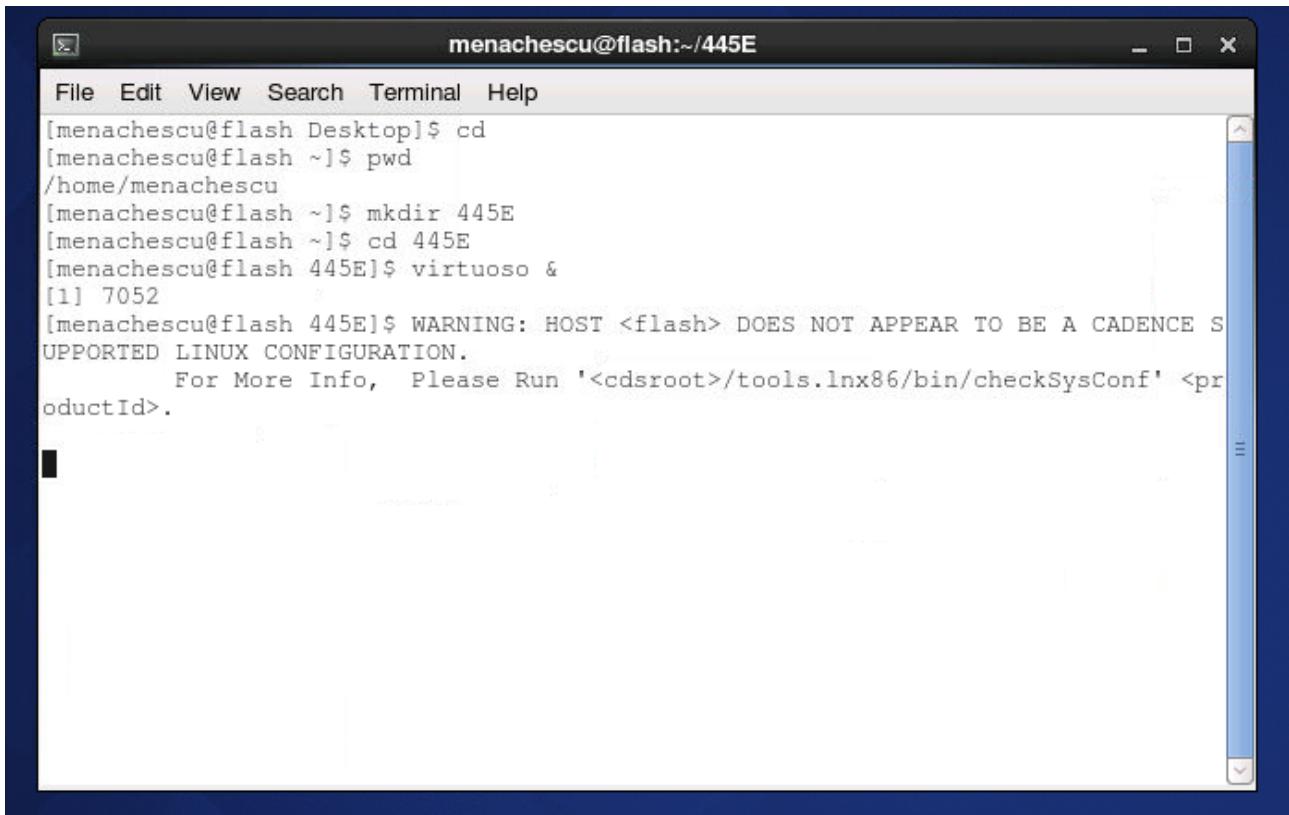
Puteți viziona lista de comenzi prezentate anterior în Figure A.3

Libraria de componente se adaugă înaintea primei porniri a programului prin crearea unui fișier în directorul de lucru cu numele *cds.lib*.

```
1 SOFTINCLUDE
2 /opt/cadence/install/IC617/share/cdssetup/cds.lib
3 DEFINE gpdk180
4 /opt/cadence/designkits/kits/gpdk180_v3.3/libs.oa22/gpdk180
5 DEFINE substrateLib
6 /opt/cadence/designkits/kits/gpdk180_v3.3/libs.oa22/substrateLib
```

Puteți viziona lista de comenzi prezentate anterior în Figure A.2

Dupa cum ati observat deja, sistemul de operare în care va rula programul Cadence este *Linux*. În Figure A.3 gasiți o lista de comenzi uzuale folosite în *Linux* și explicarea acestora. Puteți folosi comanda *man mkdir* în cazul în care vreti să obțineți mai multe informații despre comanda *mkdir*. Pentru începători este indicat să aveți tiparită aceasta listă de comenzi atunci când lucrați în *Linux*.



The screenshot shows a terminal window titled "menachescu@flash:~/445E". The window contains the following terminal session:

```
[menachescu@flash Desktop]$ cd  
[menachescu@flash ~]$ pwd  
/home/menachescu  
[menachescu@flash ~]$ mkdir 445E  
[menachescu@flash ~]$ cd 445E  
[menachescu@flash 445E]$ virtuoso &  
[1] 7052  
[menachescu@flash 445E]$ WARNING: HOST <flash> DOES NOT APPEAR TO BE A CADENCE S  
UPPORTED LINUX CONFIGURATION.  
For More Info, Please Run '<cdsroot>/tools.lnx86/bin/checkSysConf' <pr  
oductId>.
```

Figura A.1: Comenzile din terminal pentru pornirea Cadence Virtuoso

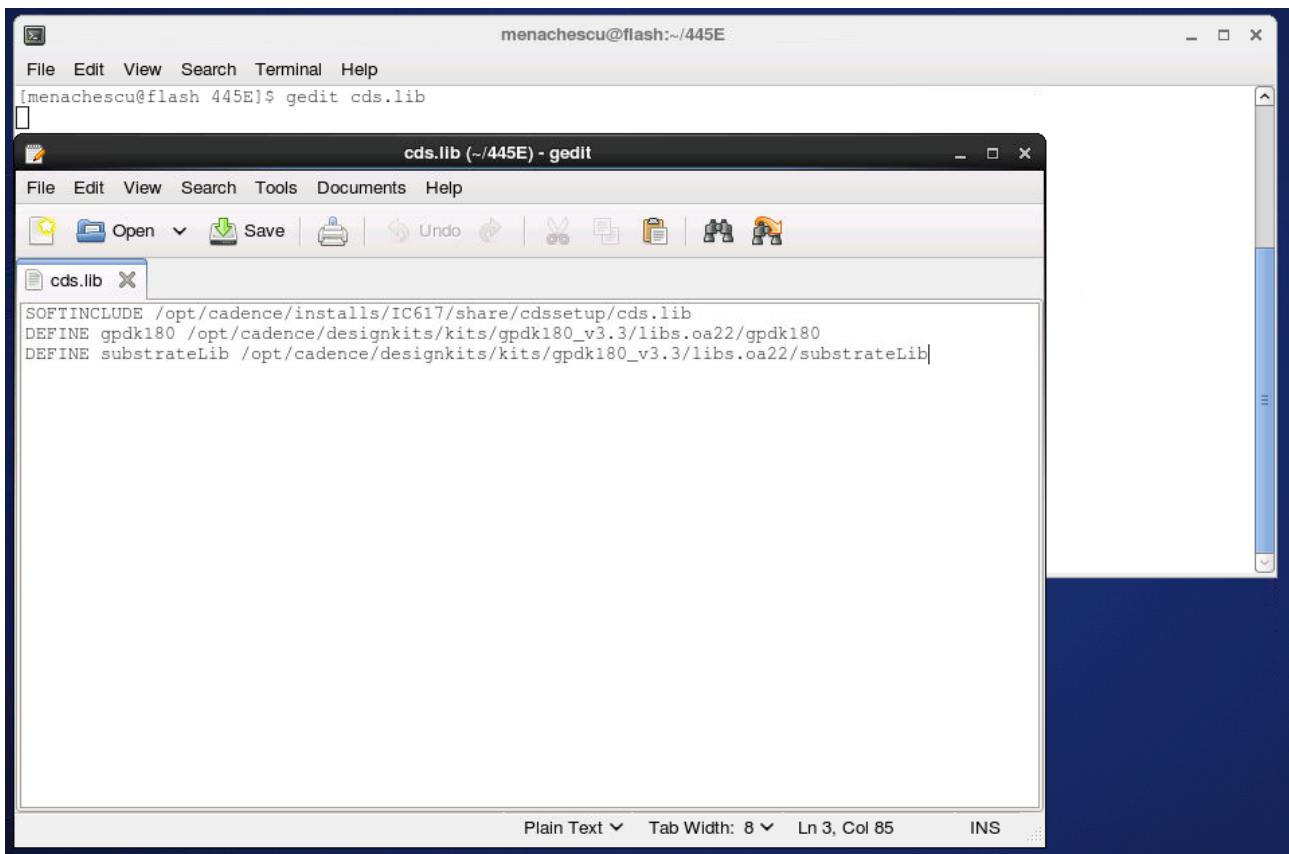


Figura A.2: Adaugarea librariei de componente in *Library Manager*

# LINUX COMMANDS CHEAT SHEET

SYSTEM		FILE PERMISSION RELATED	
uname -a	=>Display linux system information	chmod octal file-name	=>Change the permissions of file to octal
uname -r	=>Display kernel release information	Example	
uptime	=>Show how long the system has been running + load	chmod 777 /data/test.c	=>Set rwx permission for owner,group,world
hostname	=>Show system host name	chmod 755 /data/test.c	=>Set rwx permission for owner,rx for group and world
hostname -i	=>Display the IP address of the host	chown owner-user file	=>Change owner of the file
last reboot	=>Show system reboot history	chown owner-user:owner-group file-name	=>Change owner and group owner of the file
date	=>Show the current date and time	chown owner-user:owner-group directory	=>Change owner and group owner of the directory
cal	=>Show this month calendar		
w	=>Display who is online		
whoami	=>Who you are logged in as		
finger user	=>Display information about user		
HARDWARE		NETWORK	
dmesg	=>Detected hardware and boot messages	ip addr show	=>Display all network interfaces and ip address (a iproute2 command, powerful than ifconfig)
cat /proc/cpuinfo	=>CPU model	ip address add 192.168.0.1 dev eth0	=>Set ip address
cat /proc/meminfo	=>Hardware memory	ethtool eth0	=>Linux tool to show ethernet status
cat /proc/interrupts	=>Lists the number of interrupts per CPU per I/O device	mii-tool eth0	=>Linux tool to show ethernet status
lshw	=>Displays information on hardware configuration of the system	ping host	=>Send echo request to test connection
lsblk	=>Displays block device related information in Linux	whos domain	=>Get who is information for domain
free -m	=>Used and free memory (-m for MB)	dig domain	=>Get DNS information for domain
lspci -tv	=>Show PCI devices	dig -x host	=>Reverse lookup host
lsusb -tv	=>Show USB devices	host google.com	=>Lookup DNS ip address for the name
dmidecode	=>Show hardware info from the BIOS	hostname -i	=>Lookup local ip address
hdparm -i /dev/sda	=>Show info about disk sda	wget file	=>Download file
hdparm -T /dev/sda	=>Do a read speed test on disk sda	netstat -tulp	=>Listing all active listening ports
badblocks -s /dev/sda	=>Test for unreadable blocks on disk sda		
USERS		COMPRESSION / ARCHIVES	
id	=>Show the active user id with login and group	tar cf home.tar home	=>Create tar named home.tar containing home/
last	=>Show last logins on the system	tar xf tar	=>Extract the files from file.tar
who	=>Show who is logged on the system	tar czf file.tar.gz files	=>Create a tar with gzip compression
groupadd admin	=>Add group "admin"	gzip file	=>Compress file and renames it to file.gz
useradd -c "Sam Tomshi" "Sam"	=>g admin -m sam #Create user "sam"		
userdel sam	=>Delete user sam		
adduser sam	=>Add user "sam"		
usermod	=>Modify user information		
FILE COMMANDS		INSTALL PACKAGE	
ls -al	=>Display all information about files/ directories	rpm -i pkgnname.rpm	=>Install rpm based package
pwd	=>Show the path of current directory	rpm -e pkgnname	=>Remove package
mkdir directory-name	=>Create a directory		
rm file-name	=>Delete file		
rm -r directory-nam	=>Delete directory recursively		
rm -f file-name	=>Forcefully remove file		
rm -rf directory-name	=>Forcefully remove directory recursively		
cp file1 file2	=>Copy file1 to file2		
cp -r dir1 dir2	=>Copy dir1 to dir2, create dir2 if it doesn't exist		
mv file1 file2	=>Rename source to dest / move source to directory		
ln -s /path/to/file-name link-name	=#Create symbolic link to file-name		
touch file	=>Create or update file		
cat > file	=>Place standard input into file		
more file	=>Output contents of file		
head file	=>Output first 10 lines of file		
tail file	=>Output last 10 lines of file		
tail -f file	=>Output contents of file as it grows starting with the last 10 lines		
gpg -c file	=>Encrypt file		
gpg file.gpg	=>Decrypt file		
wc	=>print the number of bytes, words, and lines in files		
xargs	=>Execute command lines from standard input		
PROCESS RELATED		SEARCH	
ps	=>Display your currently active processes	grep pattern files	=>Search for pattern in files
ps aux   grep 'telnet'	=>Find all process id related to telnet process	grep -r pattern dir	=>Search recursively for pattern in dir
pmap	=>Memory map of process	locate file	=>Find all instances of file
top	=>Display all running processes	find /home/tom -name 'index*'*	=>Find files names that start with "index"
kill pid	=>Kill process with mentioned pid id	find /home -size +10000k	=>Find files larger than 10000k in /home
killall proc	=>Kill all processes named proc		
pkill process-name	=>Send signal to a process with its name		
bg	=>Resumes suspended jobs without bringing them to foreground		
fg	=>Brings the most recent job to foreground		
fg n	=>Brings job n to the foreground		
DISK USAGE		LOGIN (SSH AND TELNET)	
df -h	=>Show free space on mounted filesystems	ssh user@host	=>Connect to host as user
df -i	=>Show free inodes on mounted filesystems	ssh -p port user@host	=>Connect to host using specific port
fdisk -l	=>Show disks partitions sizes and types	telnet host	=>Connect to the system using telnet port
du -ah	=>Display disk usage in human readable form		
du -sh	=>Display total disk usage on the current directory		
findmnt	=>Displays target mount point for all filesystem		
mount device-path mount-point	=>Mount a device		
DIRECTORY TRAVERSE		FILE TRANSFER	
cd ..	=>To go up one level of the directory tree	scp	
cd	=>Go to SHOME directory	scp file.txt server2:/tmp	=>Secure copy file.txt to remote host /tmp folder
cd /test	=>Change to /test directory	rsync	
		rsync -a /home/apps /backup/	=>Synchronize source to destination



MORE DETAILED : [HTTP://LINOXIDE.COM/GUIDE/LINUX-COMMAND-SHELF.HTML](http://LINOXIDE.COM/GUIDE/LINUX-COMMAND-SHELF.HTML)

Figura A.3: Linux - Cheat Sheet.

## A.3 Utilizarea programului Cadence Virtuoso

Pentru a porni Cadence Virtuoso in directorul nou creat se foloseste comanda *virtuoso*.

### A.3.1 Configurari initiale in Cadence Virtuoso

Primul pas pentru a putea desena un circuit este sa cream o librarie in care sa depozitam schemele desenate in acest laborator. Pentru aceasta, in cazul in care fereastra *Library Manager* nu a aparut la pornirea programului, aceasta poate fi accesata din *Tools · Library Manager*.

Din fereastra *Library Manager* se acceseaza meniul *File → New → Library* (vezi Figure A.4).

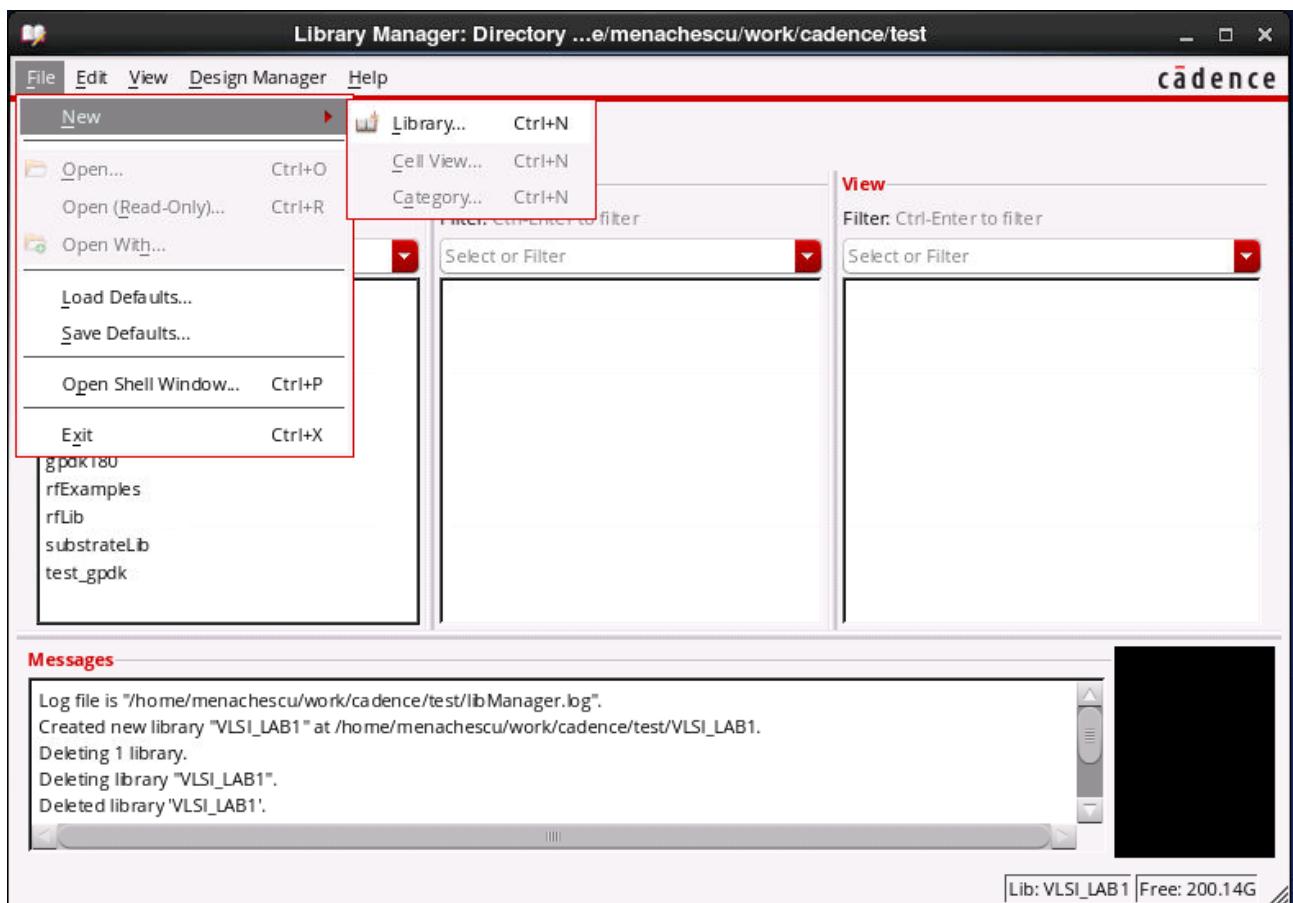


Figura A.4: Virtuoso - New Library.

Pentru laboratorul 1 vom crea libraria cu numele *vlsi1617\_44xE* (vezi Figura A.5 unde *x* e numele grupei) dupa care apasam *OK*.

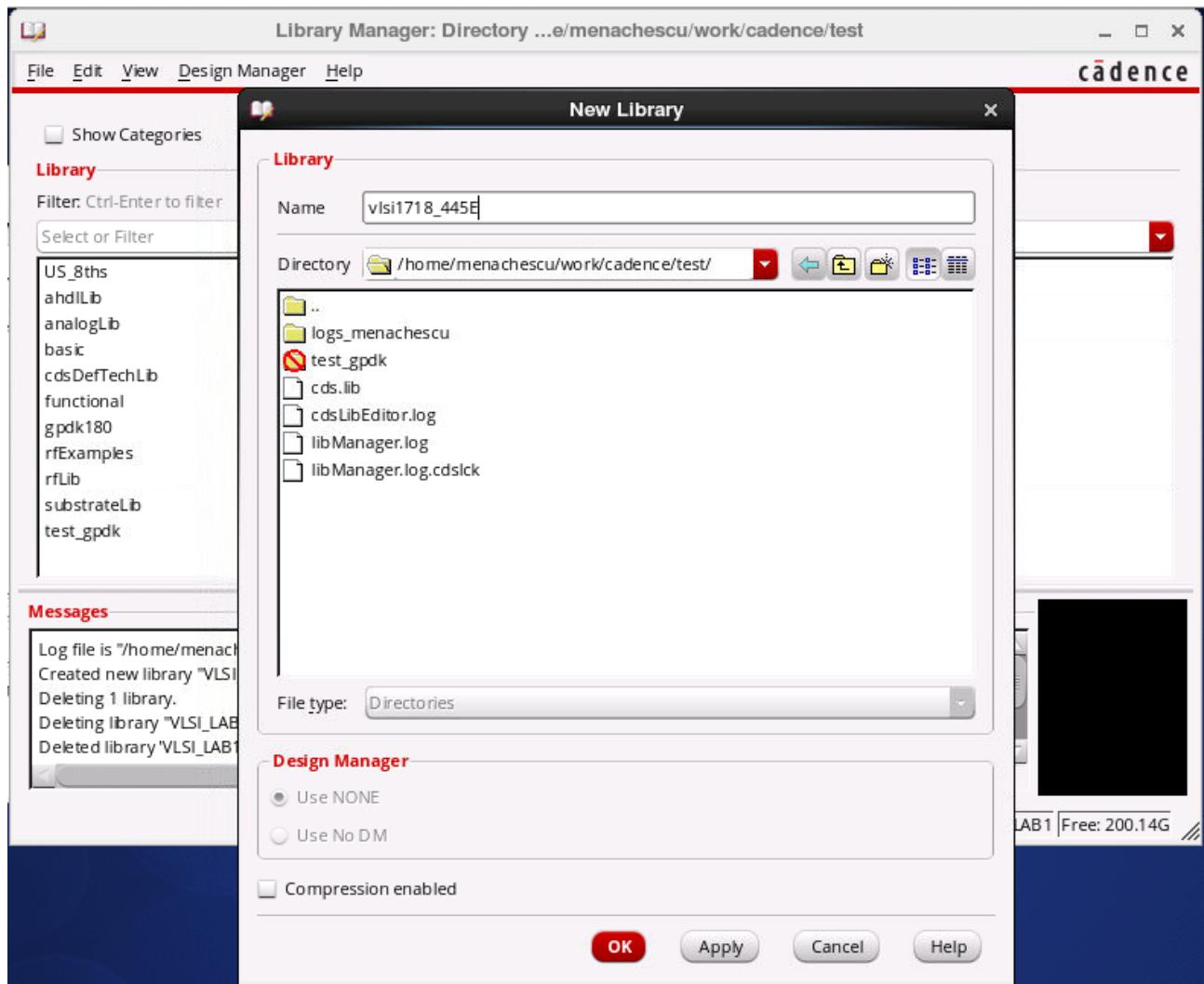


Figura A.5: Virtuoso - Library Name.

Libraria nou creata trebuie atasata unei tehnologi *CMOS*. In cadrul acestui laborator vom folosi tehnologia *gpdk180* avand latimea minima a portii unui tranzistor de  $180nm$ . Vom alege in noua fereastra *Attach to an existing Library* (vezi Figura A.6) dupa care vom apasa tasta *OK*.

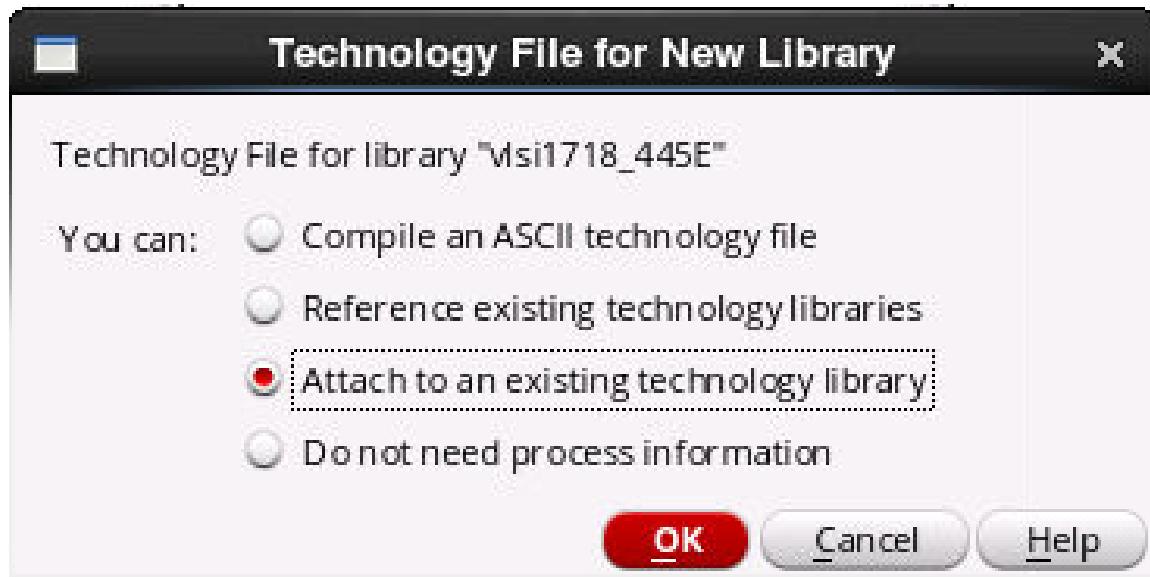


Figura A.6: Virtuoso - Attach Technology to Library.

Din lista se va selecta tehnologia *gpdk180* (vezi Figura A.7) dupa care se apasa tasta *OK*.



Figura A.7: Virtuoso - Attach Technology to Library.

În Figure A.8 și Figure A.9 este descris modul în care se creează o nouă schemă și atribuirea unui nume acestuia. Din fereastra *Library Manager* se accesează meniul *File* → *New* → *Cellview*. Primul exemplu pe care il vom crea este un *inversor*, prin urmare vom atribui numele primei scheme *inverter*.

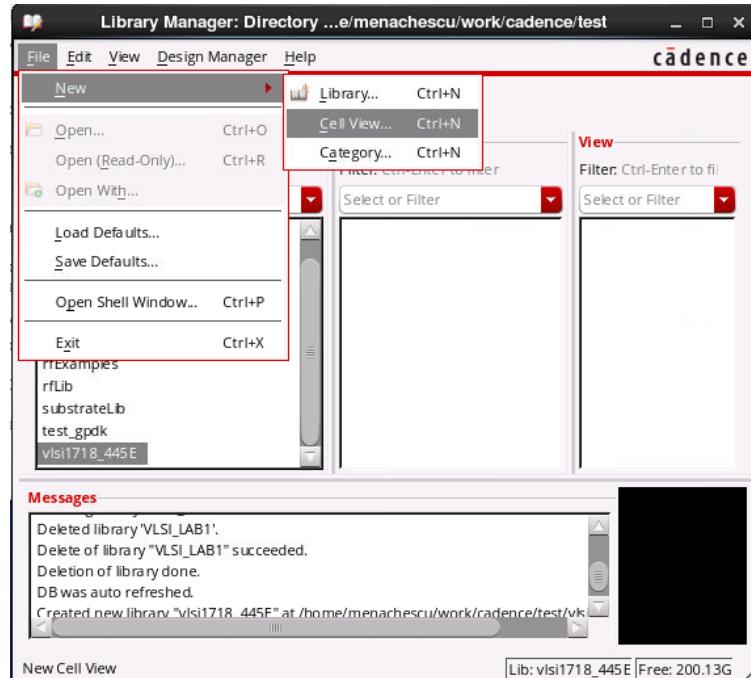


Figura A.8: Virtuoso - Create new cellview.



Figura A.9: Virtuoso - Name new Cellview.

### A.3.2 Desenarea circuitului

Adăugarea unei componente noi se face folosind tasta *i* sau din meniul *Create* → *Instance* (găsiți acest pas în Figura A.10). Elementele uzuale de circuit ideale, cum ar fi rezistențe, condensatoare, inductoare și diode, se mai pot adăuga din libraria *analogLib*. Pentru adaugarea unui tranzistor din tehnologia folosită se folosește meniul *Browse* (găsiți acest pas în Figura A.11). Navigam la libraria *gpdk180* categoria *mos*, celula *nmos*, view *symbol* (găsiți acest pas în Figura A.12).

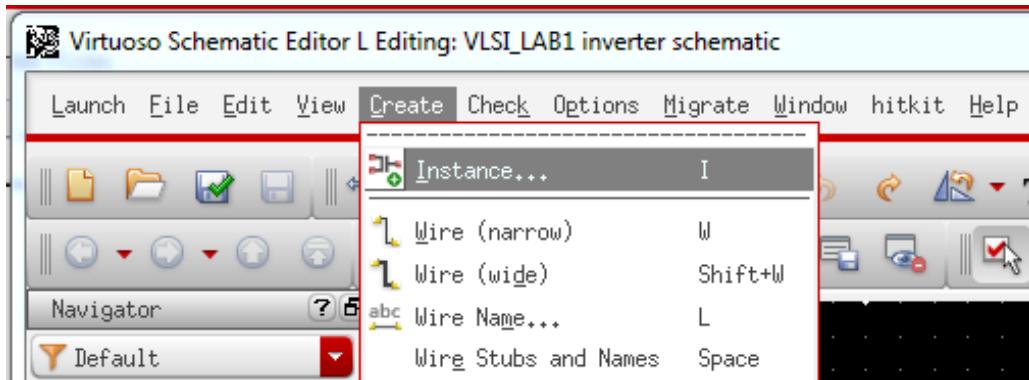


Figura A.10: Virtuoso - Add new component.

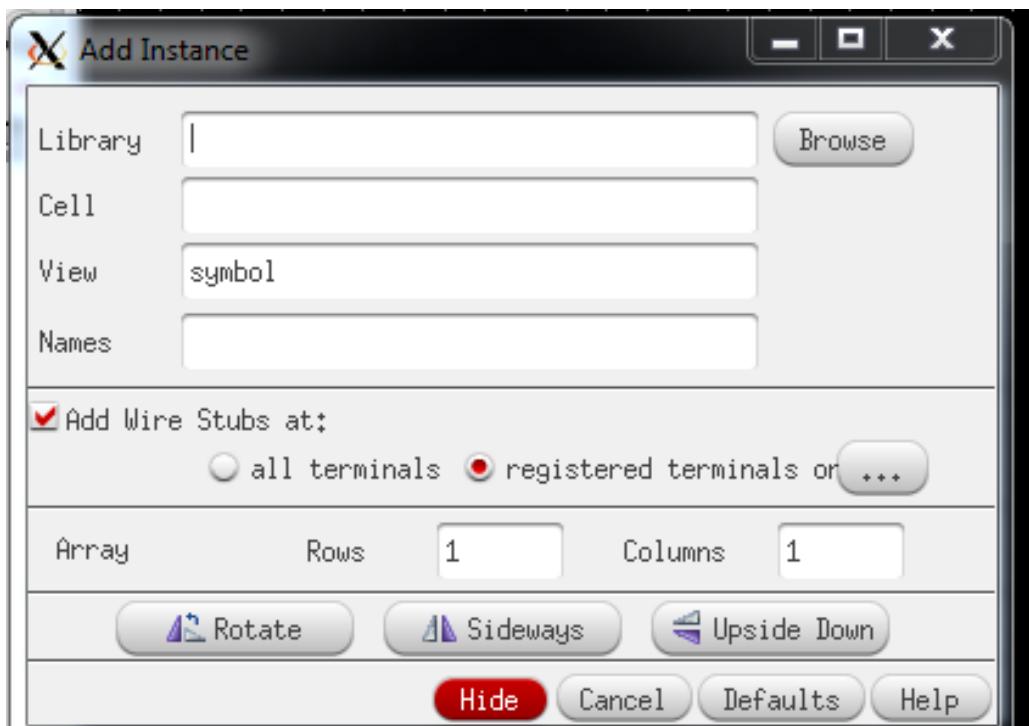


Figura A.11: Virtuoso - Add new component (2).

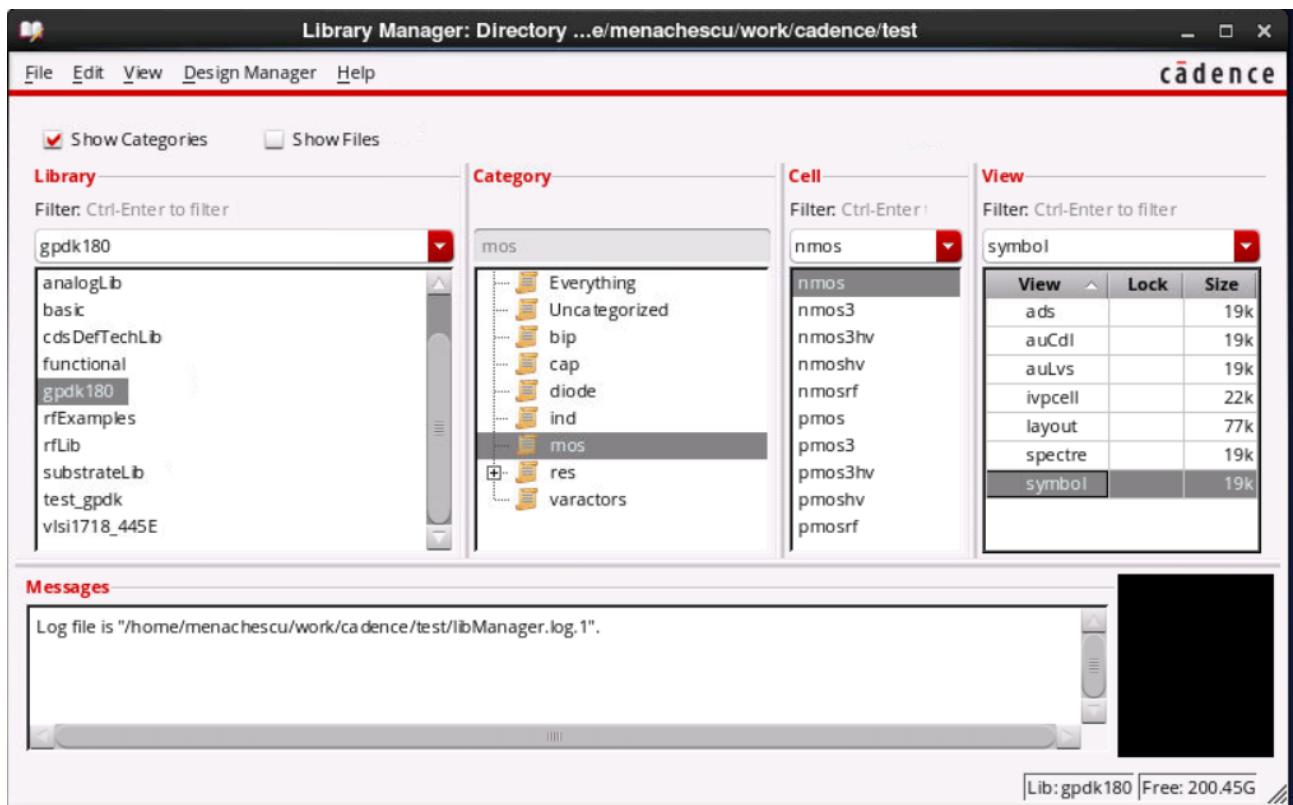


Figura A.12: Virtuoso - Add new component (3).

Interconectarea se face folosind tasta *w* (wire) sau din meniul *Create* → *Instance* (găsiți acest pas în Figure A.13), după care *Left Click* în punctul de pornire și *Left Click* la destinație. Porturile de intrare și ieșire ai circuitului se accesează folosind tasta *p* sau din meniul *Create* → *Pin* (găsiți acest pas în Figure A.14). Cu ajutorul submeniuului *Direction* putem modifica tipul portului, e.g., *input* (găsiți acest pas în Figure A.15), *output* (găsiți acest pas în Figure A.16), sau *input/output* (găsiți acest pas în Figure A.17).

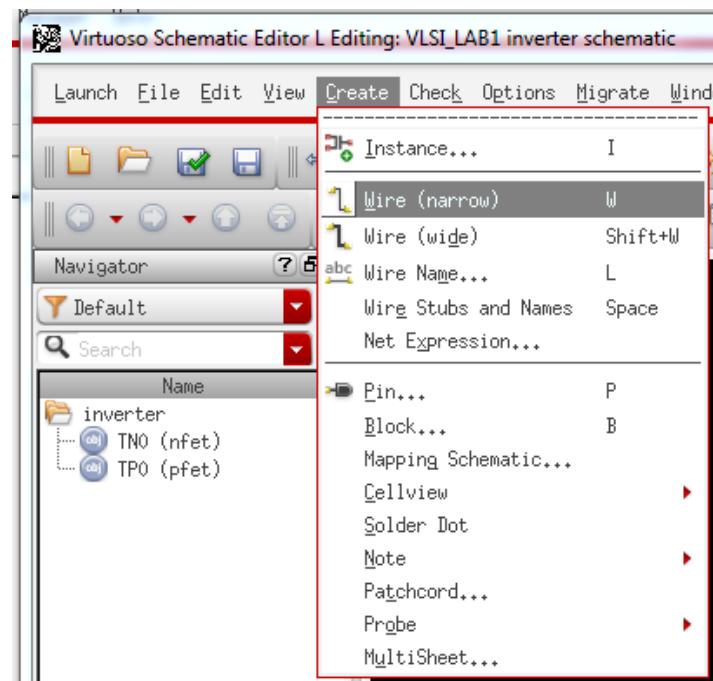


Figura A.13: Virtuoso - Add new wire.

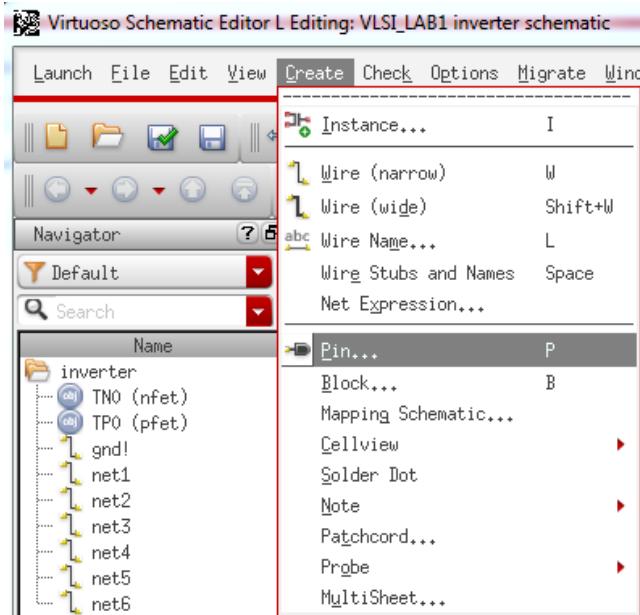


Figura A.14: Virtuoso - Add new pin.



Figura A.15: Virtuoso - Add new input pin.

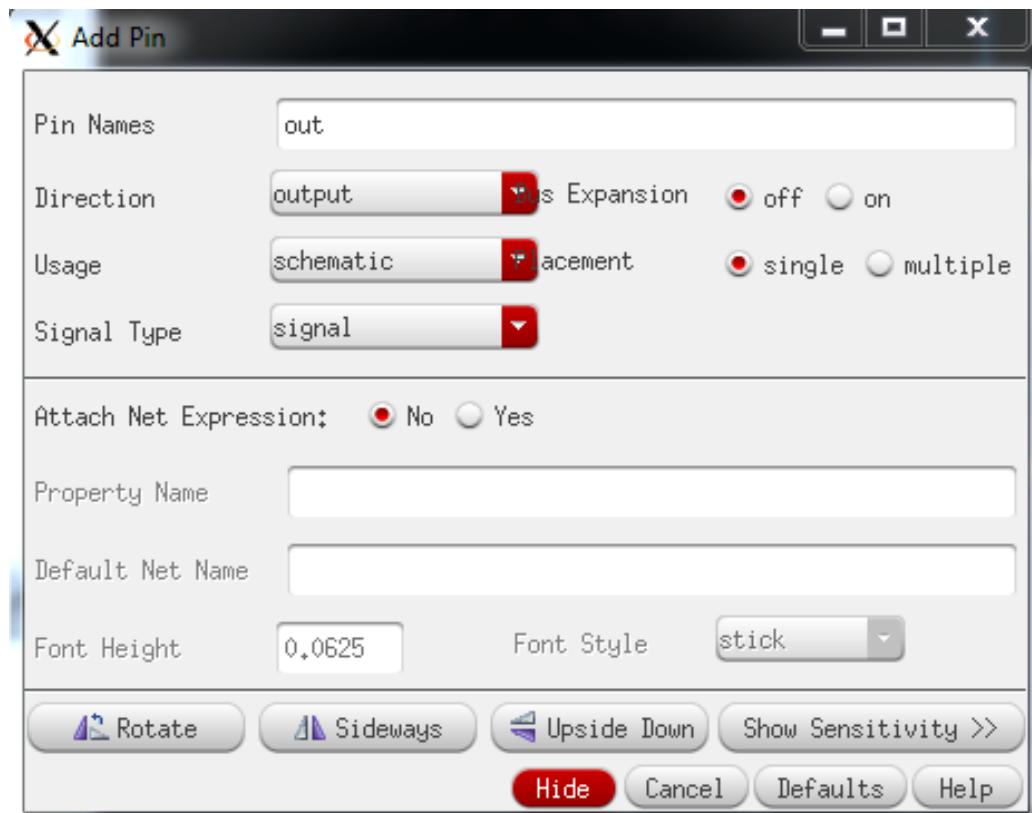


Figura A.16: Virtuoso - Add new output pin.

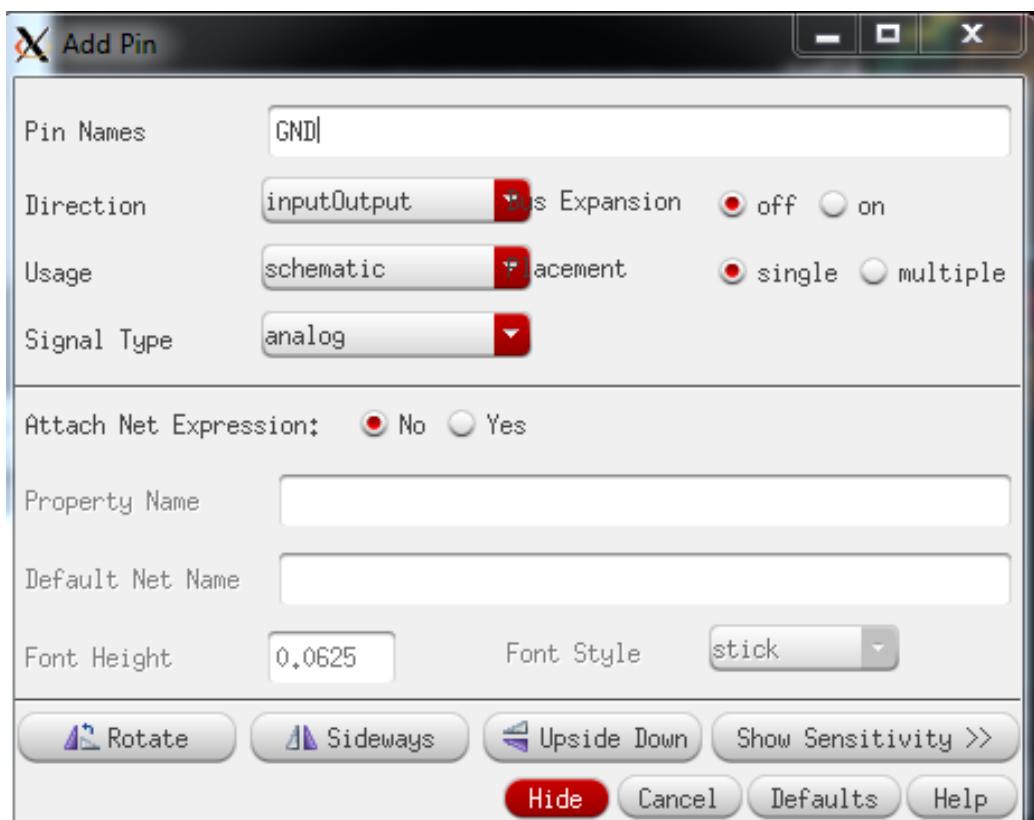


Figura A.17: Virtuoso - Add new global pin - power si ground.

Schema finală a inversorului pe care ne-am propus să-l investigăm se găseste în Figura A.18. Observăm că am folosit un tranzistor MOS cu canal de tip  $n$ , i.e.,  $nmos$ , și un tranzistor cu canal de tip  $p$ , i.e.,  $pmos$ . Substratul tranzistorului  $pmos$  este conectat la sursa acestuia, respectiv la potențialul  $VDD$ .

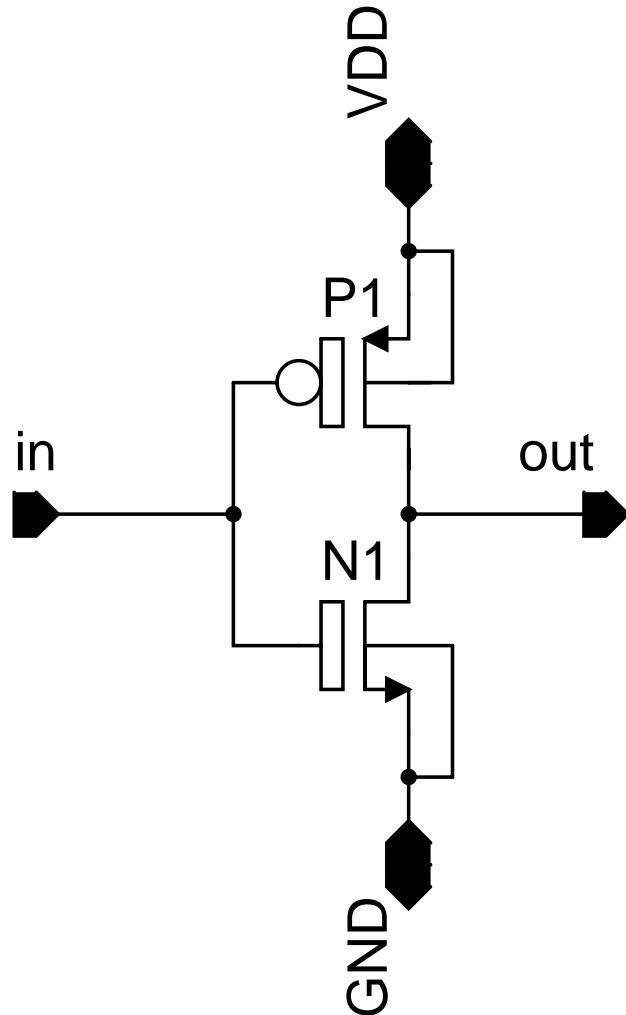


Figura A.18: Virtuoso - Interter Schematic.

Modificarea parametrilor dispozitivelor se realizează astfel:

1. Se selectează componenta respectivă,
2. Se vizualizează și editează lista parametrilor folosind tasta  $q$ ,
3. Se modifică în mod corespunzător parametrii doriti.

### A.3.3 Crearea unui simbol

Crearea unui *view symbol* asociat unei scheme, in cazul nostru inversorul, se executa din meniul *Create* → *Celview* → *From Celview*(găsiți acest pas în Figure A.19 si Figure A.20). Apasati tasta *OK*.

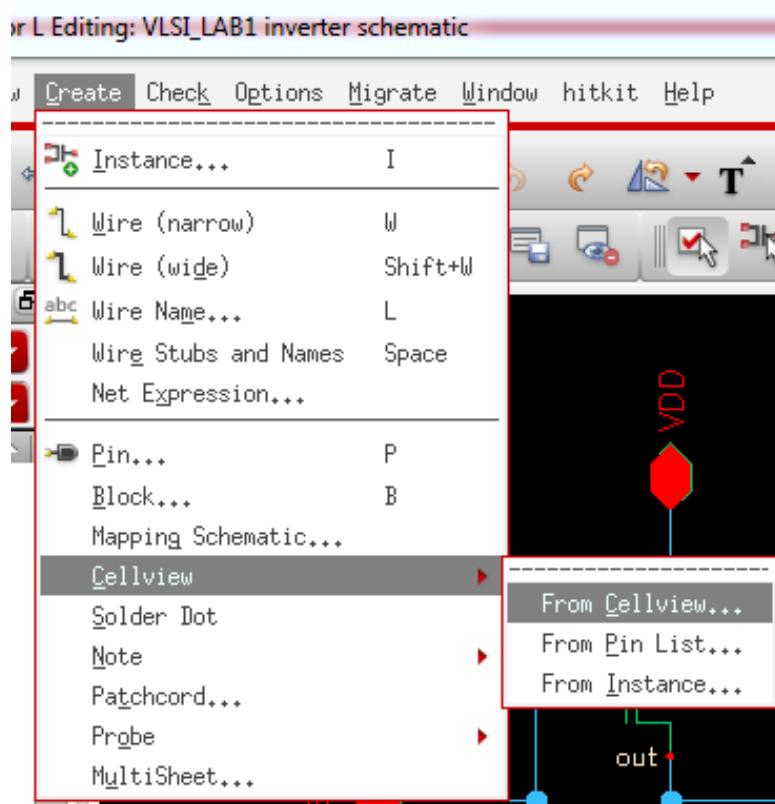


Figura A.19: Virtuoso - Create Inverter Symbol.



Figura A.20: Virtuoso - Create Inverter Symbol (2).

O noua fereastra se va deschide in care veti putea alege pozitionarea pinilor, i.e., stanga, dreapta, jos, sus (găsiți acest pas în Figure A.21). Dupa apasarea butonului *OK* puteti vizualiza simbolul nou creat, exact ca cel din Figure A.22.

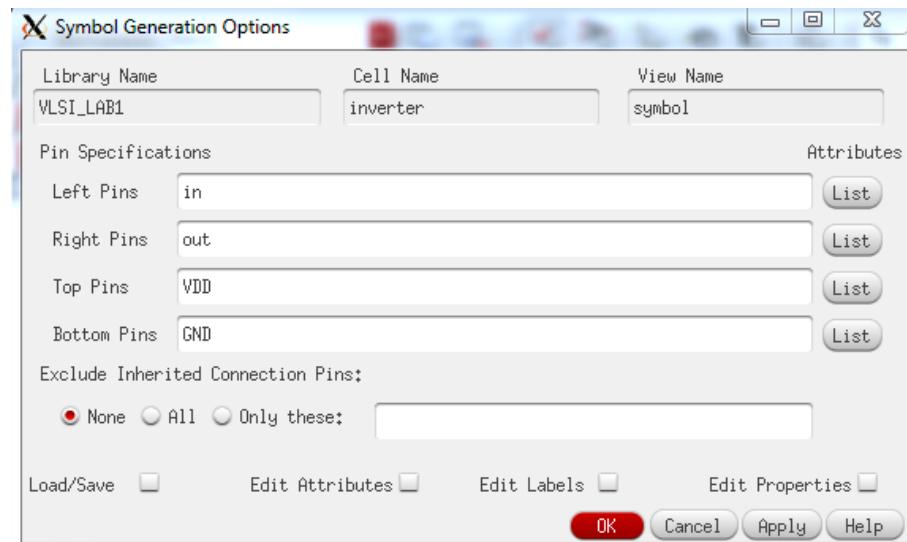


Figura A.21: Virtuoso - Create Inverter Symbol (3).

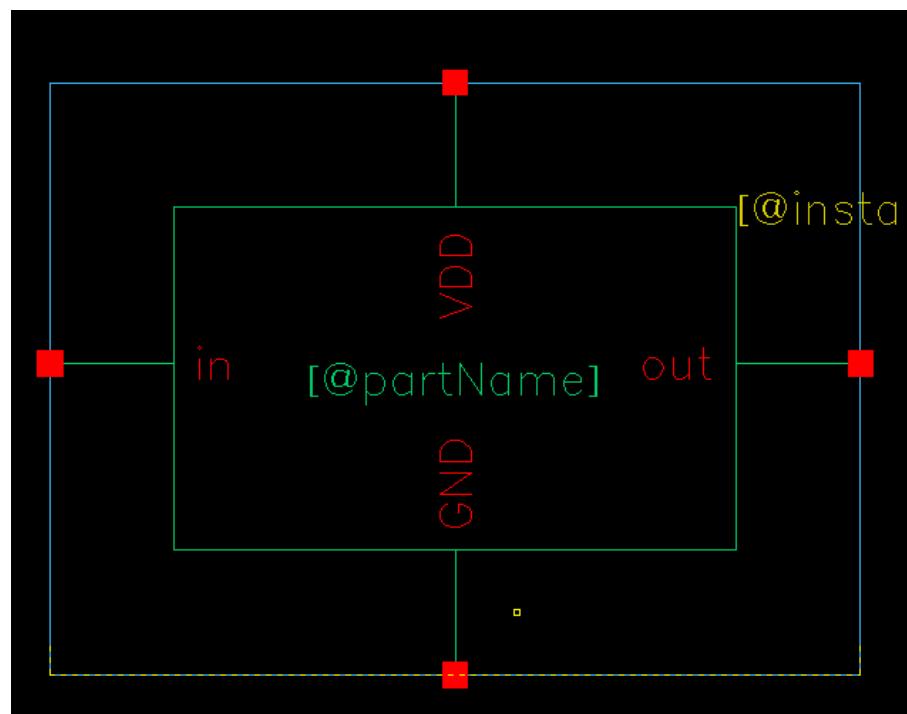


Figura A.22: Virtuoso - Create Inverter Symbol (4).

Simbolul nou creat se recomanda sa fie modificat astfel incat sa se obtina o abstractizare ierarhica a schemei. Figure A.23 si Figure A.24 prezinta procesul de modificare al simbolului pana la varianta finala.

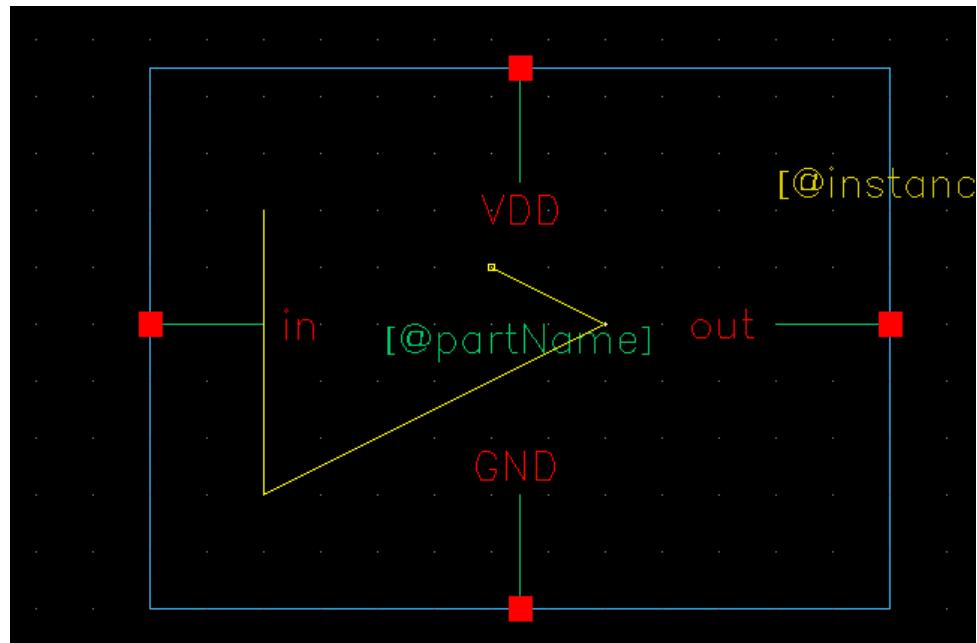


Figura A.23: Virtuoso - Modify Inverter Symbol.

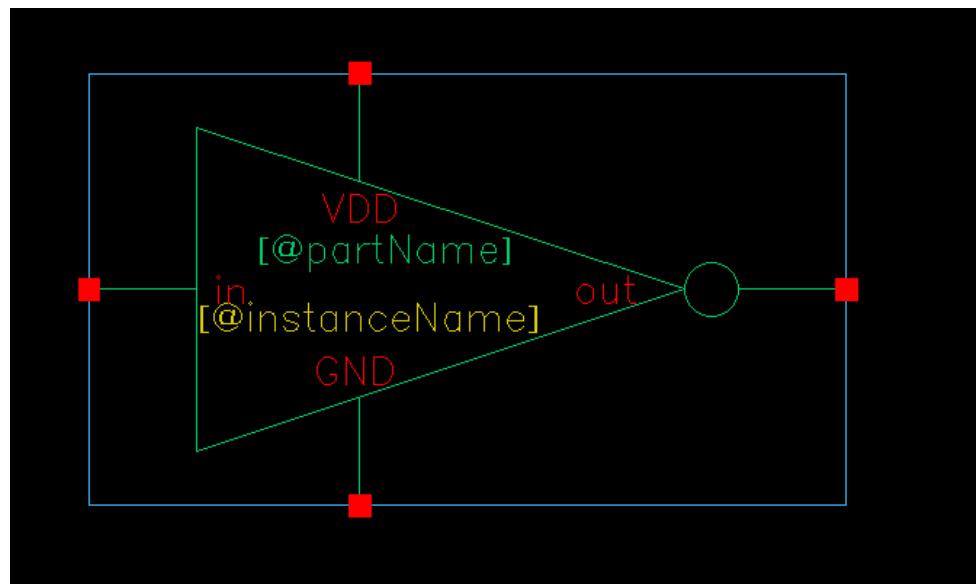


Figura A.24: Virtuoso - Modify Inverter Symbol (2).

#### A.3.4 Testbench-ul inversorului

Odată finalizat simbolul, urmatorul pas este să testam circuitul. Schema finală la care vrem să ajungem este prezentată în Figure A.25. În continuare vor fi enumerate pașii necesari desenării schemei de test:

1. Se crează un nou *cellview* de tipul *schematic* (vezi Figure A.26),
2. Se instantiază inversorul creat folosind *view-ul symbol*,
3. Se instantiază pinii globali de alimentare *vdd!* și *gnd!* din libraria *analogLib* (vezi Figure A.27),
4. Se instantiază o sursă de alimentare *vdc* din libraria *analogLib* (vezi Figure A.28), setându-se parametrul *DC Voltage* = 1.8 (vezi Figure A.29),
5. Se instantiază o sursă *vpwl* din libraria *analogLib* (vezi Figure A.30), setându-se parametrii *Number of pair of points* = 3 și restul parametrilor conform Figure A.31,
6. Optional, pentru a identifica anumite conexiuni din circuit, acestora li se poate atribui un nume dedicat folosind tasta *L* (label) conform Figure A.32.

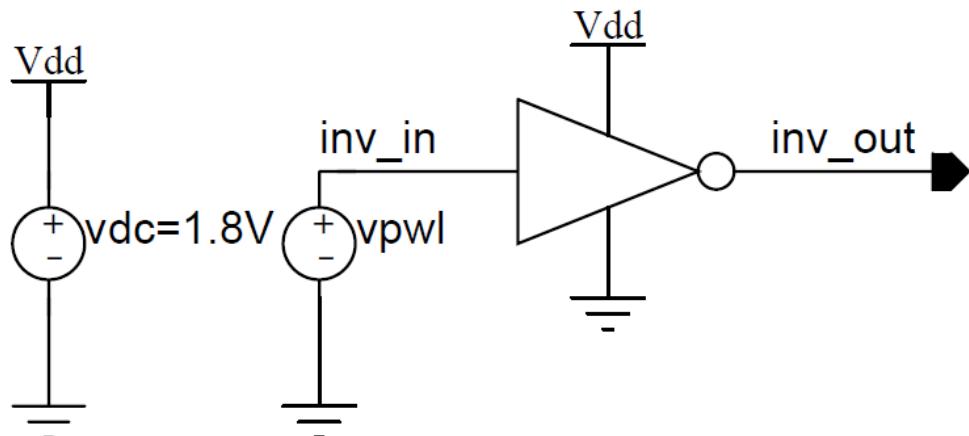


Figura A.25: Virtuoso - Schema testbench-ului inversorului.

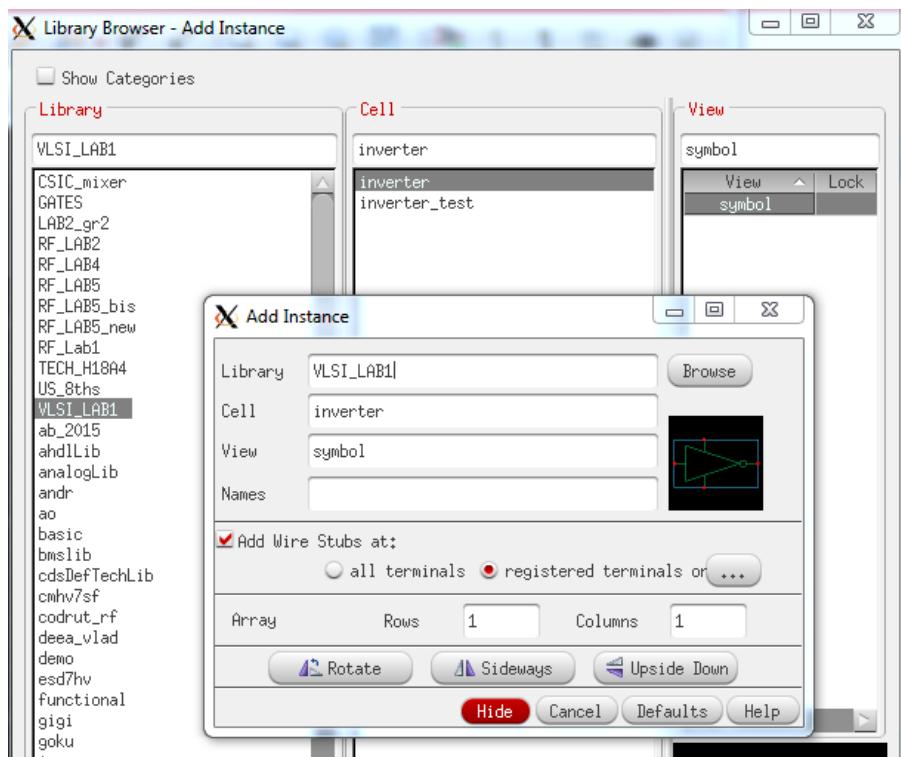


Figura A.26: Virtuoso - Testbench Inverter.

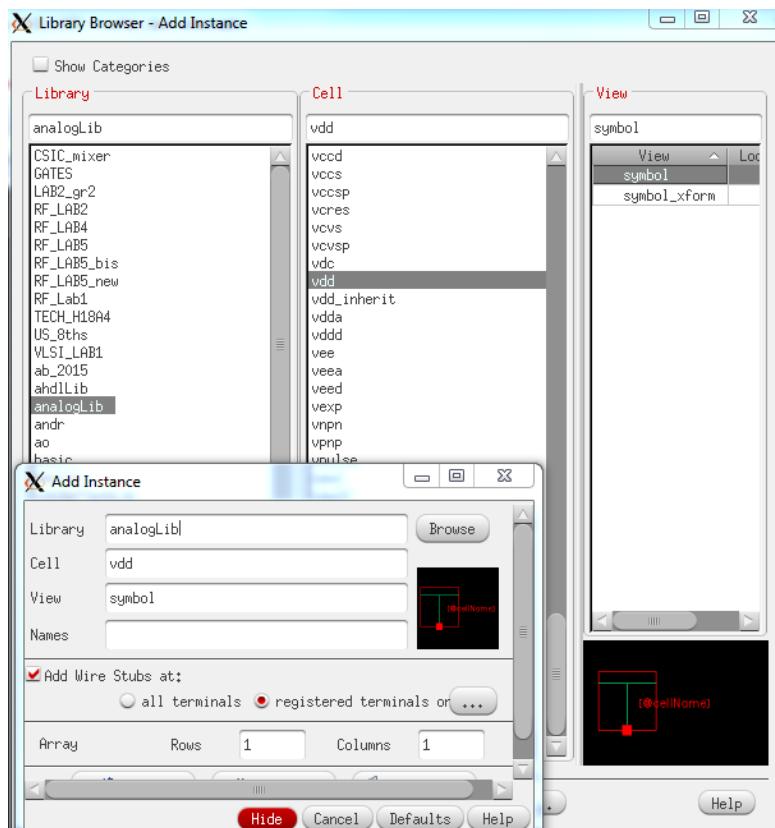


Figura A.27: Virtuoso - Testbench Inverter (2).

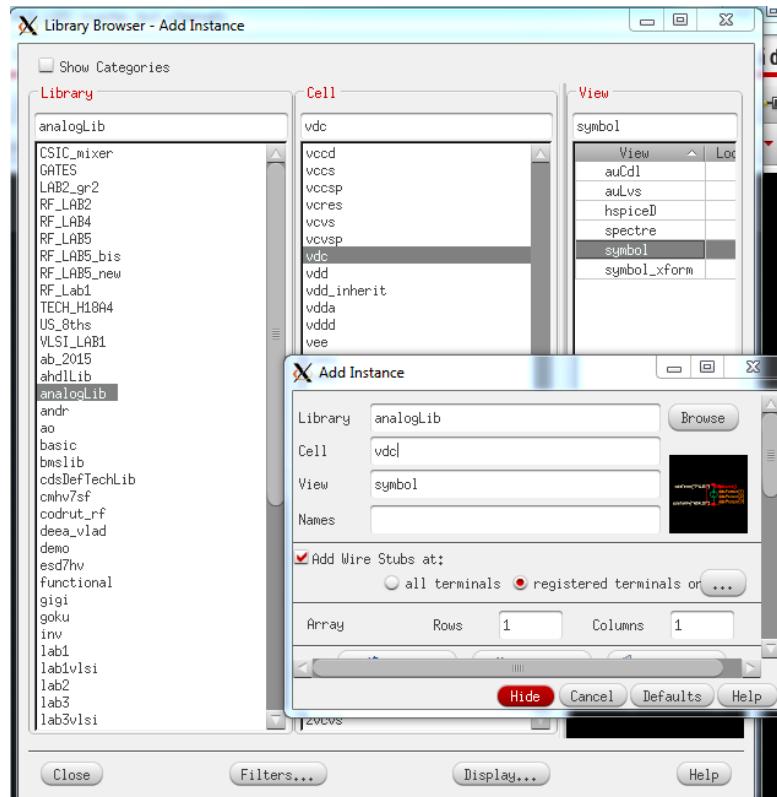


Figura A.28: Virtuoso - Testbench Inverter DC source (1).

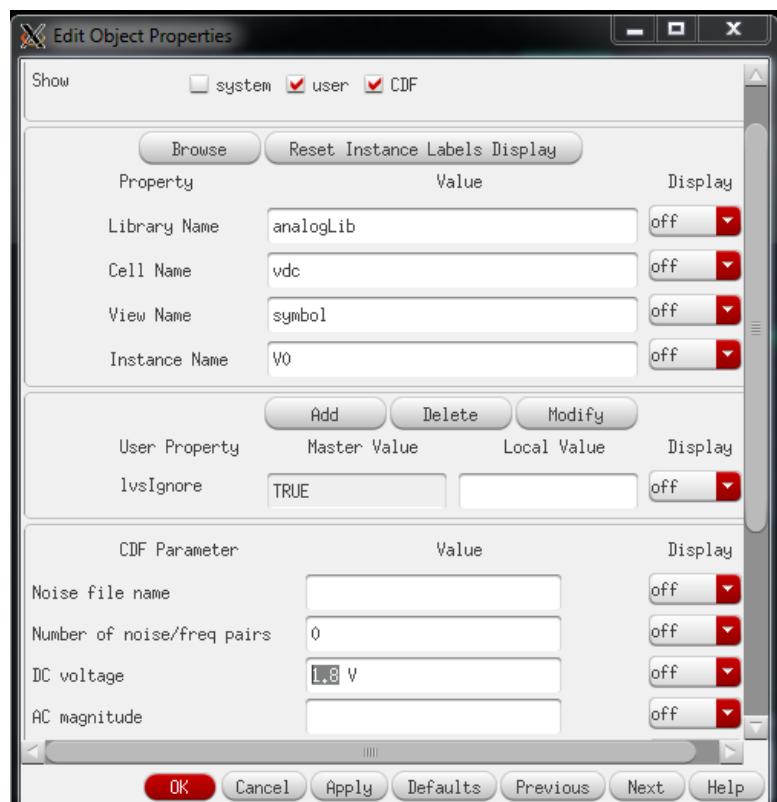


Figura A.29: Virtuoso - Testbench Inverter DC source (2).

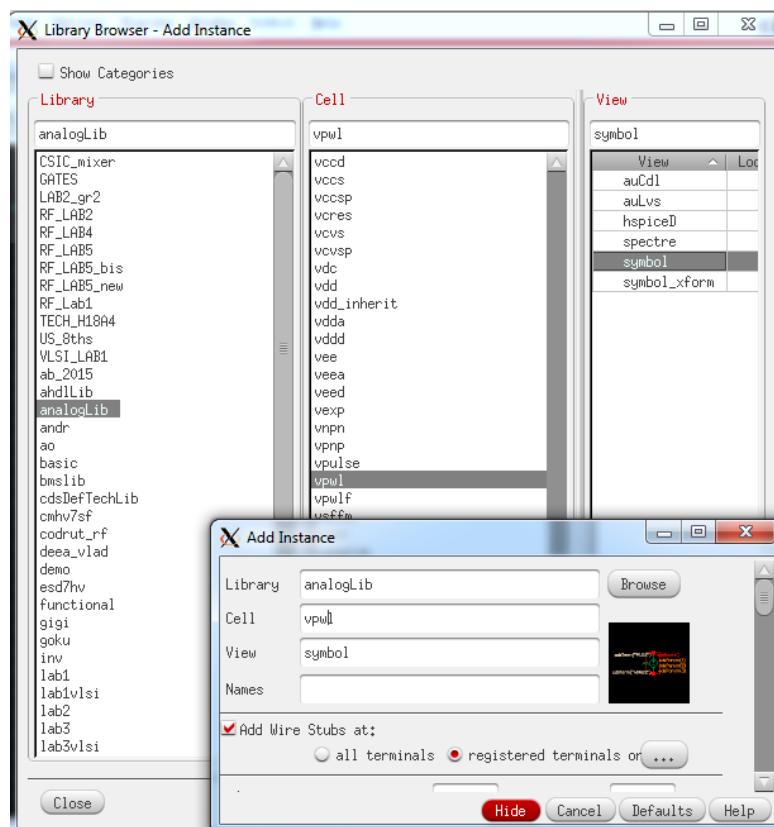


Figura A.30: Virtuoso - Testbench Inverter vpwl source.

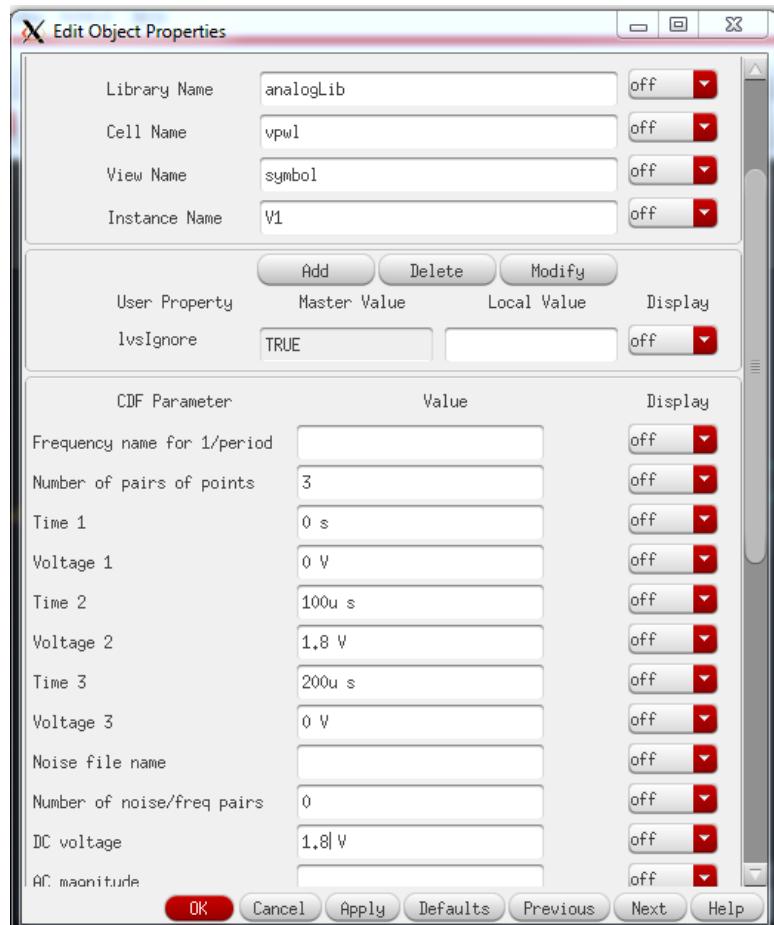


Figura A.31: Virtuoso - Testbench Inverter vpwl source (2).

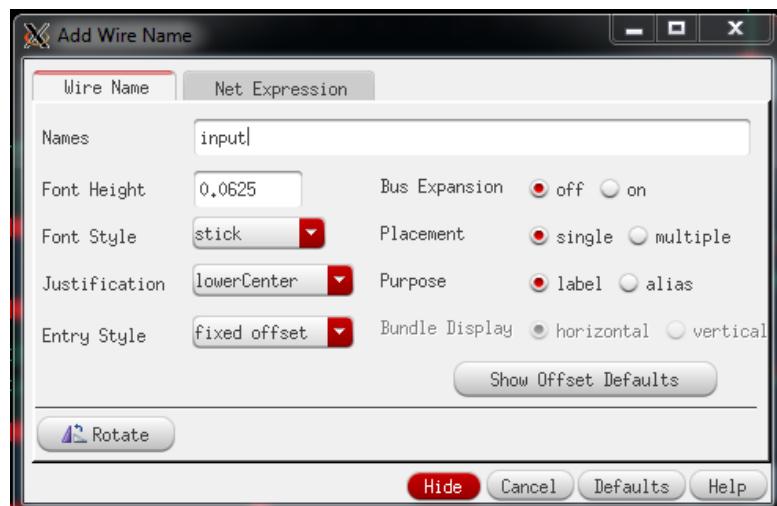


Figura A.32: Virtuoso - Label wire.

## A.3.5 Simularea Circuitului

### A.3.5.1 Analiza DC

Permite balearea unui domeniu specificat al următoarelor variabile și vizualizarea semnalului de ieșire pentru acest domeniu de variație:

1. Valoarea de curent continuu a unei surse de tensiune sau a unei surse de curent;
2. Valoarea temperaturii;
3. Valoarea unui parametru de model sau parametru global.

După finalizarea desenării schemei inversorului, se porneste simulatorul *Analog Design Environment* folosind meniul *Launch → ADE L* (vezi Figure A.33). Analiza *DC* se poate activa din meniul *Analyses → Choose → DC* cu următoarele opțiuni (vezi Figure A.34):

1. activam *Save · DC · Operating · Points*;
2. activam butonul *Enabled*, exact ca in Figure A.35.

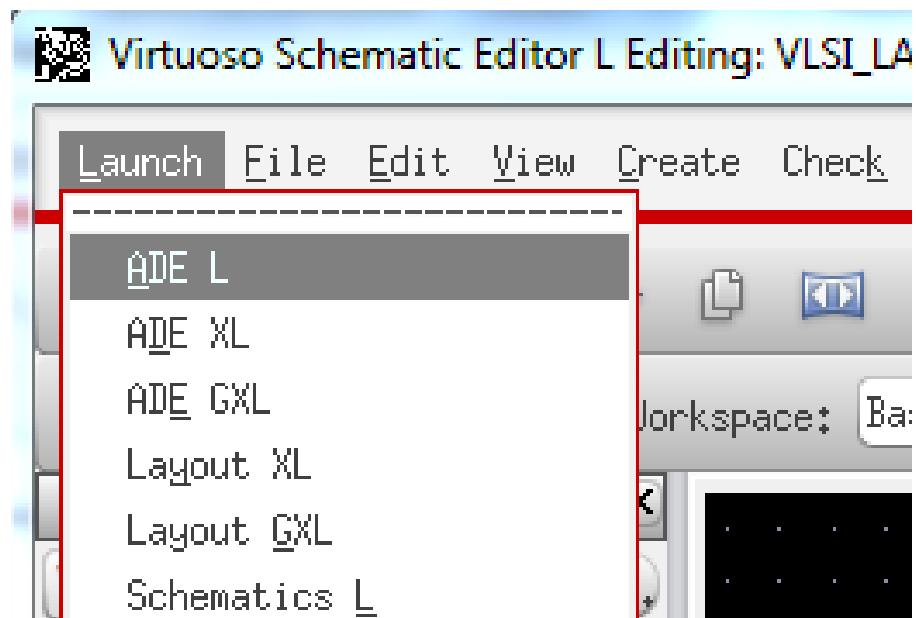


Figura A.33: Virtuoso - Launch ADE-L.

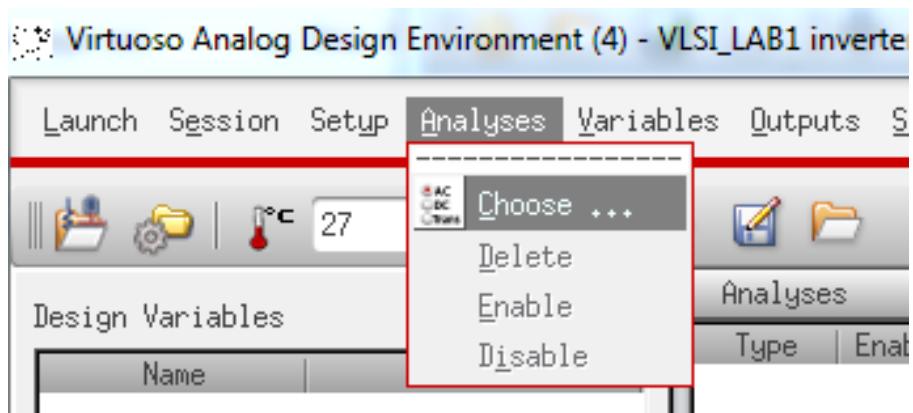


Figura A.34: Virtuoso ADEL - Choose analyses.

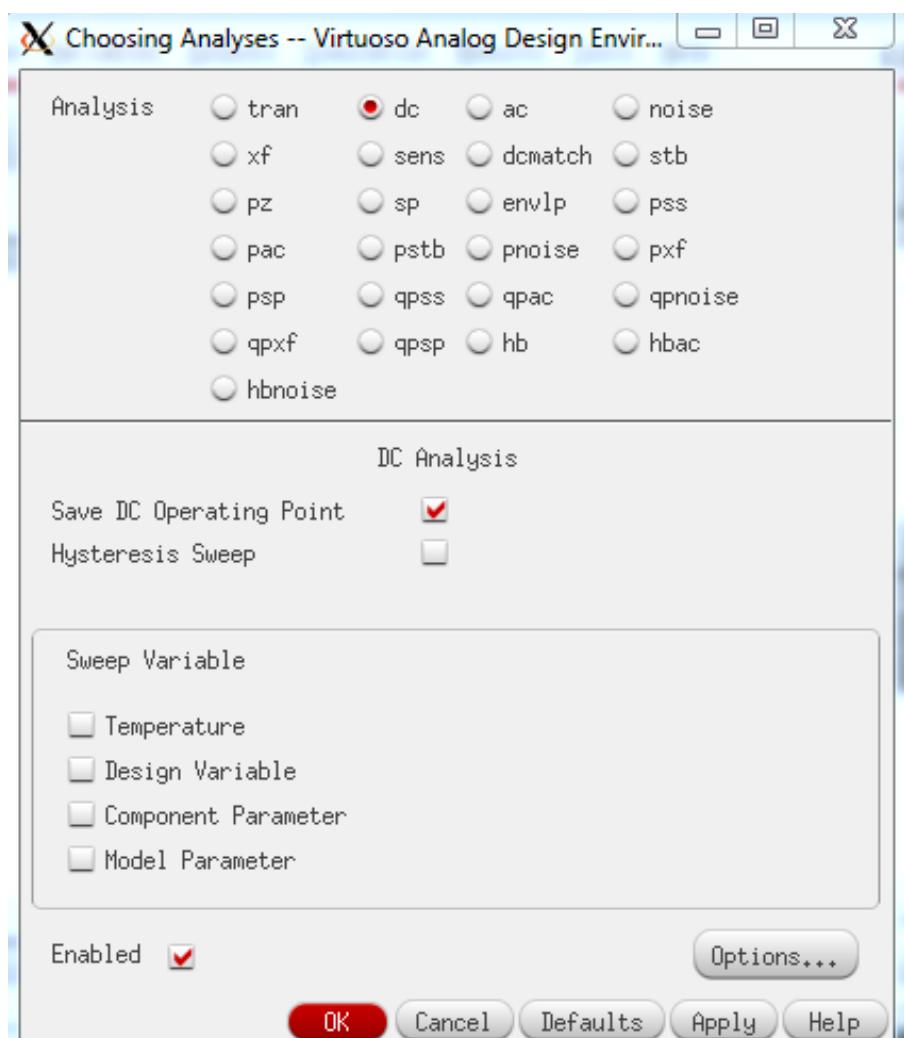


Figura A.35: Virtuoso ADEL - DC analyses.

Următorul pas este marcarea semnalelor de care suntem interesati in aceasta simulare in scopul vizualizării ulterioare ale acestora. Aceasta operatiune se realizeaza din meniul *Outputs* → *To be saved* → *Select on Design* (vezi Figure A.36). Dupa care se foloseste comanda *Left Click* pe conexiuni pentru tensiuni si *Left Click* pe porturi pentru curenti.

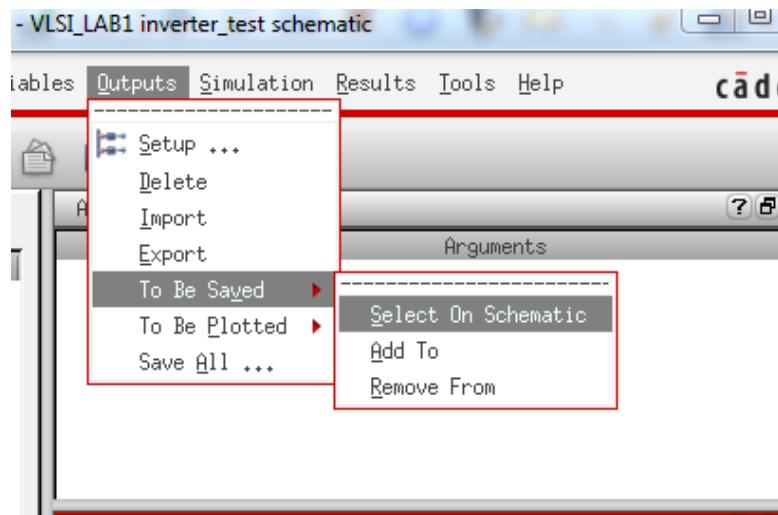


Figura A.36: Virtuoso ADEL - Save Outputs.

### A.3.5.2 Analiza Transient

Analiza *tran* se poate activa din meniul *Analyses* → *Choose* → *tran* cu următoarele opțiuni (vezi Figura A.34):

1. selectam *Stop time = 200us*;
2. activam butonul *Enabled*, exact ca în Figura A.37.

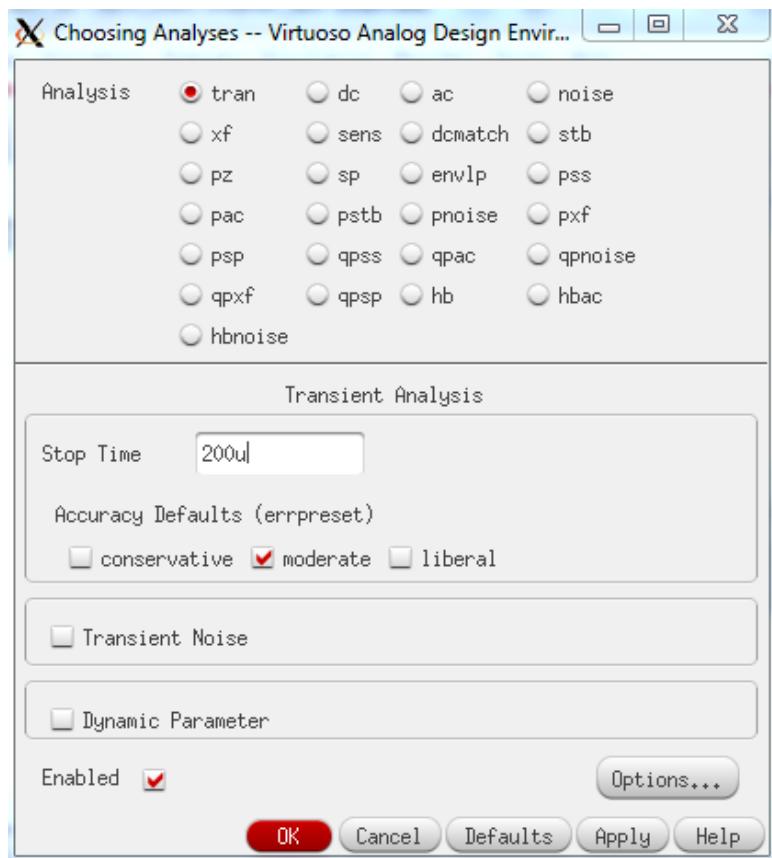


Figura A.37: Virtuoso ADEL - Transient analyses.

În final, pentru pornirea simulării se folosește meniul *Simulation* → *Run* sau butonul verde *Play*. Întotdeauna înaintea unei simulări trebuie să salvăm circuitul cu butonul *Check&Save*.

### A.3.6 Rezultatele simulării DC si Transient

Intr-o analiza tranzitorie putem identifica o simulare care se termină cu succes atunci când într-o nouă fereastră sunt afisate semnalele selectate pentru vizualizare (vezi Figure A.38).

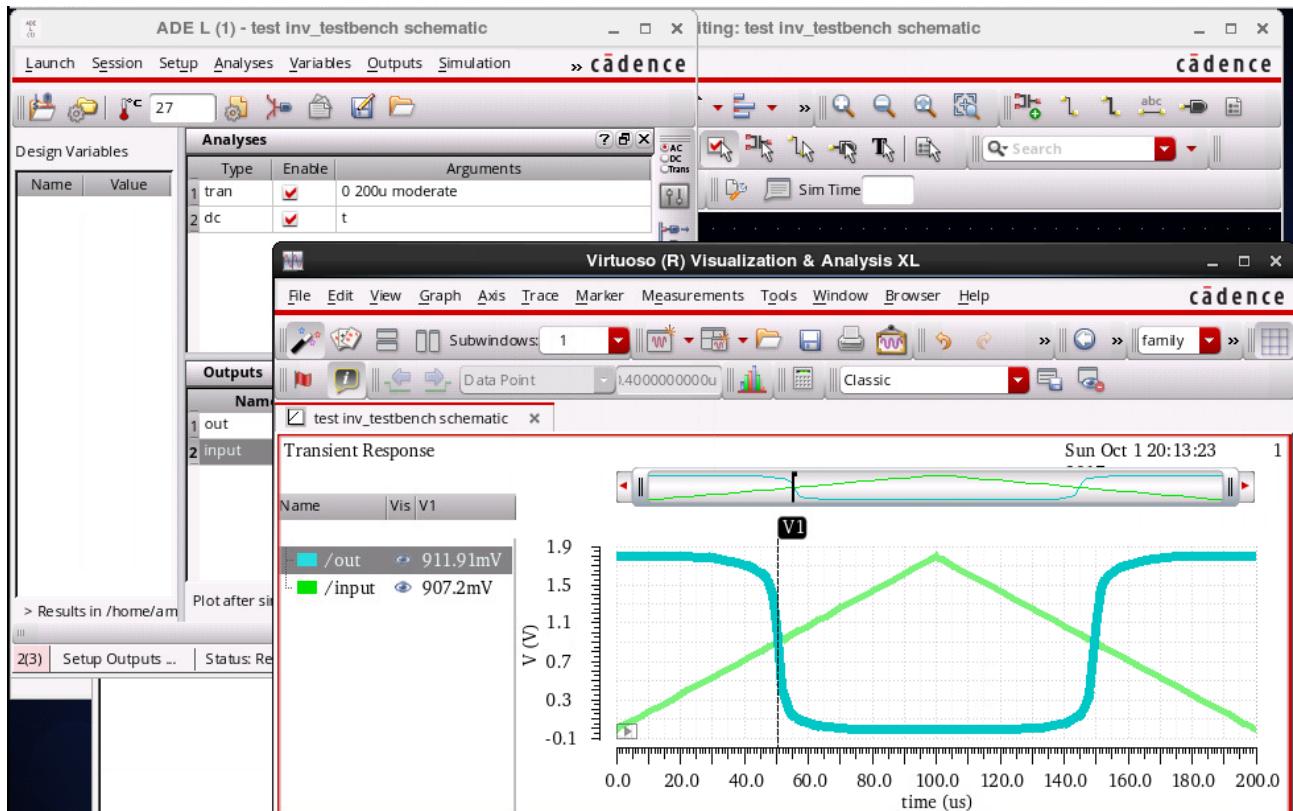


Figura A.38: Virtuoso ADEL - Results from Transient analyses.

Deoarece am rulat împreună cu analiza tranzitorie și o analiza *DC*, putem vizualiza direct pe schema valoarea tensiunii din toate nodurile circuitului din meniu *Annotate → DC Node Voltages*, vezi Figure A.39, sau parametrii dispozitivelor folosite din meniu *Annotate → DC Operating Points* Figure A.40.

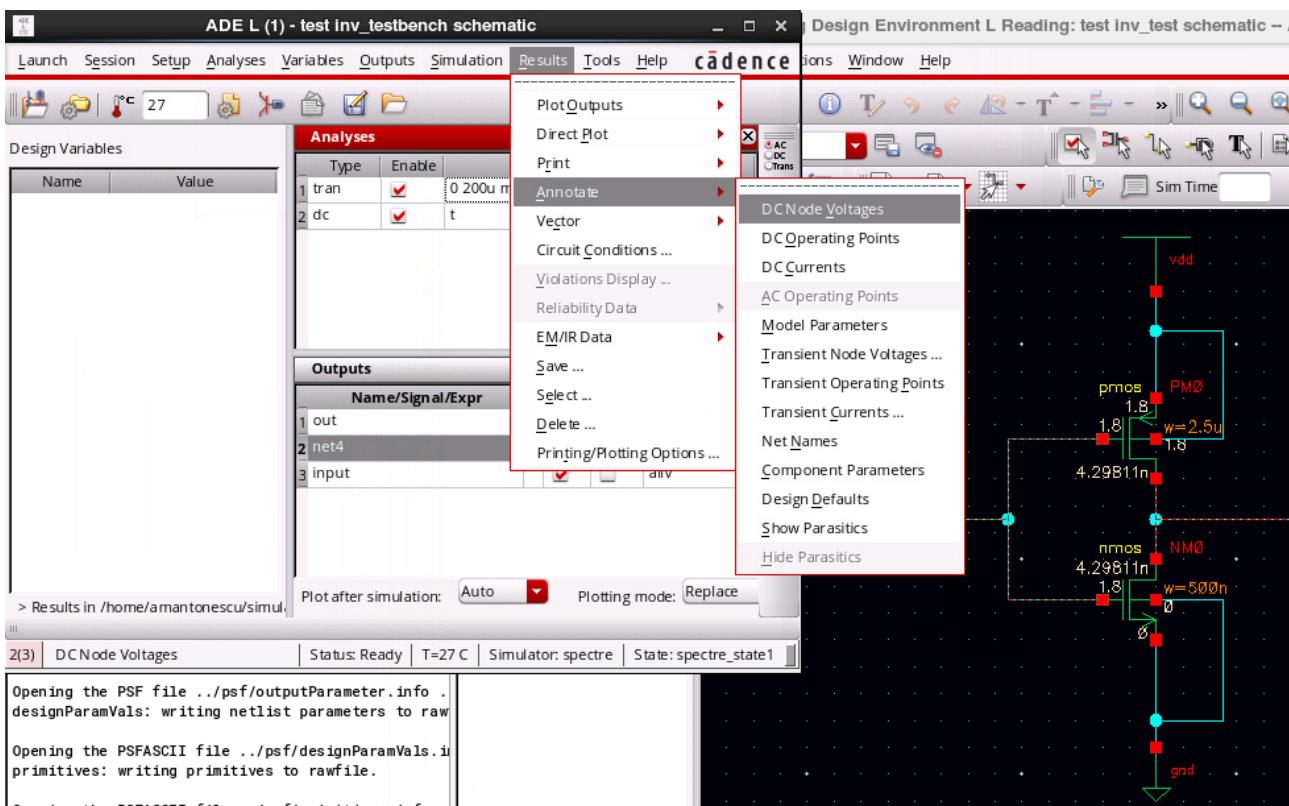


Figura A.39: Virtuoso ADEL - Results from DC analyses.

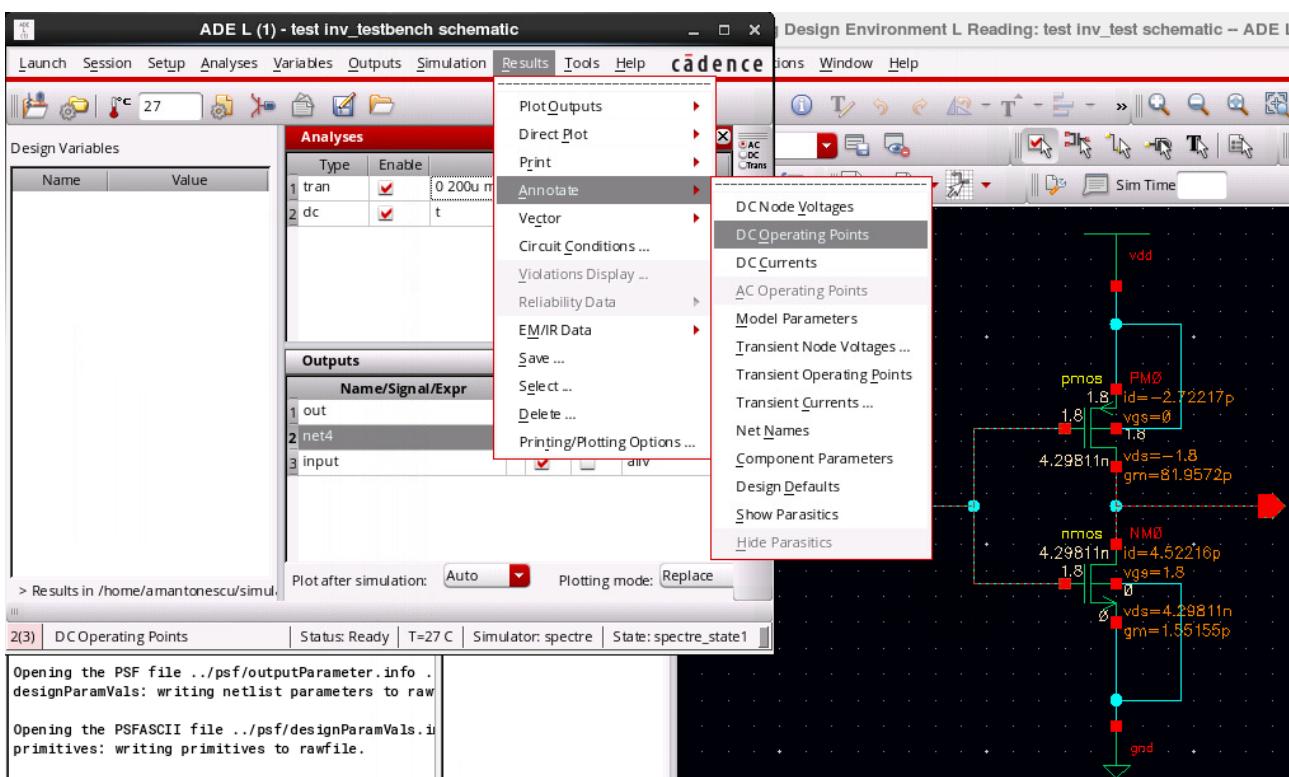


Figura A.40: Virtuoso ADEL - Results from OP analyses.